ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδίαση και Υλοποίηση Διαδικτυακής Εφαρμογής Λογισμικού ως Υπηρεσία Διαχείρισης Κρατήσεων

Δημήτριος Ράκας

Α.Μ. 711-131052

Τσίκο Γιούλι

Α.Μ. 711-131027

Εισηγητής: ΙΩΑΝΝΗΣ ΒΟΓΙΑΤΖΗΣ, Καθηγητής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΊΑΣΗ ΚΑΙ ΥΛΟΠΟΊΗΣΗ ΔΙΑΔΙΚΤΥΑΚΉΣ ΕΦΑΡΜΟΓΉΣ ΛΟΓΙΣΜΙΚΟΎ ΩΣ ΥΠΗΡΕΣΊΑ ΔΙΑΧΕΊΡΙΣΗΣ ΚΡΑΤΉΣΕΩΝ

Δημήτριος Ράκας

Α.Μ. 711-131052

Τσίκο Γιούλι

Α.Μ. 711-131027

Εισηγητής:

ΙΩΑΝΝΗΣ, ΒΟΓΙΑΤΖΗΣ, ΚΑΘΗΓΗΤΗΣ

Ioannis Voyiatzis
Digitally signed by Ioannis Voyiatzis
Date: 2023.03.21 14:58:45 +02'00'

Εξεταστική Επιτροπή:

ΝΙΚΗΤΑΣ ΚΑΡΑΝΙΚΟΛΑΣ, ΚΑΘΗΓΗΤΗΣ

Nikitas Karanikolas
Digitally signed by Nikitas Karanikolas
Date: 2023.03.22 21:16:59 +02'00'

ΚΛΕΙΩ ΣΓΟΥΡΟΠΟΥΛΟΥ, ΚΑΘΗΓΗΤΡΙΑ

Kleio Sgouropoulou
Digitally signed by Kleio Sgouropoulou
Date: 2023.03.23 11:57:12 +02'00'

Ημερομηνία εξέτασης 8/3/2023

Η διπλωματική εργασία υποβλήθηκε στο τμήμα Μηχανικών Πληροφορικής και Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής ως τμήμα του πενταετούς (5) προπτυχιακού προγράμματος σπουδών.

Αθήνα, Ελλάδα, Μάρτιος 2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΩΝ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι **Ράκας Δημήτριος** του Αλκιβιάδη με Α.Μ. **711-131052** και **Γιούλι Τσίκο** του Γκλιγκόρ με Α.Μ. **711-131027** φοιτητές του τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνουμε ότι:

«Βεβαιώνουμε ότι είμαστε συγγραφείς της παρούσας διπλωματικής εργασίας και ότι έχουμε αναφέρει ή παραπέμψει σε αυτή, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά ειδικά για την συγκεκριμένη διπλωματική εργασία»

Ευχαριστίες

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, αυτό της ανάπτυξης λογισμικού μέσω μιας διαδικτυακής εφαρμογής που αφορά τις ηλεκτρονικές κρατήσεις για χώρους ψυχαγωγίας. Την προσπάθεια μας αυτή υποστήριξε ο επιβλέπων καθηγητής μας, τον οποίο θα θέλαμε να ευχαριστήσουμε. Ακόμα, θέλαμε να ευχαριστήσουμε τις οικογένειες μας για τη συμπαράσταση κατά τη διάρκεια των σπουδών μας.

Περίληψη

Η παρούσα διπλωματική εργασία με θέμα "*Σχεδίαση και Υλοποίηση Διαδικτυακής εφαρμογής Λογισμικού ως Υπηρεσία Διαχείρισης Κρατήσεων*" εκπονήθηκε στα πλαίσια του προγράμματος σπουδών του τμήματος Μηχανικών Πληροφορικής και Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής και εμπίπτει στα γνωστικά αντικείμενα των μαθημάτων δικτυακού προγραμματισμού, αλληλεπίδραση ανθρώπου Η/Υ, βάσεων δεδομένων και τεχνολογιών λογισμικού. Σκοπός της διπλωματικής είναι η σχεδίαση και υλοποίηση ενός ολοκληρωμένου συστήματος κρατήσεων που απευθύνεται σε επαγγελματικούς κλάδους υπηρεσιών όπου οι κρατήσεις αποτελούν βασικό εργαλείο για την παροχή των υπηρεσιών τους. Τέτοια παραδείγματα είναι χώροι ψυχαγωγίας, κινηματογράφοι και άλλες δραστηριότητες. Η εφαρμογή υποστηρίζει όλες τις χρήσεις ενός σύγχρονου συστήματος κρατήσεων, προσθέτοντας σε αυτές τις ιδιαιτερότητες ενός εκ των προαναφερθέντων κλάδων: των χώρων ψυχαγωγίας τύπου δωματίων απόδρασης. Ειδικότερα, υποστηρίζει τη βασική λειτουργία, δηλαδή τις κρατήσεις προκαθορισμένου ωραρίου, την αυτόματη επικοινωνία με τον πελάτη ως προς τις κρατήσεις μέσω email και δίνεται η δυνατότητα στον διαχειριστή να ελέγχει το σύστημα μέσω μιας διαχειριστικής σελίδας, στην οποία θα μπορεί να επεξεργάζεται τις υπηρεσίες, τις κρατήσεις και συνολικότερα την λειτουργία της επιχείρησης. Η εφαρμογή επιτρέπει την επίλυση του προβλήματος των συστημάτων κρατήσεων ως προς την εναλλαγή των ωραρίων λειτουργίας τους, τη δημιουργία εφαρμογής ως διεπαφή για τις κρατήσεις που θα πραγματοποιούν οι πελάτες και τη δυνατότητα τοποθέτησης του σε παντός τύπου ιστοσελίδες. Τέλος, η ενημέρωση του ημερολογίου κρατήσεων γίνεται σε πραγματικό χρόνο για την αποφυγή διπλών κρατήσεων. Παρότι προηγούμενες προσεγγίσεις συστημάτων κρατήσεων παρουσιάζουν λύσεις για κάποια από τα επιμέρους αυτά θέματα, από την έρευνά μας, προέκυψε ότι ο συνδυασμός των παραπάνω σε μία διαδικτυακή εφαρμογή αποτελεί καινοτομία για την αγορά. Η εφαρμογή σχεδιάστηκε και αναπτύχθηκε βασιζόμενη στις αρχές τεχνολογίας λογισμικού, καθώς και στη χρήση βέλτιστων προτεινόμενων πρακτικών και μεθόδων σε όλο το εύρος της (συντηρησιμότητα, επεκτασιμότητα, σχεδίαση αρχιτεκτονικής, δοκιμών και εμπειρίας χρήστη). Τέλος, η εφαρμογή που παρουσιάζεται είναι το ελάχιστο βιώσιμο προϊόν, δηλαδή φέρει τα ελάχιστα δυνατά χαρακτηριστικά για να είναι λειτουργική με σκοπό την σταδιακή ανάπτυξη και αξιοποίηση της. Οι βασικές τεχνολογίες που χρησιμοποιήθηκαν για την σχεδίαση και την ανάπτυξη της εφαρμογής είναι: HTML, CSS, React (JavaScript), Golang, PostgreSQL, Linux, Jest, Nginx.

Abstract

This diploma thesis on "*Design and Implementation of a Web-based Software Application as a Reservation Management Service*" was prepared within the framework of the curriculum of the Department of Computer Engineering and Computer Science of the University of West Attica and falls under the subjects of web programming, human-computer interaction, databases and software engineering.The aim of the thesis is the design and implementation of an integrated reservation system addressed to professional service industries where reservations are a basic tool for the provision of their services. Such industries are entertainment venues, cinemas and other activities. The application supports all the functions of a modern booking system adding to them the specificities of one of the aforementioned sectors: the Escape rooms. More specifically, it supports the basic function of predefined time reservations and automatic communication with the customer regarding the reservations via email. It also gives the administrator the possibility to control the system through an administration page that can process the services, reservations and the booking process. The innovations of the application include the solution of the problem of reservation systems regarding the alternation of their operating hours, the creation of an application as an interface for the reservations made by customers and the possibility of placing it on all types of websites. Additionally, another innovation will be the use of technology to update the booking calendar in real time to avoid double-bookings. The application was designed and developed based on the principles of modern technologies and system security, as well as the use of best recommended practices and methods throughout the application (maintainability, scalability, architecture design, testing and user experience). Finally, the application that is presented is the minimum viable product, i.e. it carries the least possible features to be functional for the purpose of incremental deployment and exploitation. The main technologies used to design and develop the application are HTML, CSS, JavaScript, ReactJS, Golang, PostgreSQL, Linux OS, Jest, Nginx.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Δικτυακός Προγραμματισμός, Αλληλεπίδραση ανθρώπου Η/Υ, Βάσεις Δεδομένων, Τεχνολογία Λογισμικού.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: MVP, Web Services, REST, SSE, HTTP, SaaS, Booking system.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος παραρτήματος

Συντομογραφίες

Συντομογραφία	Ορολογία
UI	User Interface
UX	User Experience
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
CRM	Customer Relationship Management
CRS	Central Reservation System
SLA	Service Level Agreement
SDK	Software Development Kit
SRS	Software Requirements Specification
MVC	Model View Controller
XP	Extreme Programming
DNS	Domain Name System
SSH	Secure Shell
URL	Uniform Resource Locator
DOM	Document Object Model
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
IIS	Internet Information Services
DVCS	Distributed Version Control System
API	Application Programming Interface
RESTful	Representational State Transfer

Κεφάλαιο 1ο

Εισαγωγή

Η αυξανόμενη ανάγκη για βελτίωση της ποιότητας ζωής, έχει προκαλέσει παγκοσμίως την ανάπτυξη του τριτογενούς τομέα παραγωγής. Σε αυτές τις συνθήκες, οι πάροχοι υπηρεσιών αντιμετωπίζουν συνεχώς αυξανόμενο αριθμό καταναλωτών. Καθώς δεν μπορούν να εξυπηρετήσουν όλους τους πελάτες ταυτόχρονα, δημιουργείται ουρά αυτών που αναμένουν να εξυπηρετηθούν. Μια λύση για τη βελτίωση της εξυπηρέτησης αυτών των πελατών είναι η δημιουργία συστημάτων κρατήσεων, ώστε οι πάροχοι υπηρεσιών να μπορούν να προγραμματίζουν την εξυπηρέτησή τους και να μειώνεται η αναμονή.

Ταυτόχρονα, η ανάπτυξη και διάδοση του διαδικτύου έχουν ανοίξει νέες προοπτικές για καταναλωτές και παρόχους υπηρεσιών. Οι μεν έχουν πλέον τη δυνατότητα να αναζητήσουν και να επιλέξουν ανάμεσα σε πολλούς παρόχους υπηρεσιών τις υπηρεσίες που επιθυμούν, ενώ οι δε μπορούν να απευθύνονται σε ένα πιο ευρύ κοινό. Προκύπτει, λοιπόν, η ανάγκη για ένα σύστημα κρατήσεων το οποίο να επιτρέπει στους παρόχους να διαχειρίζονται περισσότερους πελάτες, αποδοτικότερα. Τέτοια ζητήματα έχουν επιλυθεί και προηγουμένως, όμως οι ιδιαιτερότητες στη λειτουργία του εκάστοτε τύπου επιχείρησης απαιτούν πιο στοχευμένες λύσεις.Αντίστοιχα, εντοπίζεται αυτη η ανάγκη στον τομέα χώρων ψυχαγωγίας όπως τα δωμάτια απόδρασης, ένα νέο κλάδο στην παροχή υπηρεσιών, με την επιπρόσθετη διάσταση του προκαθορισμένου ωραρίου προγράμματός τους, λόγω της φύσης της λειτουργίας τους.

Στο πλαίσιο αυτό, η διπλωματική εργασία αποσκοπεί στο σχεδιασμό μιας ψηφιακής εφαρμογής οργάνωσης κρατήσεων που θα διατίθεται ώς προϊόν σε παρόχους υπηρεσιών δωματίων απόδρασης και άλλων παρόμοιας λειτουργίας, λαμβάνοντας υπ’ όψιν τις ιδιομορφίες τους.

1.1 Συστήματα Κρατήσεων

1.1.1 Ορισμός συστήματος κρατήσεων

Τα συστήματα κρατήσεων είναι ηλεκτρονικά συστήματα που χρησιμοποιούνται για την αποθήκευση και την ανάκτηση πληροφοριών, τη διεξαγωγή συναλλαγών που σχετίζονται με αεροπορικά ταξίδια, ξενοδοχεία, ενοικιάσεις αυτοκινήτων, δραστηριοτήτων ψυχαγωγίας και πολλές άλλες δραστηριότητες. Αρχικά σχεδιάστηκαν και λειτουργούσαν για αεροπορικές εταιρείες, ενώ αργότερα τα συστήματα αυτά επεκτάθηκαν για χρήση από ταξιδιωτικά γραφεία και παγκόσμια συστήματα διανομής για κρατήσεις και πωλήσεις εισιτηρίων. Οι περισσότερες επιχειρήσεις παροχής υπηρεσιών έχουν αναθέσει τα συστήματα κρατήσεων τους σε εξωτερικούς παρόχους που επιτρέπουν την πρόσβαση των πελατών τους μέσω του διαδικτύου. Τα σύγχρονα συστήματα κρατήσεων συνήθως επιτρέπουν στους χρήστες να κάνουν κράτηση δωματίων ξενοδοχείων, χώρων ψυχαγωγίας, ενοικιαζόμενων αυτοκινήτων, αεροπορικών εισιτηρίων καθώς και άλλων δραστηριοτήτων και περιηγήσεων. Χρησιμοποιούνται επίσης για τη μετάδοση ηλεκτρονικών πληροφοριών στους χρήστες και διασφαλίζοντας ότι δεν υπάρχουν υπεράριθμες κρατήσεις. Ο κύριος στόχος ενός συστήματος κρατήσεων είναι να δημιουργήσει μια υπηρεσία ενιαίας εξυπηρέτησης και να εξαλείψει τις φυσικές και γεωγραφικές αποστάσεις μεταξύ διαμεσολαβητών και καταναλωτών.

1.1.2 Ιστορική αναδρομή στα συστήματα κρατήσεων

Πριν από την ανάπτυξη των συστημάτων κρατήσεων, οι ενδιαφερόμενοι πελάτες έπρεπε να εξαρτώνται από τις πληροφορίες που παρέχουν οι προμηθευτές μέσω εντύπων, μπροσούρων, φυλλαδίων και καταχωρήσεων σε τοπικούς και περιφερειακούς ταξιδιωτικούς οδηγούς. Ως αποτέλεσμα, το διαφημιστικό υλικό ήταν δαπανηρό, απαιτούσε αρκετή εργασίας και οι πληροφορίες παρέμειναν στατικές όταν τα δεδομένα έπρεπε να αλλάζουν συχνά. Για να διευκολυνθεί μια ομαλή και δυναμική ροή πληροφοριών, το πρώτο κεντρικό σύστημα κρατήσεων (CRS) εισήχθη ως πείραμα τη δεκαετία του 1960 από αεροπορικές εταιρείες για την παρακολούθηση των πωληθέντων θέσεων. Το 1963, το SABER (Semi-Automated Business Research Environment), το πρώτο CRS στον κόσμο εισήχθη από την American Airlines. Μετά από αυτό, το CRS έγινε το κύριο μέσο διανομής πληροφοριών αεροπορικών ταξιδιών και είχε σημαντικό αντίκτυπο στον ανταγωνισμό στον τομέα των αεροπορικών εταιρειών. Το 1976, τα ταξιδιωτικά γραφεία άρχισαν να τα χρησιμοποιούν και στο εξής έγιναν ένα παγκόσμιο χαρακτηριστικό της τουριστικής βιομηχανίας. Ένα ακόμη πείραμα που

πραγματοποιήθηκε ήταν ένα μηχάνημα με προσωρινή αποθήκευση βασισμένο σε μαγνητικό τύμπανο, το Magnetronic Reservisor. Αυτό το σύστημα αποδείχθηκε επιτυχημένο και σύντομα χρησιμοποιήθηκε από πολλές αεροπορικές εταιρείες, καθώς και από τα Sheraton Hotels και την Goodyear για τον έλεγχο των αποθεμάτων.[3]

1.1.3 Βασικές λειτουργίες ενός συστήματος κρατήσεων

- **Ενημερώσεις πελατών μέσω E-mails και SMS**: Εκτός από την αποστολή τακτικής επιβεβαίωσης κράτησης και μιας απόδειξης πληρωμής, η αυτοματοποίηση email/SMS μπορεί να χρησιμοποιηθεί για προωθητικούς σκοπούς, εξατομικευμένες προσφορές και ενθαρρυντικά σχόλια.
- **Διαχείριση ακυρώσεων**: Αυτή η δυνατότητα επιτρέπει τον χειρισμό ακυρώσεων, την ενημέρωση της διαθεσιμότητας των δωματίων σε όλα τα κανάλια και την πιθανή αυτοματοποίηση της διαδικασίας επιστροφής χρημάτων.
- **Διαχείριση λίστας αναμονής**
- **Διασύνδεση με άλλα συστήματα**
- **Παρουσίαση των προσφερόμενων υπηρεσιών με τιμές και εικόνες**
- **Διαχείριση κρατήσεων και συγχρονισμός**
- **Φιλικό προς τον χρήστη σύστημα κρατήσεων**
- **Ηλεκτρονικές πληρωμές**
- **Διαχείριση πελατολογίου**

1.1.4 Ηλεκτρονικό Σύστημα κρατήσεων

Ένα ηλεκτρονικό σύστημα κρατήσεων είναι ένα λογισμικό που χρησιμοποιείται για τη διαχείριση κρατήσεων για τις εκάστοτε υπηρεσίες. Είτε πρόκειται για γυμναστήριο, χώρους απόδρασης και κέντρα αναψυχής, ένα ηλεκτρονικό σύστημα κρατήσεων επιτρέπει σε επιχειρήσεις παροχής υπηρεσιών να δέχονται κρατήσεις και ραντεβού διαδικτυακά και να διαχειρίζονται εύκολα τις τηλεφωνικές και προσωπικές κρατήσεις τους. Πιο συγκεκριμένα, ένα ηλεκτρονικό σύστημα κρατήσεων επιτρέπει σε έναν πιθανό πελάτη να κάνει κράτηση και να πληρώσει για μια υπηρεσία απευθείας μέσω ενός ιστότοπου. Αυτό σημαίνει ότι από τη στιγμή που ένας πελάτης αποφασίσει ότι θέλει να κλείσει μια θέση για την υπηρεσία μέχρι την επιλογή ημερομηνίας, την επιλογή ώρας και την πληρωμή για την κράτηση, όλα διεκπεραιώνονται διαδικτυακά και επομένως τα συστήματα αυτά μειώνουν σημαντικά τον φόρτο εργασίας του προσωπικού και επιλύουν προβλήματα όπως οι διπλοκρατήσεις.

Τα βασικότερα πλεονεκτήματα χρήσης των ηλεκτρονικών συστημάτων κράτησης:

- **Αυξάνει τις κρατήσεις**: Ένα από τα σημαντικότερα πλεονεκτήματα ενός διαδικτυακού συστήματος κρατήσεων είναι ότι είναι πάντα διαθέσιμο. Μπορεί να δέχεται κρατήσεις 24 ώρες το 24ωρο, 7 ημέρες την εβδομάδα, ώστε οι πελάτες να μην χρειάζεται να περιμένουν την επόμενη μέρα για να κάνουν κράτηση ή ακόμα χειρότερα, να απευθυνθούν σε έναν από τους ανταγωνιστές. Οι πελάτες έχουν την ευκολία να κάνουν κράτηση όταν αυτοί το επιλέξουν, από όπου αυτοί το επιλέξουν και η επιχείρηση να συλλέγει δεδομένα καθόλη τη διάρκεια της ημέρας.
- **Εξοικονομεί χρήματα**: Το ηλεκτρονικό σύστημα κρατήσεων βοηθά να γίνονται περισσότερα με λιγότερο κόστος, αναθέτοντας τις εργασίες που σχετίζονται με τις κρατήσεις σε μια αυτοματοποιημένη πλατφόρμα. Ένα σύστημα κρατήσεων δίνει τη δυνατότητα να αυτοματοποιηθούν οι ειδοποιήσεις προς τους πελάτες. Όλες αυτές οι ενέργειες εκτελούνται στο παρασκήνιο, ώστε το προσωπικό να μπορεί να εστιάσει στην παροχή καλύτερης εμπειρίας και υπηρεσιών στους πελάτες.
- **Αυξάνει την αποδοτικότητα**: Οι πελάτες σήμερα περιμένουν άμεση ικανοποίηση και ένα ηλεκτρονικό σύστημα κρατήσεων μπορεί να βοηθήσει προς αυτή την κατεύθυνση. Το λογισμικό κρατήσεων μπορεί να βοηθήσει στην διαχείριση της λίστα αναμονής που ενδεχομένως μπορεί να δημιουργηθεί για τις προσφερόμενες υπηρεσίες της επιχείρησης.
- **Ανάλυση των δεδομένων**: Ένα λογισμικό κρατήσεων παρακολουθεί κάθε κράτηση και κάθε λεπτομέρεια που την συνοδεύει. Οι αναφορές αυτές μπορούν να βοηθήσουν ώστε να γίνει πιο διακριτό το τι είδους κρατήσεις αποφέρουν τα περισσότερα χρήματα και να κατανοηθεί καλύτερα ποιος είναι ο τυπικός - "*καλός*" πελάτης. Ένα διαδικτυακό λογισμικό κρατήσεων δίνει τη δυνατότητα να λαμβάνονται αποφάσεις βάσει δεδομένων, πράγμα πολύ σημαντικό καθώς ζούμε την εποχή της πληροφορίας.[4]

1.1.5 Συστήματα κρατήσεων στην Covid εποχή

Στις αρχές του 2020 ξέσπασε σε παγκόσμιο επίπεδο η γνωστή σε όλους μας πανδημία COVID-19, σε αυτή την περίοδο επιβλήθηκαν περιορισμοί κινήσεων, συναθροίσεων ακόμη και επιχειρηματικής δραστηριότητας στον τομέα της εστίασης και ψυχαγωγίας. Αυτή τη περίοδο, η χρήση της τεχνολογίας και η μετάβαση σε αυτή έγινε πιο έντονη από ποτέ καθώς ήταν επιβεβλημένη ανάγκη. Στις περισσότερες εργασίες που μπορούσαν να γίνουν εξ αποστάσεως εφαρμόστηκε αυτή η μορφή εργασίας, τα ψώνια του σπιτιού γινόντουσαν ακόμα και με ηλεκτρονικές παραγγελίες και το πιο σημαντικό: οποιαδήποτε επίσκεψη σε κάποιον οργανισμό ή εταιρεία μπορούσε να πραγματοποιηθεί μέσω ραντεβού. Τα συστήματα

κρατήσεων έγιναν αναγκαιότητα και εγκαταστάθηκαν σχεδόν παντού. Χρησιμοποιούνται μέχρι και σήμερα, καθώς αποδείχτηκε η χρησιμότητα και βολικότητα τους τόσο για τις επιχειρήσεις όσο και για τους πελάτες. Στην αρχική περίοδο της πανδημίας έγιναν έντονες διεργασίες και δουλειά από εταιρίες πληροφορικής για να αναπτύξουν γρήγορα συστήματα κρατήσεων, συνήθως προκαθορισμένων ωραρίων, ώστε να αποφεύγεται ο συνωστισμός του κόσμου για την μείωση της έξαρσης της πανδημίας. Μετά από μία δύσκολη περίοδο όπου η αρωγή της τεχνολογίας συνέβαλε καθοριστικά στην κοινωνική προστασία και στη βελτίωση παροχής υπηρεσιών σε όλους σχεδόν τους τομείς, μας έδειξε το δρόμο για την αναζήτηση χρήσιμων και βέλτιστων πρακτικών για να βελτιώνουμε την ζωή μας καθημερινά. Με το πέρασμα των δεκαετιών, η τεχνολογία γίνεται αναπόσπαστο κομμάτι της ζωής μας και η παροχή υπηρεσιών μία εύκολη διαδικασία που προσφέρει τρομερά πλεονεκτήματα σε σχέση με τις παλιές κλασσικές μεθόδους (τηλεφωνικά ή δια ζώσης). Τα συστήματα κρατήσεων ήρθαν για να μείνουν στην καθημερινότητα μας, να βελτιώνονται συνεχώς και να προσφέρουν ολοένα και περισσότερα στους χρήστες για την παροχής της βέλτιστης δυνατής εμπειρίας.

1.1.6 Πρόβλημα διπλοκρατήσεων

Ένα από τα κυριότερα προβλήματα που καλούνται να λύσουν τα συστήματα κρατήσεων είναι αυτό της διπλοκράτησης για την ίδια συνεδρία. Η εμπειρία των πελατών εξαρτάται πλήρως από την πορεία της κράτησης τους μέσω του συστήματος κρατήσεων και των ενημερωτικών μηνυμάτων που θα λαμβάνουν καθόλη τη διάρκεια της διαδικασίας. Σε αυτή την εμπειρία, ένα σύνηθες σφάλμα που παρατηρείται είναι η διπλοκράτηση μιας συνεδρίας από 2 ή παραπάνω χρήστες, πράγμα που οδηγεί σε δυσαρέσκεια των πελατών και πιθανόν μετάβαση σε ανταγωνιστές. Ένα τέτοιου είδους σφάλμα προφανώς δημιουργείται από τεχνικό λάθος και κενά στην υλοποίηση της εφαρμογής που λανθασμένα εμφανίζουν κλεισμένες συνεδρίες ως ελεύθερες ή σε ταυτόχρονη κράτηση δεν γίνεται κατάλληλη διαχείριση της κατάστασης δίνοντας την συνεδρία στον πρώτο πελάτη που την επέλεξε. Ένας τρόπος να αποφευχθούν οι διπλοκρατήσεις είναι οι χρήστες να ενημερώνονται σε πραγματικό χρόνο για την κατάσταση της διαθεσιμότητας.

1.1.7 Πρόβλημα αλλαγής ωραρίων

Ένα πρόβλημα με το οποίο ήρθαμε αντιμέτωποι κατά την σχεδίαση/υλοποίηση της εφαρμογής ήταν η εναλλαγή των ωραρίων λειτουργίας μιας υπηρεσίας καθώς και διαχείρισης των υπαρχόντων κρατήσεων με το υφιστάμενο πρόγραμμα. Κατόπιν μελέτης και σχεδίασης

της βέλτιστης επιλογής για την εναλλαγή προγράμματος, επιλέξαμε την δυνατότητα δημιουργίας μελλοντικού προγράμματος από μία ημερομηνία που θα επιλέγεται από τον διαχειριστή. Επιπλέον, αν υπάρχουν τυχόν περιορισμοί με παλαιότερες συνεδρίες και το νέο πρόγραμμα, θα εμφανίζεται κατάλληλη ένδειξη στον διαχειριστή για επίλυση του περιορισμού. Ταυτόχρονα και για όσο διάστημα θα υπάρχει αυτός ο περιορισμός μεταξύ κάποιων συνεδριών ανάμεσα στο παλιό και το νέο πρόγραμμα, δεν θα παρουσιάζονται στους χρήστες τα νέα ραντεβού που επικαλύπτονται εν μέρη με παλαιά ώστε να αποφευχθεί το πρόβλημα της διπλοκράτησης ή της μερικής επικάλυψης. Υποχρέωση του διαχειριστή θα είναι όσο το δυνατόν γρηγορότερη επίλυση των περιορισμών μέσω επικοινωνίας με τους πελάτες. Ακόμη, έχει προβλεφθεί στο γραφικό στοιχείο της εφαρμογής που θα βλέπει ο χρήστης, να δίνεται ένα εύρος 3 εβδομάδων που θα προπορεύεται από την τρέχουσα ημερομηνία για να πραγματοποιήσει μία κράτηση. Έτσι ο διαχειριστής έχει την δυνατότητα της έγκαιρης σχεδίασης του νέου προγράμματος του, ώστε να μην επικαλύπτεται χρονικά με το υπάρχον και με αυτόν τον τρόπο θα εξαλείφονται τυχόν περιορισμοί που μπορεί να υπάρξουν στην εναλλαγή ωραρίου λειτουργίας.

1.2 Διαδικτυακές εφαρμογές

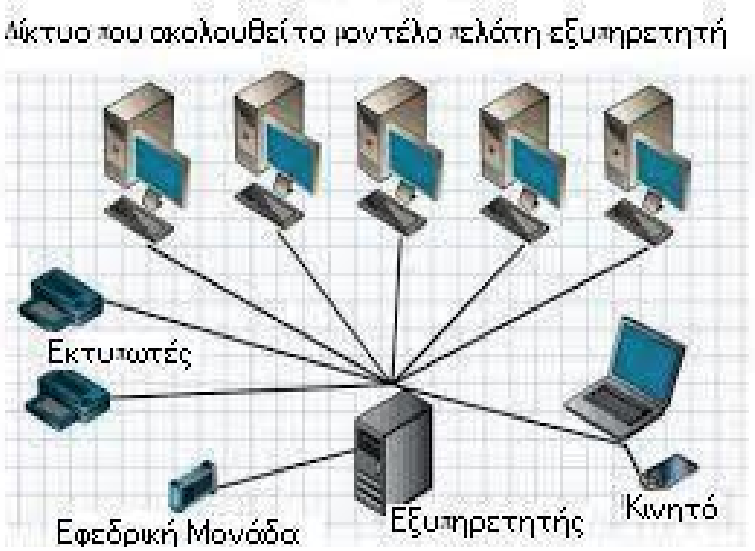
Προκειμένου να κατανοήσουμε την έννοια της διαδικτυακής εφαρμογής που αποτελεί βασικό στοιχείο της διπλωματικής εργασίας, θα πρέπει πρώτα να έχουμε αποσαφηνίσει τι ονομάζεται εφαρμογή. Εφαρμογή ονομάζεται το λογισμικό το οποίο εγκαθίσταται σε έναν ηλεκτρονικό υπολογιστή και έχει σχεδιαστεί και προγραμματιστεί έτσι ώστε να πραγματοποιεί συγκεκριμένες διεργασίες, να επιτυγχάνει συγκεκριμένους στόχους και να εξάγει στον χρήστη την επιθυμητή πληροφορία ή αποτέλεσμα. Οι βασικοί τύποι εφαρμογών είναι οι εξής:

- Τοπικές Εφαρμογές (Local Applications)
- Δικτυακές Εφαρμογές (Network Applications)
- Εφαρμογές Κινητών Συσκευών (Mobile Applications)
- Διαδικτυακές Εφαρμογές (Web Applications)

1.2.1 Ορισμός Διαδικτυακή Εφαρμογή

Διαδικτυακή εφαρμογή (Web Application) ορίζουμε μία εφαρμογή ή το λογισμικό το οποίο είναι διαθέσιμο στους χρήστες μέσω του διαδικτύου (Internet) ή του ενδοδικτύου (Intranet) ενός οργανισμού ή μιας εταιρείας και ο χρήστης μπορεί να την χρησιμοποιεί έχοντας μόνο

πρόσβαση σε έναν περιηγητή (Browser). Αυτός ο τύπος εφαρμογής χρησιμοποιεί το μοντέλο client-server, δηλαδή οι εφαρμογές εκτελούνται σε υπολογιστικές μηχανές στον διακομιστή (Server) και έχουν ως κύριο ρόλο το σημείο εξυπηρέτησης έτσι ώστε να παρέχουν τις υπηρεσίες τους σε περισσότερους εκ του ενός χρήστες. Οι διαδικτυακές εφαρμογές μπορούν να είναι γραμμένες σχεδόν σε οποιαδήποτε γλώσσα προγραμματισμού και επίσης κάνουν χρήση του πρωτοκόλλου επικοινωνίας HTTP (HyperText Transfer Protocol), το οποίο είναι η βάση της υπηρεσίας World Wide Web (WWW). Τα πρωτόκολλα επικοινωνίας του Διαδικτύου είναι σαφώς καθορισμένα και διαθέσιμα στους προγραμματιστές για την ανάπτυξη της εφαρμογής. Οι διαδικτυακές εφαρμογές εκτελούνται σε έναν διαδικτυακό διακομιστή ανεξαρτήτως της χρησιμοποιούμενης πλατφόρμα. Είναι υπεύθυνοι για την επεξεργασία των δεδομένων και την σύνταξη των αποτελεσμάτων για την εμφάνιση τους στο χρήστη, σύμφωνα με την πρότυπη γλώσσα σήμανσης εγγράφων υπερκειμένου HTML/XHTML. Από την πλευρά του πελάτη, δεν χρειάζεται να γνωρίζει τις χρησιμοποιούμενες τεχνολογίες και λεπτομέρειες στον εξυπηρετητή καθώς δεν επηρεάζουν καθόλου την εμπειρία του στην διαδικασία χρήσης της εφαρμογής.



Εικόνα 1.1 Δίκτυο μοντέλου client-server
(http://ebooks.edu.gr/ebooks/v/html/8547/2714/Pliroforiki_A-Lykeiou_html-empl/index3_9.html)

Ο πελάτης είναι επιφορτισμένος μόνο με την απεικόνιση των πληροφοριών που δέχεται όπως για παράδειγμα μέσω HTML αρχείων και δεν χρειάζεται ιδιαίτερα μεγάλες απαιτήσεις σε επεξεργαστική ισχύ και μνήμη. Ο πελάτης λαμβάνει την απαραίτητη πληροφορία που μπορεί να απεικονιστεί από πληθώρα περιηγητών, δηλαδή προσωπικούς υπολογιστές, κινητές συσκευές κλπ.

Τα κυριότερα πλεονεκτήματα των διαδικτυακών εφαρμογών είναι:

- **Συμβατότητα**: Ένα από τα βασικά πλεονέκτημα των διαδικτυακών εφαρμογών είναι η συμβατότητα με κάθε τύπου λειτουργικό σύστημα. Η βασική απαίτηση της εφαρμογής είναι ένας περιηγητής και αυτό είναι το χαρακτηριστικό που δίνει την ευελιξία στις διαδικτυακές εφαρμογές να εκτελούνται σε παντός τύπου λειτουργικό σύστημα. Φυσικά αυτή η δυνατότητα έρχεται να υποστηριχθεί και από τις τεχνολογίες που πλαισιώνουν αυτού του είδους εφαρμογές, δηλαδή προτυποποιημένες γλώσσες προγραμματισμού.
- **Χαμηλή κατανάλωση σε πόρους και χώρο**: Ως συνέχεια του παραπάνω και εφόσον οι διαδικτυακές εφαρμογές δεν εκτελούνται στον ηλεκτρονικό υπολογιστή του χρήστη αλλά μόνο απεικονίζουν την πληροφορία που ζητά ο χρήστης δεν καταναλώνουν και ιδιαίτερα μεγάλους πόρους. Με την ίδια λογική οι διαδικτυακές εφαρμογές δεν καταναλώνουν καθόλου ή σχεδόν καθόλου χώρο στο δίσκο του χρήστη, αφού το σύνολο της εφαρμογής και των διαδικασιών που τρέχουν είναι αποθηκευμένο στον εξυπηρετητή. Μόνο κατά την επικοινωνία του πελάτη με τον εξυπηρετητή γίνεται μεταφορά δεδομένων προς την υπολογιστική μονάδα του χρήστη.
- **Προσβασιμότητα**: Οι χρήστες των διαδικτυακών εφαρμογών έχουν άμεση προσβασιμότητα στις εφαρμογές που θέλουν να χρησιμοποιήσουν από οποιονδήποτε υπολογιστή, κινητό ή τάμπλετ χωρίς την εγκατάσταση κάποιου επιπρόσθετου λογισμικού. Η μόνη απαίτηση που πρέπει ο χρήστης να διαθέτει είναι ο περιηγητής, ο οποίος είναι προεγκατεστημένος σε όλα τα λειτουργικά συστήματα ακόμα και στις φορητές συσκευές. Η ιδιότητα αυτή των διαδικτυακών εφαρμογών είναι ιδιαίτερα σημαντική για μεγάλες επιχειρήσεις με πολλούς χρήστες και όχι μόνο, καθώς αν λάβουμε υπόψιν μας την περίπτωση μιας τοπικής εφαρμογής χρειάζεται να εγκατασταθεί σε κάθε ένα υπολογιστή ξεχωριστά. Επιπρόσθετα, οι χρήστες των διαδικτυακών εφαρμογών μπορούν να τις χρησιμοποιούν ακόμα και αν δεν έχουν φυσική παρουσία στο κατάστημα, την επιχείρηση ή το χώρο εργασίας τους. Η δυνατότητα αυτή δίνει ευελιξία στους χρήστες ώστε να χρησιμοποιούν τις εφαρμογές οπουδήποτε αυτοί επιθυμούν επιτρέποντας τους ακόμα και να εργάζονται, να παραγγέλνουν, να κάνουν κράτηση από απομακρυσμένες περιοχές ή το σπίτι τους.
- **Γρήγορη αναβάθμιση**: Ένα βασικό πλεονέκτημα συγκριτικά με τις τοπικές/δικτυακές εφαρμογές εμφανίζεται στις περιπτώσεις όπου η εφαρμογή χρειάζεται κάποια αναβάθμιση ή αλλαγές λόγω κάποιου σφάλματος. Σε μια κλασική τοπική/δικτυακή

εφαρμογή η αναβάθμιση του συστήματος θα πρέπει να γίνει σε κάθε έναν υπολογιστή ξεχωριστά, το οποίο προϋποθέτει επιπλέον κόστος σε χρόνο και χρήμα. Αντιθέτως, σε μια διαδικτυακή εφαρμογή η αναβάθμιση πραγματοποιείται μόνο στον εξυπηρετητή που φιλοξενεί την εφαρμογή και το αναβαθμισμένο πρόγραμμα είναι διαθέσιμο ταυτόχρονα σε όλους τους χρήστες. Με τον τρόπο αυτό εξοικονομείται χρόνος ο οποίος είναι ιδιαίτερα πολύτιμος κυρίως για τις μεγάλες επιχειρήσεις, ενώ ως συνέπεια του παραπάνω το κόστος της αναβάθμισης είναι μειωμένο κατά ένα μεγάλο βαθμό .

- **Δυνατότητα χρήσης και εκτός δικτύου**: Ένα ακόμα πλεονέκτημα των σύγχρονων διαδικτυακών εφαρμογών (εφαρμογές με χρήση HTML5) είναι η δυνατότητα της εκτός διαδικτύου χρήσης μιας διαδικτυακής εφαρμογής με την προϋπόθεση ότι η εφαρμογή έχει κατασκευαστεί με ανάλογο τρόπο. Εφόσον έχει γίνει πρόβλεψη η ανάπτυξη της εφαρμογής να υποστηρίζει και αυτή την δυνατότητα, ο χρήστης μπορεί να χρησιμοποιεί κανονικά την εφαρμογή παρά την έλλειψη σύνδεσης στο δίκτυο. Αυτή η δυνατότητα μπορεί να επιτευχθεί αποθηκεύοντας ένα αντίγραφο από τα αρχεία που είναι απαραίτητα για χρήση της εφαρμογής και εκτός διαδικτύου.

Τα κυριότερα μειονεκτήματα των διαδικτυακών εφαρμογών είναι:

- **Χρήση της εφαρμογής εκτός διαδικτύου**: Ένα μειονέκτημα που ταυτόχρονα είναι και πλεονέκτημα είναι η χρήση της εφαρμογής εκτός διαδικτύου όπως επεξηγήθηκε και παραπάνω. Προκειμένου να γίνει αυτό εφικτό, θα πρέπει να έχει γίνει πρόβλεψη κατά την σχεδίαση της εφαρμογής καθώς επίσης να υποστηρίζεται και από τους περιηγητές. Σε περίπτωση που κάτι τέτοιο δεν έχει πραγματοποιηθεί, τότε η εφαρμογή δεν είναι εφικτό να χρησιμοποιηθεί χωρίς την σύνδεση του χρήστη με το διαδίκτυο ή το ενδοδίκτυο της εταιρείας/οργανισμού.
- **Ασυμβατότητα όλων των περιηγητών**: Ένα ακόμη μειονέκτημα που αφορά την τελευταία έκδοση της HTML είναι η μη πλήρης συμβατότητα των περιηγητών με αυτή την έκδοση. Αν και τα πλεονεκτήματα και οι δυνατότητες της HTML5 είναι πολλά αρκετοί από τους περιηγητές δεν είναι ακόμα πλήρως συμβατοί με αυτά. Επίσης, σε περίπτωση που δεν έχει προβλεφθεί η μη λειτουργία κάποιου χαρακτηριστικού της εφαρμογής σε κάποιον περιηγητή, αυτό μπορεί να δημιουργήσει προβλήματα με αποτέλεσμα να μην λειτουργεί σωστά ή να μην λειτουργεί καθόλου. Για το λόγο αυτό ο κατασκευαστής της εφαρμογής με τον πελάτη πρέπει από κοινού να αποφασίζουν ποιος περιηγητής θα είναι ο προτεινόμενος για την εφαρμογή αλλά ταυτόχρονα να προβλέπεται και η περίπτωση χρήσης άλλων περιηγητών..

- **Άμεση αναβάθμιση**: Ένα επιπλέον χαρακτηριστικό των διαδικτυακών εφαρμογών είναι η ταυτόχρονη αναβάθμιση της εφαρμογής. Οι χρήστες δεν ερωτώνται για την αναβάθμιση του συστήματος που γίνεται από την εταιρεία που συντηρεί την εφαρμογή ελλοχεύοντας έτσι κινδύνους σφαλμάτων που μπορεί να υπάρχουν στο αναβαθμισμένο προϊόν να εκτεθούν στους χρήστες. Δηλαδή δεν δίνεται στις περισσότερες περιπτώσεις η δυνατότητα σταδιακής αναβάθμισης για τον περιορισμό τέτοιων ανεπιθύμητων καταστάσεων. Η νέα έκδοση της εφαρμογής εφαρμόζεται καθολικά, οπότε οι κατασκευαστές της εφαρμογής οφείλουν να έχουν εξασφαλίσει μέσω εντατικών δοκιμών την άρτια λειτουργία της νέας έκδοσης.
- **Ασυμβατότητα τμημάτων της εφαρμογής με ενημερώσεις περιηγητών**: Τέλος, μια πιθανή μη συμβατότητα κάποιων χαρακτηριστικών της εφαρμογής με μια μελλοντική έκδοση του περιηγητή μπορεί να οδηγήσει σε αρνητικές καταστάσεις. Αυτό συμβαίνει συχνά διότι πολλές εταιρείες που παρέχουν τους περιηγητές, πραγματοποιούν την αναβάθμιση τους και σταματούν να υποστηρίζουν χαρακτηριστικά της παλαιότερης έκδοσης. Άμεσο αποτέλεσμα έχει η δυσλειτουργία της εφαρμογής σε μεμονωμένα κομμάτια ή και ολόκληρης της εφαρμογής δίνοντας στον χρήστη μια αρνητική εμπειρία που μπορεί να προκαλέσει την δυσφορία του χρήστη.

1.2.2 Επιθυμητά χαρακτηριστικά

- **Ευκολία στη χρήση**: Η πλοήγηση των χρηστών σε μία διαδικτυακή εφαρμογή θα πρέπει να γίνεται με την μέγιστη δυνατή ευκολία. Μεγάλο μέρος τον χρηστών ακόμη και σήμερα δεν είναι πλήρως εξοικειωμένο με την χρήση διαδικτυακών εφαρμογών και ενδεχομένως κάποιες πιο πολύπλοκες διαδικασίες να οδηγούν σε δυσαρέσκεια των χρηστών. Καλό είναι να ακολουθούνται γενικοί κανόνες και μοτίβα που οι χρήστες έχουν συνηθίσει με την μέχρι τώρα εμπειρία τους, όπως σύνδεσμοι, φόρμες, κουμπιά κλπ. Επιπλέον, θα ήταν πολύ χρήσιμη και μια σελίδα βοήθειας που θα δίνει γενικές οδηγίες χρήσης της εφαρμογής και θα απαντάει στα συχνότερα ερωτήματα των χρηστών (FAQ).
- **Ασφάλεια**: Ένα τεράστιο κεφάλαιο στις εφαρμογές αποτελεί η ασφάλεια τους, τόσο των προσωπικών δεδομένων των χρηστών αλλά και πιθανών χρηματικών ποσών που μπορεί να διακινούνται μέσω ηλεκτρονικών πληρωμών σε διάφορα είδη εφαρμογών. Μια εφαρμογή θα πρέπει να εξασφαλίζει στον χρήστη την εγκυρότητα και ασφάλεια των προσωπικών του δεδομένων μέσω τεχνικών κρυπτογράφησης,

ώστε να μην δύναται κανείς άλλος να έχει πρόσβαση στα στοιχεία του. Οι εφαρμογές πλέον θα πρέπει να εναρμονίζονται πλήρως με τους ευρωπαϊκούς κανόνες για τα προσωπικά δεδομένα (GDPR) καθώς και να έχουν τα κατάλληλα συστήματα ώστε σε πιθανό περιστατικό παραβίασης των δεδομένων να δράσουν έγκαιρα. Όλες οι διαδικτυακές εφαρμογές θα πρέπει να διατηρούν συστήματα ασφαλείας, να τα συντηρούν και να είναι όσο το δυνατόν πιο αυτοματοποιημένα γίνεται ώστε να έχει ο ανθρώπινος παράγοντας τη μικρότερη δυνατή πρόσβαση σε αυτά.

- **Ταχύτητα**: Η ταχύτητα μιας εφαρμογής παίζει καταλυτικό ρόλο στην εμπειρία που θα αποκομίσουν οι χρήστες της, είτε θετική είτε αρνητική. Η ταχύτητα με την οποία θα πρέπει να φορτώνει μια εφαρμογή δεν θα πρέπει να ξεπερνά τα 3s καθώς μετά παρατηρείται η φυγή των χρηστών σε άλλους ιστότοπους και εφαρμογές. Οι εφαρμογές θα πρέπει να είναι σχεδιασμένες και δομημένες με τέτοιο τρόπο έτσι ώστε να μην απαιτείται μεγάλο χρονικό διάστημα για την αρχική φόρτωση της εφαρμογής αλλά και κατά την χρήση της εξίσου. Για τον σκοπό αυτό υπάρχουν βέλτιστες πρακτικές όπου η εφαρμογή καλεί μόνο την πληροφορία που είναι απαραίτητη για να εμφανίσει την εκάστοτε στιγμή.
- **Καλός Σχεδιασμός**: Ο σχεδιασμός μιας εφαρμογής αποτελεί ένα από τα σημαντικότερα δομικά της στοιχεία. Θα πρέπει οι χρήστες που χρησιμοποιούν για πρώτη φορά στην εφαρμογή να εντυπωσιάζονται, να τους δίνεται εύκολα και άμεσα η πληροφορία που τους ενδιαφέρει ώστε να επισκεφτούν ξανά στην εφαρμογή. Ο συνδυασμός της αρχιτεκτονικής, της στοίχισης και των χρωματισμών θα πρέπει να συνδυάζονται με τέτοιο τρόπο ώστε να μην κουράζουνι ή μπερδεύουν τον χρήστη. Για το λόγο αυτό θα πρέπει να επιμεληθεί προσεκτικά η σχεδίαση του User Interface (UI) και User Experience (UX), δηλαδή το που θα τοποθετηθούν τα κουμπιά, οι φόρμες και πόσο εύκολη είναι η διαδρομή ενός χρήστη στην εφαρμογή για να λάβει το επιθυμητό αποτέλεσμα. Η σχεδίαση φυσικά δεν περιορίζεται μόνο στο εμφανισιακό κομμάτι αλλά στηρίζεται πρωταρχικά στην αρχιτεκτονική σχεδίαση της διαδικτυακής εφαρμογής ως θεμέλια για την περαιτέρω ανάπτυξη, συντήρηση και επέκταση της εφαρμογής.
- **Εγκυρότητα και Ενημέρωση**: Κάθε εφαρμογή οφείλει να διατηρεί το μέγιστο βαθμό εγκυρότητας και αξιοπιστίας απέναντι στους πελάτες της. Αυτός είναι άλλωστε και ο μοναδικός τρόπος για να μπορέσει να διατηρεί σε πρώτη φάση και να επεκτείνει σε δεύτερη το πελατολόγιο της. Οι πληροφορίες που θα παρέχει η εφαρμογή θα πρέπει να αντικατροπτίζονται πλήρως στην πραγματικότητα και στα δεδομένα που στηρίζονται. Επίσης, για ενδεχόμενα προβλήματα και αντιμετώπιση τους, θα πρέπει

η εφαρμογή να διαθέτει σύστημα βοήθειας και υποστήριξης των πελατών της, το λεγόμενο "*Support*", με αυτόν τον τρόπο θα βρίσκεται πάντα κοντά στον πελάτη δίνοντας του την μέγιστη δυνατή εμπειρία μέσω της εφαρμογής.

Ανάπτυξη του Ηλεκτρονικό Σύστημα κρατήσεων σαν διαδικτυακή εφαρμογή

Καθώς είναι απαραίτητη η πρόσβαση πολλών χρηστών στο ηλεκτρονικό σύστημα κρατήσεων, η ανάπτυξή του σαν διαδικτυακή εφαρμογή είναι μονόδρομος. Με αυτόν τον τρόπο οι πάροχοι θα μπορουν να διαχειρίζονται τις υπηρεσίες και τις κρατήσεις τους και οι καταναλωτές θα μπορούν να ενημερώνονται για αυτές και να δημιουργούν κρατήσεις ανα πάσα στιγμή. Επίσης οι πάροχοι θα εκμεταλλεύονται και τα υπόλοιπα πλεονεκτήματα μιας διαδικτυακής εφαρμογής όπως το οτι δεν θα χρειάζονται προσωπικό και επιπλέον πόρους για την συντήρηση του και θα έχουν άμεση πρόσβαση από κάθε κοινή συσκευή που προσφέρει πρόσβαση στο διαδίκτυο.

1.2.3 Παραδείγματα

Μία από τις βασικότερες διαδικτυακές εφαρμογές, που χρησιμοποιούνται ευρύτατα και καθημερινά στον τομέα της εκπαίδευσης, είναι η διαδικτυακή εφαρμογή-πλατφόρμα εκπαίδευσης Open eClass. Στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Υπολογιστών χρησιμοποιούμε κατά κόρον αυτήν την εφαρμογή για την εκπαιδευτικές διαδικασίες όπως διαμοιρασμός πληροφορίας, κοινοποίηση ανακοινώσεων, αποστολή εργασιών και ερωτήσεων. Πιο συγκεκριμένα, το Open eClass είναι ελεύθερο λογισμικό διαχείρισης εκπαιδευτικού περιεχομένου (Course Management System). Δημιουργήθηκε από την Ομάδα Ασύγχρονης Τηλεκπαίδευσης του Ελληνικού Ακαδημαϊκού Διαδικτύου GUnet, έχει εγκατασταθεί και χρησιμοποιείται από πολλά ελληνικά ακαδημαϊκά ιδρύματα. Όσον αφορά το κομμάτι των κρατήσεων, που αποτελεί και στοιχείο της διπλωματικής εργασίας, ένα παράδειγμα διαδικτυακής εφαρμογής είναι η AirBnB. Είναι μία από τις μεγαλύτερες εταιρείες στον τομέα των κρατήσεων που ειδικεύεται σε καταλύματα και ολόκληρα σπίτια, δηλαδή στην παροχή υπηρεσιών για διαμονή πελατών σε βραχυπρόθεσμη μίσθωση και η εταιρεία έχει ώς ρόλο τον μεσίτη ανάμεσα στους ιδιοκτήτες και τους πελάτες.

Εικόνα 1.2 Λογότυπο της διαδικτυακής εφαρμογής Open E-class
(https://www.openeclass.org/?fbclid=IwAR0NWQXZ0AUwRN5rICcmhpvwFGmmsV6-Z6pWI7zvOMO6v872wFBW0lyLd6s)

1.3 Λογισμικό ως Υπηρεσία (SaaS)

Η έννοια "*ως Υπηρεσία*" αντιπροσωπεύει κατά κύριο λόγο υπηρεσίες υπολογιστικού νέφους που προσφέρονται από έναν τρίτο πάροχο και εστιάζουν στις προτεραιότητες και ανάγκες του πελάτη. Κάθε τύπος υπολογιστικού νέφους αναλαμβάνει να διαχειρίζεται όλο και περισσότερες αρμοδιότητες ώς προς την δομή του συστήματος και της εφαρμογής, εξοικονομώντας έτσι πολύτιμο χρόνο στους πελάτες του SaaS λογισμικού. Το υπολογιστικό νέφος είναι ένα πολύ σημαντικό θέμα για μικρές και μεγάλες επιχειρήσεις ή οργανισμούς, ώστε να μπορούν ευκολότερα και δυναμικότερα να παρέχουν τις υπηρεσίες τους σε όσο το δυνατόν μεγαλύτερο εύρος του διαδικτύου. Το κομμάτι του υπολογιστικού νέφους μεγαλώνει μέρα με τη μέρα όλο και περισσότερο λόγω των συνεχόμενων προκλήσεων της τεχνολογίας με τα βασικότερα μοντέλα υπολογιστικού νέφους να είναι:

- Υποδομή ως Υπηρεσία (Infrastructure as a Service - **IaaS**)
- Πλατφόρμα ως Υπηρεσία (Platform as a Service - **PaaS**)
- Λογισμικό ως Υπηρεσία (Software as a Service - **SaaS**)

Υποδομή ως Υπηρεσία (IaaS): Οι υπηρεσίες υποδομής υπολογιστικού νέφους αποτελούν εξαιρετικά επεκτάσιμα εργαλεία για αυτοματοποιημένους υπολογιστικούς πόρους. Το σύστημα υποδομής ως υπηρεσία είναι πλήρως αυτοεξυπηρετούμενο για πρόσβαση και παρακολούθηση υπολογιστών, δικτύωσης και αποθήκευσης άλλων υπηρεσιών.

Τα κυριότερα χαρακτηριστικά της Υποδομής ως Υπηρεσία (IaaS):

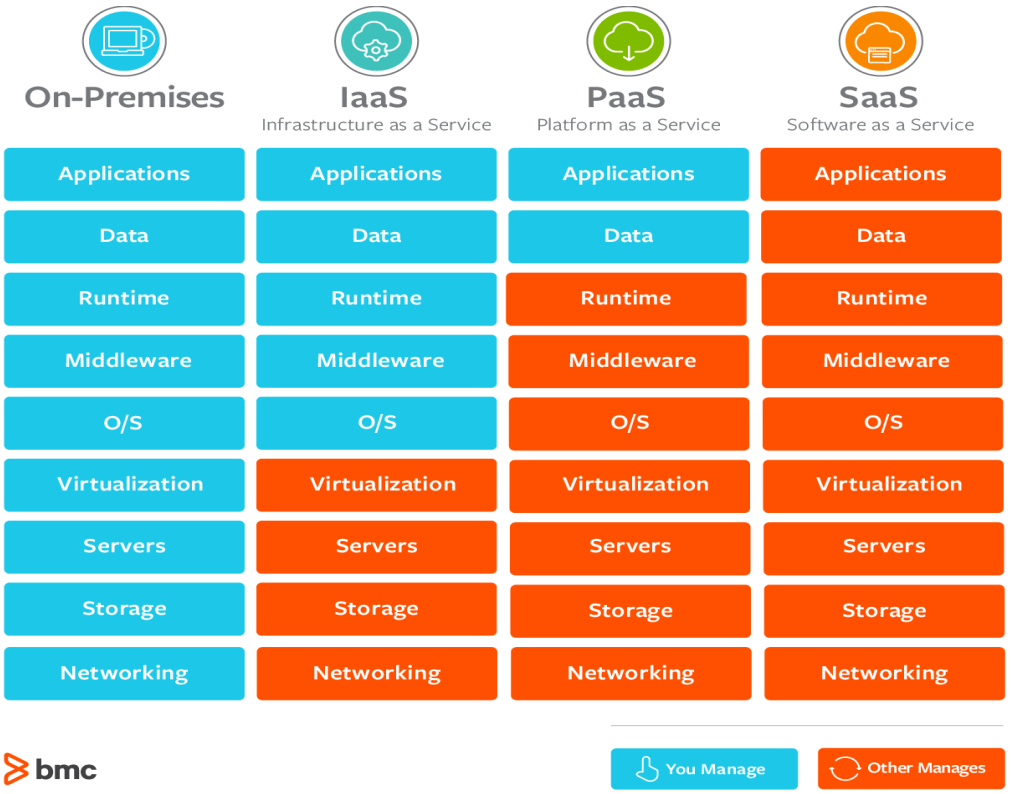
- Οι πόροι είναι διαθέσιμοι ως υπηρεσία.

- Το κόστος ποικίλλει ανάλογα με την κατανάλωση.
- Οι υπηρεσίες προσφέρουν δυνατότητα επεκτασιμότητας.
- Πολλαπλοί χρήστες σε ένα μόνο κομμάτι υλικού.
- Ο οργανισμός διατηρεί τον πλήρη έλεγχο της υποδομής.

Πλατφόρμα ως Υπηρεσία (PaaS): Οι υπηρεσίες πλατφόρμας, παρέχουν στοιχεία νέφους σε συγκεκριμένο λογισμικό ενώ χρησιμοποιούνται κυρίως για εφαρμογές. Το PaaS παρέχει ένα πλαίσιο για προγραμματιστές στο οποίο μπορούν να βασιστούν και να χρησιμοποιήσουν, για να δημιουργούν προσαρμοσμένες εφαρμογές. Όλοι οι διακομιστές, η αποθήκευση και η δικτύωση μπορούν να διαχειρίζονται από την επιχείρηση ή από τρίτο πάροχο, ενώ οι προγραμματιστές μπορούν να διατηρήσουν τη διαχείριση των εφαρμογών. Το μοντέλο PaaS παρέχει μια πλατφόρμα για τη δημιουργία λογισμικού, αυτή η πλατφόρμα παρέχεται μέσω του ιστού, δίνοντας στους προγραμματιστές την ελευθερία να επικεντρωθούν στην κατασκευή του λογισμικού χωρίς να χρειάζεται να ανησυχούν για λειτουργικά συστήματα, ενημερώσεις λογισμικού, αποθήκευση ή υποδομή.

Τα κυριότερα χαρακτηριστικά της Πλατφόρμας ως Υπηρεσία (PaaS) είναι:

- Βασίζεται στην τεχνολογία εικονικοποίησης, ώστε οι πόροι να μπορούν εύκολα να κλιμακώνονται κάθετα ή οριζόντια καθώς αλλάζουν οι ανάγκες.
- Παρέχει μια ποικιλία υπηρεσιών για να βοηθήσει στην σχεδίαση, τη δοκιμή και την ανάπτυξη εφαρμογών.
- Είναι προσβάσιμο σε πολλούς χρήστες μέσω της ίδιας εφαρμογής ανάπτυξης.
- Ενσωματώνει διαδικτυακές υπηρεσίες και βάσεις δεδομένων.



Εικόνα 1.3 Διαχείριση χαρακτηριστικών στα συστήματα ως Υπηρεσία (https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas)

1.3.1 Ορισμός Λογισμικού ως Υπηρεσία (SaaS)

Το **Λογισμικό ως Υπηρεσία (SaaS)** είναι ένα μοντέλο αδειοδότησης λογισμικού στο οποίο η πρόσβαση στο λογισμικό παρέχεται σε συνδρομητική βάση, με το λογισμικό να βρίσκεται σε εξωτερικούς διακομιστές και όχι σε διακομιστές που βρίσκονται εντός της εταιρείας. Η πρόσβαση στο λογισμικό γίνεται συνήθως μέσω ενός περιηγητή, με τους χρήστες να συνδέονται στο σύστημα χρησιμοποιώντας ένα όνομα χρήστη και έναν κωδικό πρόσβασης. Αντί κάθε χρήστης να χρειάζεται να εγκαταστήσει το λογισμικό στον υπολογιστή του, ο χρήστης μπορεί να έχει πρόσβαση στο πρόγραμμα μέσω του διαδικτύου. Η άνοδος του SaaS συμπίπτει με την άνοδο του υπολογιστικού νέφους. Το υπολογιστικό νέφος είναι η διαδικασία προσφοράς τεχνολογικών υπηρεσιών μέσω του διαδικτύου, η οποία συχνά περιλαμβάνει αποθήκευση δεδομένων, δικτύωση και διακομιστές. Προτού διατεθεί το SaaS, οι εταιρείες που ήθελαν να ενημερώνουν το λογισμικό στους υπολογιστές τους έπρεπε να αγοράσουν συμπαγείς δίσκους που περιείχαν τις ενημερώσεις και να τις κατεβάσουν στα συστήματα τους. Για μεγάλους οργανισμούς, η ενημέρωση του λογισμικού ήταν μια χρονοβόρα διαδικασία, με τον καιρό όμως οι ενημερώσεις λογισμικού έγιναν διαθέσιμες για λήψη μέσω του διαδικτύου, με τις εταιρείες να πληρώνουν πρόσθετες άδειες αντί για

πρόσθετους δίσκους. Ωστόσο, ένα αντίγραφο του λογισμικού έπρεπε να εγκατασταθεί σε όλες τις συσκευές που χρειαζόντουσαν πρόσβαση σε αυτό. Με το SaaS, οι χρήστες δεν χρειάζεται να εγκαταστήσουν ή να ενημερώσουν οποιοδήποτε λογισμικό, αντιθέτως οι χρήστες μπορούν να συνδεθούν μέσω ενός περιηγητή. Οι εταιρείες τεχνολογίας, οι εταιρείες χρηματοοικονομικών υπηρεσιών και οι επιχειρήσεις κοινής ωφέλειας έχουν οδηγήσει τον επιχειρηματικό κόσμο στην υιοθέτηση της τεχνολογίας SaaS.

Ιστορική αναδρομή στο Λογισμικό ως Υπηρεσία (SaaS)

Το SaaS μπορεί να ανιχνεύσει τις ρίζες του σε μια ιδέα που ονομάζεται “time-sharing”, η οποία αναπτύχθηκε στα τέλη της δεκαετίας του 1950 και στις αρχές της δεκαετίας του 1960 για να κάνει πιο οικονομικά αποδοτική χρήση του ακριβού χρόνου του επεξεργαστή. Τις επόμενες δεκαετίες, το υλικό και οι υπολογιστές έγιναν λιγότερο δαπανηρή. Οι οργανισμοί έκαναν στροφή στην ατομική ιδιοκτησία προσωπικών υπολογιστών χρησιμοποιώντας λογισμικό εσωτερικής εγκατάστασης. Αλλά δυστυχώς, το σύστημα εξακολουθούσε να αποδεικνύεται αναποτελεσματικό σε μεγαλύτερη κλίμακα, καθώς οι εταιρείες επιβαρύνονται από τη συνεχή συντήρηση λογισμικού και υλικού των μεμονωμένων υπολογιστών. Στα μέσα της δεκαετίας του '90, το διαδίκτυο έφτασε σε νέα ύψη όσον αφορά τις συναλλαγές ηλεκτρονικού εμπορίου. Η ανάπτυξη του Διαδικτύου στη συνέχεια τροφοδότησε τη γέννηση του "*online cloud*", το οποίο επέτρεψε στους οργανισμούς να έχουν πρόσβαση στο λογισμικό από οπουδήποτε. Το 1999, η Salesforce μπήκε δυναμικά στο SaaS μέσω τη δικής της πλατφόρμας διαχείρισης πελατειακών σχέσεων (**CRM**). Χάρη στην πρωτοπορία της, η Salesforce έγινε σύντομα ο πρώτος σούπερ σταρ στον χώρο του SaaS. Παραμένει μια από τις μεγαλύτερες αμιγώς εταιρείες SaaS στις Η.Π.Α. Με τη Salesforce να έχει αποδείξει τη βιωσιμότητα του επιχειρηματικού μοντέλου SaaS, εταιρείες όλων των σχημάτων και μεγεθών από νεοφυείς επιχειρήσεις έως καθιερωμένους κολοσσούς του κλάδου, όπως η Microsoft, η Oracle και η SAP ήταν πρόθυμοι να κινηθούν προς αυτήν την κατεύθυνση. Σήμερα, το SaaS είναι πανταχού παρόν με εταιρείες όπως η Adobe, η Salesforce, η Shopify και η Intuit να πρωτοστατούν.

1.3.2 Χαρακτηριστικά και πλεονεκτήματα χρήσης

Το Λογισμικό ως Υπηρεσία (SaaS) παρέχει πολυάριθμα πλεονεκτήματα σε υπαλλήλους και εταιρείες μειώνοντας σημαντικά τον χρόνο και τα χρήματα που δαπανώνται σε κουραστικές εργασίες όπως η εγκατάσταση, η διαχείριση και η αναβάθμιση λογισμικού. Αυτό ελευθερώνει

αρκετό χρόνο για το τεχνικό προσωπικό για να ξοδέψει σε διαφορετικά θέματα και ζητήματα εντός του οργανισμού.

Τα κυριότερα χαρακτηριστικά του Λογισμικού ως Υπηρεσία (SaaS) είναι:

- Διαχείριση από κεντρική τοποθεσία.
- Φιλοξενείται σε απομακρυσμένο διακομιστή.
- Προσβάσιμο μέσω διαδικτύου.
- Οι χρήστες δεν είναι υπεύθυνοι για ενημερώσεις υλικού ή λογισμικού.

Τα βασικά πλεονεκτήματα του Λογισμικού ως Υπηρεσία (SaaS) είναι:

Το SaaS προσφέρει μια ποικιλία πλεονεκτημάτων σε σχέση με τα παραδοσιακά μοντέλα αδειοδότησης λογισμικού. Είναι εύκολο στην εφαρμογή, εύκολο στην ενημέρωση και τον εντοπισμό σφαλμάτων και μπορεί να είναι λιγότερο δαπανηρό, καθώς οι χρήστες πληρώνουν για το SaaS χωρίς να αγοράζουν πολλαπλές άδειες χρήσης λογισμικού για πολλούς υπολογιστές. Αν απαριθμήσουμε τα πλεονεκτήματα έχουμε τα εξής:

- Υπηρεσίες email.
- Λειτουργίες ελέγχου.
- Αυτοματοποίηση εγγραφής για προϊόντα και υπηρεσίες.
- Διαχείριση εγγράφων, συμπεριλαμβανομένης της κοινής χρήσης αρχείων και της συνεργασίας εγγράφων.
- Κοινόχρηστα εταιρικά ημερολόγια, τα οποία μπορούν να χρησιμοποιηθούν για τον προγραμματισμό εκδηλώσεων.
- Συστήματα διαχείρισης πελατειακών σχέσεων (CRM).

Πότε ενδείκνυται να χρησιμοποιηθεί το Λογισμικό ως Υπηρεσία (SaaS):

- Οι νεοφυείς και μικρές εταιρείες που πρέπει να ξεκινήσουν γρήγορα την διάθεση των υπηρεσιών τους και δεν έχουν τον απαιτούμενο χρόνο, προσωπικό και οικονομική δυνατότητα να αναπτύξουν δικά τους συστήματα.
- Βραχυπρόθεσμα έργα που απαιτούν γρήγορη, εύκολη και οικονομικά προσιτή συνεργασία.

Κίνδυνοι που ελλοχεύουν από την χρήση Λογισμικού ως Υπηρεσία (SaaS):

- **Διαλειτουργικότητα**: Η ενσωμάτωση με υπάρχουσες εφαρμογές και υπηρεσίες μπορεί να δημιουργήσει ενδεχομένως κάποια ζητήματα εάν η εφαρμογή SaaS δεν έχει σχεδιαστεί για να ακολουθεί ανοιχτά πρότυπα ενοποίησης. Σε αυτήν την περίπτωση, οι οργανισμοί μπορεί να χρειαστεί να σχεδιάσουν τα δικά τους συστήματα ή να μειώσουν τις εξαρτήσεις με τις υπηρεσίες SaaS, κάτι που μπορεί να μην είναι πάντα εφικτό.
- **Εξάρτηση από τον πάροχο**: Οι πάροχοι μπορεί να διευκολύνουν τη συμμετοχή σε μια υπηρεσία και να δυσκολεύουν την απεμπλοκή από αυτήν. Δεν ακολουθεί κάθε προμηθευτής τυπικά APIs, πρωτόκολλα και εργαλεία, ωστόσο δίνει τις μέγιστες δυνατότητες σε κάθε πελάτη για την διαχείριση των δεδομένων του. Επίσης, μια πιθανή κατάρρευση του προμηθευτή ή ανικανότητας υποστήριξης νέων χαρακτηριστικών οδηγεί την επιχείρηση σε επώδυνες διαδικασίες για την απεξάρτηση ή ενσωμάτωση νέων στοιχείων στον υπάρχοντα πάροχο.
- **Έλλειψη υποστήριξης**: Πολλοί οργανισμοί απαιτούν ενσωματώσεις με εφαρμογές, δεδομένα και υπηρεσίες εσωτερικής εγκατάστασης. Ο προμηθευτής SaaS μπορεί να προσφέρει περιορισμένη υποστήριξη από αυτή την άποψη, αναγκάζοντας τους οργανισμούς να επενδύσουν εσωτερικούς πόρους στο σχεδιασμό και τη διαχείριση της ενσωμάτωσης. Η πολυπλοκότητα των ενσωματώσεων μπορεί να περιορίσει περαιτέρω τον τρόπο χρήσης της εφαρμογής SaaS ή άλλων εξαρτώμενων υπηρεσιών.
- **Ασφάλεια δεδομένων**: Μπορεί να χρειαστεί να ανταλλάσσονται μεγάλοι όγκοι δεδομένων στα κέντρα δεδομένων υποστήριξης των εφαρμογών SaaS προκειμένου να εκτελεστούν οι απαραίτητες λειτουργίες λογισμικού. Η μεταφορά ευαίσθητων πληροφοριών σε υπηρεσία SaaS που βασίζεται στο δημόσιο νέφος μπορεί να οδηγήσει σε κίνδυνο της ασφάλειας και της συμμόρφωσης, καθώς και ένα σημαντικό κόστος για τη μετεγκατάσταση μεγάλου φόρτου εργασίας δεδομένων.
- **Προσαρμογή**: Οι εφαρμογές SaaS προσφέρουν μικρές δυνατότητες προσαρμογής. Δεδομένου ότι δεν υπάρχει λύση που να ταιριάζει σε όλους, οι χρήστες ενδέχεται να περιορίζονται σε συγκεκριμένες λειτουργίες, επιδόσεις και ενσωματώσεις όπως προσφέρονται από τον προμηθευτή. Αντίθετα, οι λύσεις on-premise που συνοδεύονται από πολλά κιτ ανάπτυξης λογισμικού (**SDK**) προσφέρουν υψηλό βαθμό επιλογών προσαρμογής.

- **Έλλειψη ελέγχου**: Οι λύσεις SaaS περιλαμβάνουν την παράδοση του ελέγχου σε τρίτο πάροχο υπηρεσιών. Αυτά τα στοιχεία ελέγχου δεν περιορίζονται στο λογισμικό όσον αφορά την έκδοση, τις ενημερώσεις ή την εμφάνιση αλλά και τα δεδομένα. Ως εκ τούτου, οι πελάτες μπορεί να χρειαστεί να επαναπροσδιορίσουν τα μοντέλα ασφάλειας δεδομένων ώστε να ταιριάζουν στις δυνατότητες και τη λειτουργικότητα της υπηρεσίας SaaS.
- **Περιορισμοί χαρακτηριστικών**: Δεδομένου ότι οι εφαρμογές SaaS έρχονται συχνά σε τυποποιημένη μορφή, η επιλογή των χαρακτηριστικών μπορεί να είναι μια συμβιβαστική αντιστάθμιση έναντι της ασφάλειας, του κόστους, της απόδοσης ή άλλων πολιτικών του οργανισμού. Επιπλέον, ανησυχίες σχετικά με την εξάρτηση από τον προμηθευτή, το κόστος ή την ασφάλεια μπορεί να σημαίνουν ότι δεν είναι βιώσιμο να αλλαχθούν προμηθευτές ή υπηρεσίες για την εξυπηρέτηση νέων απαιτήσεων.
- **Απόδοση και διακοπή λειτουργίας**: Επειδή ο προμηθευτής ελέγχει και διαχειρίζεται την υπηρεσία SaaS, οι πελάτες εξαρτώνται πλέον από τους προμηθευτές για τη διατήρηση της ασφάλειας και της απόδοσης της υπηρεσίας. Η προγραμματισμένη και μη προγραμματισμένη συντήρηση, οι επιθέσεις στον κυβερνοχώρο ή τα ζητήματα δικτύου ενδέχεται να επηρεάσουν την απόδοση της εφαρμογής SaaS, παρά την ισχύουσα προστασία κατάλληλων συμφωνιών επιπέδου υπηρεσιών (**SLA**).

Επιλογή του SaaS για την ανάπτυξη της διαδικτυακής εφαρμογής κρατήσεων:

Κατόπιν έρευνας πάνω στα μοντέλα ώς υπηρεσία, αλλά και των αναγκών που παρουσιάζει ένα σύστημα κρατήσεων για να είναι διαθέσιμο σε όλους τους χρήστες, καταλήξαμε να ακολουθήσουμε το μοντέλο SaaS. Μέσω των χαρακτηριστικών που προσφέρει το μοντέλο SaaS, θα μπορέσουμε να εκμεταλλευτούμε την εύκολη πρόσβαση που θέλουμε να προσφέρουμε στους χρήστες του συστήματος κρατήσεων χωρίς να χρειάζεται να εγκαταστήσουν κάποια εφαρμογή στο κινητό τους παρά μόνο να έχουν πρόσβαση σε ένα περιηγητή και σύνδεση στο διαδίκτυο. Μέσω της διαχείρισης μεγάλων ενοτήτων της εφαρμογής από έναν πάροχο, μας δόθηκε η ευκαιρία να επικεντρωθούμε κυρίως στην ανάπτυξη της εφαρμογής και των υποσυστημάτων της. Μέσω αυτής της επιλογής, παρουσιάζουμε ένα υπαρκτό μοντέλο ανάπτυξης εφαρμογών που χρησιμοποιείται ευρέως στην αγορά, δηλαδή ένα μοντέλο "*Λογισμικού κατ' απαίτηση*" το οποίο προσφέρεται στους πελάτες αξιοποιώντας το υπολογιστικό νέφος. Επιδιώξαμε να αναδείξουμε το γεγονός ότι οι μικρές και νεοφυείς εταιρείες συνήθως δεν έχουν την τεχνογνωσία ούτε την οικονομική ικανότητα για να οργανώσουν την υποδομή πληροφορικής με ιδίους πόρους. Αυτός είναι ο κύριος λόγος που το SaaS είναι μια έξυπνη λύση για την ανάγκη μείωσης του κόστους και

αύξησης της αποτελεσματικότητας. Τέλος, μέσω της ανάπτυξης ενός ελάχιστου βιώσιμου προϊόντος πάνω στο μοντέλο SaaS, αναδεικνύεται και η δυνατότητα χρήσης της εφαρμογής ως συνδρομητική υπηρεσία προς τους χρήστες εξασφαλίζοντας έτσι και την βιωσιμότητα της εφαρμογής. [1] [2]

1.3.3 Παραδείγματα

Ένα παράδειγμα των πιο διάσημων SaaS υπηρεσιών είναι το Google Docs. Πιο αναλυτικά:

Google Docs: Ένα από τα πιο απλά παραδείγματα SaaS στον πραγματικό κόσμο είναι τα έγγραφα Google, ο δωρεάν διαδικτυακός επεξεργαστής κειμένου της Google. Για να χρησιμοποιήσουμε τα έγγραφα Google, το μόνο που χρειάζεται να κάνουμε είναι να συνδεθούμε σε ένα πρόγραμμα περιήγησης ιστού. Τα έγγραφα Google μας επιτρέπουν να γράφουμε, να επεξεργαζόμαστε, ακόμη και να συνεργαζόμαστε με άλλους όπου κι αν βρίσκονται. Το Google Docs κυκλοφόρησε τον Οκτώβριο του 2012.

Κεφάλαιο 2ο

Τεχνολογία λογισμικού

Ο σύγχρονος κόσμος προκειμένου να διευκολύνει τις δραστηριότητες του στηρίζεται σε σύνθετα συστήματα λογισμικού. Όλες οι υποδομές και οι υπηρεσίες εξαρτώνται από ένα υπολογιστικό σύστημα, ενώ τα περισότερα ηλεκτρονικά προϊόντα περιλαμβάνουν κάποιο λογισμικό ελέγχου. Επίσης τις τελευταίες δύο δεκαετίες η εκρηκτική ανάπτυξη της τεχνολογίας οδήγησε τις κοινωνίες μας στην απορρόφηση της τεχνολογίας που πλέον αποτελεί κομμάτι της καθημερινότητας μας. Από αυτό προκύπτει ότι η αποτελεσματική ανάπτυξη και συντήρηση του λογισμικού είναι αναγκαία για την λειτουργία των συστημάτων και την εύρυθμη χρήση τους από τους ανθρώπους. Τα συστήματα λογισμικού δεν έχουν φυσικούς περιορισμούς καθώς είναι αφηρημένα και άυλα. Το γεγονός αυτό απλουστεύει την διαδικασία ανάπτυξής τους αλλά επιτρέπει να γίνουν εξαιρετικά πολύπλοκα και δυσνόητα καθώς και κοστοβόρα σε τροποποιήσεις. Με την άνοδο της ισχύς του υπολογιστικού υλικού, το παραγόμενο λογισμικό ήταν κατά πολλές τάξης μεγέθους μεγαλύτερο και πιο περίπλοκο από τα προηγούμενα συστήματα λογισμικού. Ήταν σαφές ότι η ανάπτυξη λογισμικού χωρίς τυποποιημένες διαδικασίες ήταν ανεπαρκής, καθώς το λογισμικό που προέκυπτε ήταν αναξιόπιστο, δύσκολο να συντηρηθεί και ξεπερνούσε το προβλεπόμενο κόστος και χρόνο παράδοσης του. Η τεχνολογία λογισμικού (Software engineering), λοιπόν, είναι ο τεχνικός κλάδος που αποσκοπεί στην αποδοτικότερη παραγωγή (ανάπτυξη) συστημάτων λογισμικού.

Οι τεχνικές τεχνολογίας λογισμικού δεν εφαρμόζονται πάντα σωστά και πολλά έργα αποτυγχάνουν να παράξουν λογισμικό υψηλής ποιότητας. Καθώς οι απαιτήσεις για λογισμικό εξελίσσονται και αυξάνονται, οι υπάρχουσες τεχνικές δεν είναι πάντα επαρκείς οπότε δημιουργούνται νέες για να καλύψουν τις καινούριες ανάγκες. Η ανάπτυξη ενός συστήματος επικοινωνίας, ενός λογισμικου ελέγχου κάποιου επιστημονικού οργάνου ή μιας διασύνδεσης χρήστη μέσω γραφικού περιβάλλοντος χρήζει διαφορετικής τεχνικής προσέγγισης. Πλέον διαθέτουμε αποτελεσματικές μεθόδους για την εξαγωγή προδιαγραφών, σχεδιασμού και υλοποίησης λογισμικού, καθώς και εργαλεία που διευκολύνουν την ανάπτυξη πολύπλοκων συστημάτων λογισμικού. Η ανάπτυξη της τεχνολογίας λογισμικού έχει βελτιώσει το λογισμικό που διαθέτουμε σήμερα και χωρίς αυτή δεν θα μπορούσαμε να αναπτύξουμε πιο σύνθετα

λογισμικά συστήματα που είναι απαραίτητα για την σημερινή κοινωνία. Η τεχνολογία λογισμικού προορίζεται για την υποστήριξη επαγγελματικής ανάπτυξης λογισμικού, παρά τον μεμονωμένο προγραμματισμό. Περιλαμβάνει τεχνικές που υποστηρίζουν τις προδιαγραφές προγράμματος, την σχεδίαση και την εξέλιξη, κανένα από τα οποία δεν σχετίζεται συνήθως με την ανάπτυξη προσωπικού λογισμικού. Παρακάτω παραθέτουμε ορισμένες απόψεις του **Ian Sommerville** που διατυπώθηκαν στο βιβλίο του Software Engineering Ninth Edition.

Σύμφωνα με τον I. Sommerville σαν λογισμικό εννοούμε ένα πρόγραμμα υπολογιστή και την σχετική τεκμηρίωση του. Τα προϊόντα λογισμικού μπορούν να αναπτύσσονται για έναν συγκεκριμένο πελάτη κατα παραγγελία ή να είναι γενικής χρήσης για όποιον διατίθεται να το αγοράσει. Εν συνεχεία η τεχνολογία που εφαρμόζεται στον λογισμικό σαν όρος ονομάζεται “Τεχνολογία λογισμικού” και είναι το πεδίο που ασχολείται με όλες τις πτυχές της παραγωγής λογισμικού, από τα πρώτα στάδια της εξαγωγής προδιαγραφών ενός συστήματος λογισμικού μέχρι την συντήρηση του συστήματος μετά την διάθεση του για χρήση. Κατά την ανάπτυξη του λογισμικού φυσικά θα πρέπει να αντιμετωπιστούν σημαντικές προκλήσεις όπως η αντιμετώπιση της αυξανόμενης ετερογένειας, των απαιτήσεων για μείωση του χρόνου παράδοσης, και της ανάπτυξης έμπιστου λογισμικού καθώς και να διαθέτει γνωρίσματα όπως η απόδοση, η συντηρησιμότητα, η φερεγγυότητα και η ευχρηστία.

Η διαδικασία παραγωγής λογισμικού είναι ένα σύνολο δραστηριοτήτων με σκοπό την ανάπτυξη ή την εξέλιξη λογισμικού. Οι τέσσερις θεμελιώδεις δραστηριότητες, κοινές σε όλες τις διαδικασίες παραγωγής λογισμικού, που αναπτύσσονται στα επόμενα υποκεφάλαια, είναι οι εξής: Η εξαγωγή προδιαγραφών, η ανάπτυξη του λογισμικού, η επικύρωση του λογισμικού και η εξέλιξη του λογισμικού.

2.1 Διαδικασίες παραγωγής λογισμικού

Η τεχνολογία λογισμικού αφορά την απόκτηση αποτελεσμάτων της απαιτούμενης ποιότητας εντός του χρονοδιαγράμματος και του προϋπολογισμού που συχνά οδηγεί σε συμβιβασμούς. Γενικά, οι μηχανικοί λογισμικού υιοθετούν μια συστηματική και οργανωμένη προσέγγιση στην εργασία τους, καθώς αυτός είναι συχνά ο πιο αποτελεσματικός τρόπος παραγωγής λογισμικού υψηλής ποιότητας. Η συστηματική προσέγγιση που χρησιμοποιείται στη τεχνολογία λογισμικού ονομάζεται διαδικασία παραγωγής λογισμικού. Διαδικασία παραγωγής λογισμικού είναι ένα σύνολο δραστηριοτήτων με σκοπό την ανάπτυξη και την εξέλιξη

λογισμικού. Υπάρχουν τέσσερις θεμελιώδεις δραστηριότητες, κοινές σε όλες τις διαδικασίες παραγωγής λογισμικού. Αυτές οι δραστηριότητες, οι οποίες αναπτύσσονται αναλυτικότερα στα επόμενα υποκεφάλαια, είναι:

1. Η εξαγωγή προδιαγραφών (Software specification), όπου οι πελάτες και οι μηχανικοί ορίζουν το λογισμικό που πρόκειται να παραχθεί και τους περιορισμούς στη λειτουργία του.
2. Η ανάπτυξη του λογισμικού (Software development), όπου σχεδιάζεται και προγραμματίζεται το λογισμικό.
3. Η επικύρωση του λογισμικού (Software validation), όπου το λογισμικό ελέγχεται για να διασφαλιστεί ότι είναι αυτό που το απαιτεί ο πελάτης.
4. Η εξέλιξη του λογισμικού (Software evolution), όπου το λογισμικό τροποποιείται ώστε να αντικατοπτρίζει τις μεταβαλλόμενες απαιτήσεις των πελατών και της αγοράς.

Αυτές οι γενικές δραστηριότητες μπορεί να οργανωθούν με διαφορετικούς τρόπους και να περιγράφονται σε διαφορετικά επίπεδα λεπτομέρειας ανάλογα με τον τύπο του λογισμικού που αναπτύσσεται.

2.1.1 Μοντέλα διαδικασίας παραγωγής

Μοντέλο διαδικασίας παραγωγής λογισμικού ονομάζεται μια απλοποιημένη περιγραφή κάποιας διαδικασίας παραγωγής λογισμικού, η οποία αντιπροσωπεύει μια διαδικασία από μια συγκεκριμένη προοπτική και έτσι παρέχει μόνο μερικές πληροφορίες σχετικά με αυτή τη διαδικασία. Για παράδειγμα, ένα μοντέλο διαδικασίας δείχνει τις δραστηριότητες και τη σειρά τους, αλλά μπορεί να μην παρουσιάζει τους ρόλους των ατόμων που εμπλέκονται σε αυτές τις δραστηριότητες.

Αυτά τα γενικά μοντέλα δεν είναι οριστικές περιγραφές διαδικασιών λογισμικού, αλλά περισσότερο, είναι προσεγγίσεις της διαδικασίας που μπορούν να χρησιμοποιηθούν για να εξηγήσουν διαφορετικές προσεγγίσεις στην ανάπτυξη λογισμικού. Μπορούμε να τα σκεφτούμε ως πλαίσια διεργασιών που μπορούν να επεκταθούν και να προσαρμοστούν για να δημιουργήσουν πιο συγκεκριμένες διαδικασίες μηχανικής λογισμικού. Παρακάτω θα παρουσιάσουμε τα τρία γενικά μοντέλα ανάπτυξης λογισμικού που βασίζονται τα περισσότερα μοντέλα διαδικασιών παραγωγής λογισμικού.

1. **Προσέγγιση καταρράκτη** (Waterfall). Αυτή η προσέγγιση λαμβάνει τις θεμελιώδεις δραστηριότητες διαδικασίας της προδιαγραφής, της ανάπτυξης, της επικύρωσης και της εξέλιξης και τις αναπαριστά ως ξεχωριστές φάσεις διαδικασίας, όπως η προδιαγραφή απαιτήσεων, ο σχεδιασμός λογισμικού, η υλοποίηση, η δοκιμή κλπ. Μόλις οριστεί ένα στάδιο επισημαίνεται ως κλειστό, και η ανάπτυξη προχωράει στο επόμενο στάδιο.
2. **Εξελικτική ανάπτυξη** (Incremental). Αυτή η προσέγγιση εμπλέκει τις δραστηριότητες της εξαγωγής προδιαγραφών, της ανάπτυξης και της επικύρωσης. Το σύστημα αναπτύσσεται ως μια σειρά εκδόσεων (ενημερώσεις), με κάθε έκδοση να προσθέτει λειτουργικότητα στην προηγούμενη έκδοση μέχρι να καταλήξει σε ένα σύστημα που ικανοποιεί τις ανάγκες του πελάτη.
3. **Τεχνολογία λογισμικού** βάση συστατικών στοιχείων (Component-based software engineering, CBSE). Αυτή η προσέγγιση βασίζεται στην ύπαρξη ενός σημαντικού αριθμού επαναχρησιμοποιήσιμων εξαρτημάτων. Η διαδικασία ανάπτυξης συστήματος επικεντρώνεται στην ενσωμάτωση αυτών των στοιχείων σε ένα σύστημα αντί στην ανάπτυξη τους από την αρχή.

Αυτά τα μοντέλα δεν αλληλοαποκλείονται και χρησιμοποιούνται συχνά μαζί, ειδικά για την ανάπτυξη μεγάλων συστημάτων. Για μεγάλα συστήματα ή συστήματα που αποτελούνται από επιμέρους υποσυστήματα, είναι λογικό να συνδυάζονται μερικά από τα καλύτερα χαρακτηριστικά του καταρράκτη και τα μοντέλα σταδιακής ανάπτυξης. Πρέπει να υπάρχουν πληροφορίες σχετικά με τις βασικές απαιτήσεις συστήματος για να σχεδιαστεί μια αρχιτεκτονική λογισμικού που να υποστηρίζει αυτές τις απαιτήσεις, κάτι που δεν μπορεί να αναπτυχθεί σταδιακά. Τα υποσυστήματα σε ένα μεγαλύτερο σύστημα μπορούν να αναπτυχθούν χρησιμοποιώντας διαφορετικές προσεγγίσεις. Μέρη του συστήματος που είναι καλά κατανοητά μπορούν να καθοριστούν και να αναπτυχθούν χρησιμοποιώντας μια διαδικασία που βασίζεται σε καταρράκτη ενώ μέρη του συστήματος που είναι δύσκολο να προσδιοριστούν εκ των προτέρων, όπως η διεπαφή χρήστη, θα πρέπει πάντα να αναπτύσσονται χρησιμοποιώντας μια σταδιακή προσέγγιση.

Παραγωγή του λογισμικού με εξελικτική ανάπτυξης

Η εξελικτική ανάπτυξη σε κάποια μορφή είναι πλέον η πιο κοινή προσέγγιση για την ανάπτυξη συστημάτων εφαρμογών. Αυτή η προσέγγιση μπορεί να είναι είτε με γνώμονα το σχέδιο (plan-driven), είτε ευέλικτη (agile) ή, συνηθέστερα, ένα μείγμα αυτών των προσεγγίσεων. Σε μια προσέγγιση βάσει σχεδίου, οι ενημερώσεις (updates) του συστήματος

προσδιορίζονται εκ των προτέρων. Υιοθετώντας μια ευέλικτη προσέγγιση, εντοπίζουμε τις πρώτες ενημερώσεις με σκοπό να αναπτυχθεί ταχύτερα ένα έτοιμο προς χρήση προϊόν, και η ανάπτυξη των μεταγενέστερων ενημερώσεων θα εξαρτηθεί από την πρόοδο και τις προτεραιότητες των πελατών.

Κεφάλαιο 3ο

Τεχνολογίες που χρησιμοποιήθηκαν

3.1 Διαδίκτυο

Το Διαδίκτυο ή Internet είναι το μεγαλύτερο παγκόσμιο επικοινωνιακό δίκτυο που επιτρέπει την διασύνδεση των ηλεκτρονικών υπολογιστών για να επικοινωνούν και να ανταλλάσουν δεδομένα χρησιμοποιώντας το πρωτόκολλο επικοινωνίας TCP/IP (Transmission Control Protocol/Internet Protocol). Η τεχνολογία του διαδικτύου βασίζεται κυρίως στην διασύνδεση επιμέρους δικτύων ανά τον κόσμο και πολυάριθμα πρωτοκολλα επικοινωνίας. Το Διαδίκτυο (Internet) διασυνδέει τοπικά δίκτυα που βρίσκονται σε εκπαιδευτικά ιδρύματα, νοσοκομεία, πανεπιστήμια, βιβλιοθήκες, εταιρείες, διεθνής οργανισμούς, εκπαιδευτικά κέντρα, κλπ, σε ένα τεράστιο δίκτυο που διαρκώς επεκτείνεται. Κάθε υπολογιστής στο διαδίκτυο έχει τουλάχιστον μία διεύθυνση IP με την οποία μπορεί να ταυτοποιείται. Το Διαδίκτυο, σε συνδυασμό με την ολοένα αναπτυσσόμενη ψηφιακή τεχνολογία, έχει δημιουργήσει μία τεράστια αγορά γνώσεων και πληροφοριών. Τα πρώτα δειλά βήματα του Διαδικτύου ξεκίνησαν το 1969 όταν για πρώτη φορά εγκαταστάθηκε και λειτούργησε με 4 κόμβους το δίκτυο. Το 1984 υλοποιείται το πρώτο **DNS** (Domain Name System) σύστημα στο οποίο καταγράφονται 1000 κεντρικοί κόμβοι και οι υπολογιστές του διαδικτύου πλέον αναγνωρίζονται από διευθύνσεις κωδικοποιημένων αριθμών. Ο όρος Διαδίκτυο/Ίντερνετ ξεκίνησε να χρησιμοποιείται ευρέως την εποχή που συνδέθηκε το ARPANET με το NSFNet και Internet σημαίνει οποιοδήποτε δίκτυο χρησιμοποιούσε TCP/IP. Η μεγάλη άνθηση του Διαδικτύου όμως, ξεκίνησε με την εφαρμογή της υπηρεσίας του Παγκόσμιου Ιστού από τον Τιμ Μπέρνερς-Λι στο ερευνητικό ίδρυμα CERN το 1989, ο οποίος είναι στην ουσία, η "*πλατφόρμα*", η οποία κάνει εύκολη την πρόσβαση στο Ίντερνετ, ακόμα και στη μορφή που είναι γνωστό σήμερα.

Ο Παγκόσμιος Ιστός αποτελεί μια τεράστια συλλογή πληροφοριών, αποθηκευμένων σε διάφορες μορφές (κείμενο, εικόνα, ήχος, κινούμενη εικόνα, γραφικά κ.ά.). Αυτή η συλλογή είναι κατανεμημένη και διατίθεται στο Διαδίκτυο μέσω ειδικών εφαρμογών, που ονομάζονται εξυπηρετητές Παγκόσμιου Ιστού (WWW servers). Κάθε χρήστης του Διαδικτύου μπορεί να περιηγηθεί στον Παγκόσμιο Ιστό και να προσπελάσει τις διαθέσιμες πληροφορίες

χρησιμοποιώντας μια ειδική εφαρμογή πελάτη, που ονομάζεται πρόγραμμα περιήγησης (Browser).

Εικόνα 3.1 Απεικόνιση διασυνδεδεμένου διαδικτύου
(https://dimpapp.gr/wp-content/uploads/2015/10/internet.jpg)

Τα βασικότερα πλεονεκτήματα χρήσης του διαδικτύου είναι:

- Εύρεση επιστημονικών και ερευνητικών δεδομένων και γνώσεις.
- Δημοσιεύσεις άρθρων.
- Χρήση δημοσίων υπηρεσιών.
- Δυνατότητα εκπαίδευσης, κατάρτισης και επιμόρφωσης για όλους.
- Ευκαιρία προβολής και διαφήμισης για τις επιχειρήσεις.

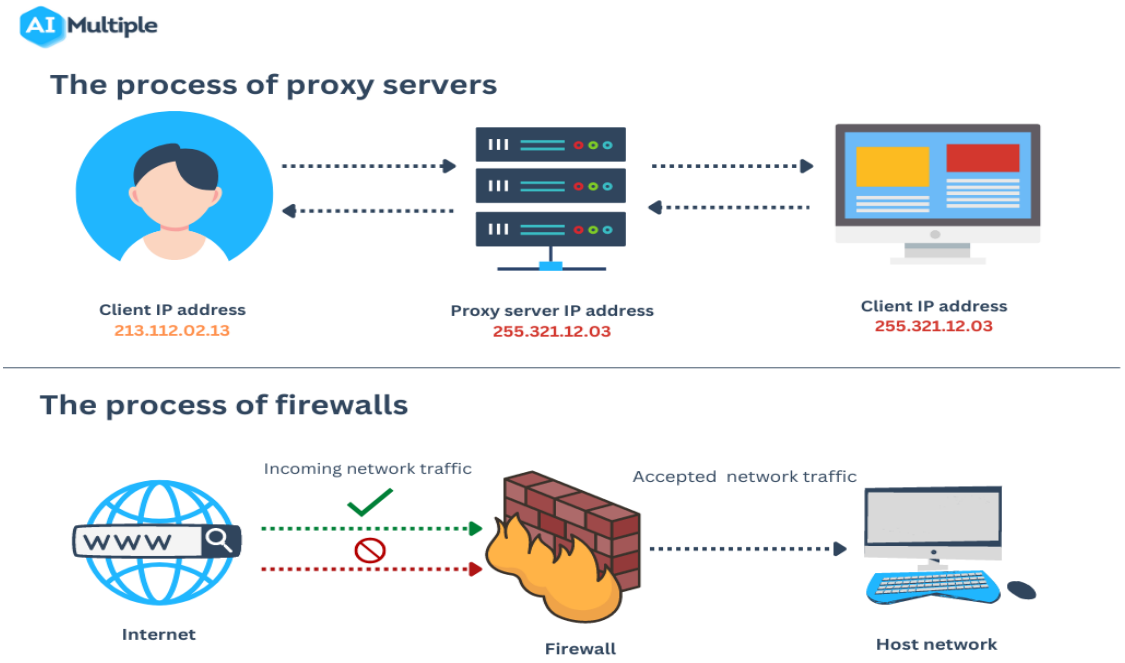
Τείχος προστασίας

Για να αποφύγουμε ευπαθείς στιγμές, συνήθως δημιουργούμε ένα σημείο πρόσβασης στο τοπικό δίκτυο ώστε όλα τα δεδομένα να εισέρχονται και να εξέρχονται μέσω αυτού του σημείου. Το hardware και το software που εγκαθίστανται σε αυτό το σημείο μεταξύ του Διαδικτύου και του τοπικού δικτύου ελέγχουν όλα τα εισερχόμενα ή εξερχόμενα δεδομένα για να επιτραπεί ή να απορριφθεί η διέλευσή τους βάσει κανόνων. Αυτό το σημείο πρόσβασης ονομάζεται τείχος προστασίας ή Firewall. Συνήθως αποτελεί μέρος του δρομολογητή που συνδέει το τοπικό δίκτυο με το Διαδίκτυο και μπορεί να εκτελεί και άλλες λειτουργίες, όπως τη μετάφραση διευθύνσεων δικτύου. Το τείχος προστασίας είναι υπεύθυνο για τον έλεγχο όλων των δεδομένων που διέρχονται από τη διασύνδεση δικτύου και την αποδοχή ή απόρριψή

τους βάσει κανόνων φιλτραρίσματος. Οι κανόνες αυτοί συνήθως βασίζονται σε διευθύνσεις και θύρες δικτύου.

Διακομιστές μεσολάβησης

Οι διακομιστές μεσολάβησης (Proxy Servers) σχετίζονται με το τείχος προστασίας. Αν υπάρχει τείχος προστασίας που αποτρέπει τους υπολογιστές ενός δικτύου από το να πραγματοποιούν απευθείας συνδέσεις με τον έξω κόσμο, τότε ο διακομιστής μεσολάβησης μπορεί να λειτουργήσει ως μεσάζοντας. Έτσι, ένα μηχάνημα το οποίο δεν επιτρέπεται να συνδεθεί στο εξωτερικό δίκτυο εξαιτίας κάποιου τοίχους προστασίας, δεν θα στείλει αίτηση για μια ιστοσελίδα απευθείας στον απομακρυσμένο διακομιστή Ιστού, αλλά θα απευθυνθεί στον τοπικό διακομιστή μεσολάβησης. Ο τελευταίος θα προωθήσει την αίτηση για τη σελίδα στον διακομιστή Ιστού και θα επιστρέψει την απάντηση στο μηχάνημα που έστειλε αρχικά την αίτηση. Διακομιστές μεσολάβησης μπορούν επίσης να χρησιμοποιούνται για τις υπηρεσίες FTP και άλλες συνδέσεις. Ένα από τα πλεονεκτήματα της χρήσης διακομιστή μεσολάβησης όσον αφορά την ασφάλεια είναι το γεγονός ότι οι εξωτερικοί υπολογιστές υπηρεσίας αντλούν πληροφορίες μόνο για τον διακομιστή μεσολάβησης. Εξακολουθούν να μην γνωρίζουν τα ονόματα και τις διευθύνσεις IP των εσωτερικών μηχανημάτων, κάτι που απομακρύνει το ενδεχόμενο να εισβάλουν χάκερ στα εσωτερικά συστήματα. [9]



Εικόνα 3.2 Διεργασία διακομιστή μεσολάβησης και τείχους προστασίας (https://research.aimultiple.com/proxy-server-types)

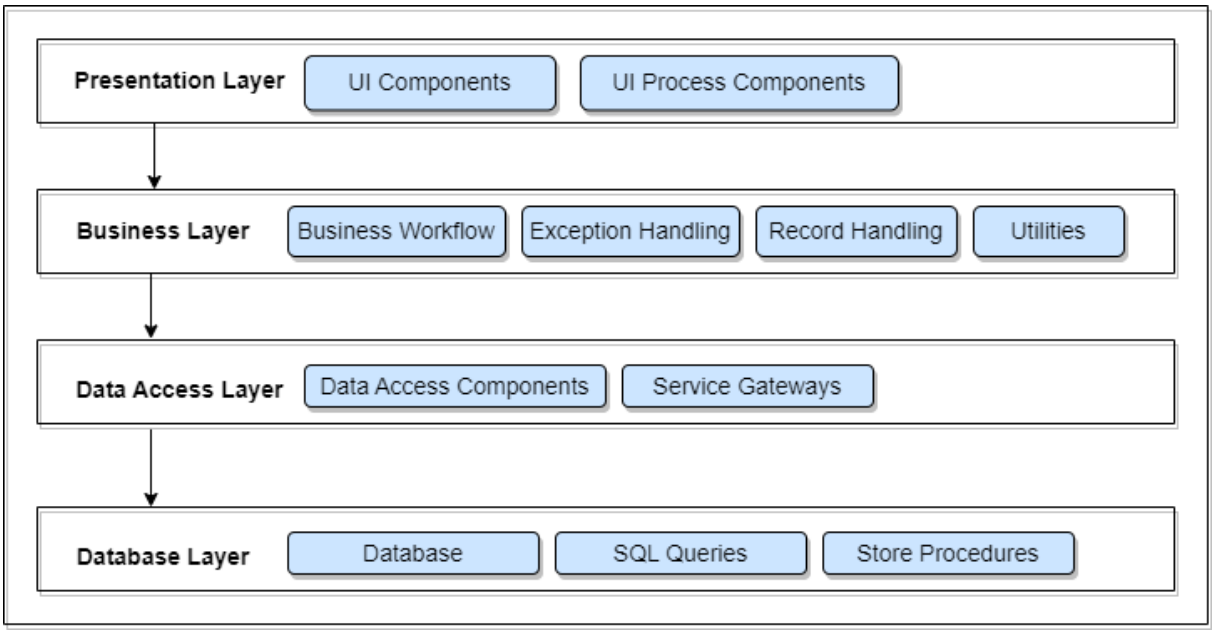
3.2 Αρχιτεκτονική Διαδικτυακών Εφαρμογών

Αν προσπαθήσουμε να δώσουμε έναν ορισμό για την αρχιτεκτονική των διαδικτυακών εφαρμογών θα ήταν: "*Η αρχιτεκτονική των διαδικτυακών εφαρμογών είναι ο «σκελετός» ή η διάταξη που εμφανίζει τις αλληλεπιδράσεις μεταξύ των στοιχείων της εφαρμογής, των συστημάτων ενδιάμεσου λογισμικού, των διεπαφών χρήστη και των βάσεων δεδομένων. Αυτό το είδος αλληλεπίδρασης επιτρέπει σε έναν αριθμό εφαρμογών να συνεργάζονται ταυτόχρονα.*" Πολλές φορές προκαλείται σύγχυση όσον αφορά την αρχιτεκτονική και την σχεδίαση μιας διαδικτυακής εφαρμογής. Για να αποσαφηνίσουμε τις δύο έννοιες, η αρχιτεκτονική λογισμικού απευθύνεται σε όλα τα στοιχεία υψηλού επιπέδου ενός συστήματος και την αλληλεπίδραση μέσα σε αυτά. Εν αντιθέση με τη σχεδίαση λογισμικού που εστιάζει σε επίπεδο κώδικα στη διανομή της λογικής σε διάφορες ενότητες με τους δικούς της συγκεκριμένους όρους. Αυτό βοηθά στη δημιουργία και τη διαχείριση της λογικής της εφαρμογής. Μια αρχιτεκτονική εφαρμογών καθορίζει επίσης τον τρόπο παράδοσης των δεδομένων μέσω HTTP και διασφαλίζει ότι ο διακομιστής πελάτη και ο διακομιστής υποστήριξης μπορούν να το κατανοήσουν. Επιπλέον, διασφαλίζει ότι υπάρχουν έγκυρα δεδομένα σε όλα τα αιτήματα των χρηστών, δημιουργεί και διαχειρίζεται τον σχεδιασμό που καθορίζει την ανάπτυξη του συστήματος, την αξιοπιστία, τη διαλειτουργικότητα και τις μελλοντικές ανάγκες. Ως εκ τούτου, είναι σημαντικό να κατανοήσουμε τα στοιχεία που αποτελούν την αρχιτεκτονική των διαδικτυακών εφαρμογών.

Τα δομικά στοιχεία που αποτελούν μια διαδικτυακή εφαρμογή είναι τρία:

1. **Περιηγητής Ιστού**: Ο περιηγητής ή καλύτερα το front-end κομμάτι είναι ο τρόπος επικοινωνίας και αλληλεπίδρασης με τον χρήστη, λαμβάνει επίσης δεδομένα, τα διαχειρίζεται και τα παρουσιάζει στο επίπεδο της εφαρμογής που έχει πρόσβαση ο χρήστης.
2. **Διαδικτυακός Διακομιστής**: Ο διαδικτυακός διακομιστής ή καλύτερα το back-end κομμάτι χειρίζεται την λογική και τις διαδικασίες για την εξαγωγή αποτελεσμάτων έπειτα από αιτήματα των χρηστών. Διαχειρίζεται με λίγα λόγια όλες τις λειτουργικές διαδικασίες της εφαρμογής και εξυπηρετεί ταυτόχρονα πολλαπλά αιτήματα διαφορετικού είδους.
3. **Διακομιστής Βάσης Δεδομένων**: Η βάση δεδομένων παρέχει τα απαραίτητα δεδομένα για μία διαδικτυακή εφαρμογή. Χειρίζεται τις διαδικασίες που αφορούν τα δεδομένα και σε μία πολυστρωματική αρχιτεκτονική αναλαμβάνει και την λογική με τη βοήθεια αποθηκευμένων διαδικασιών.

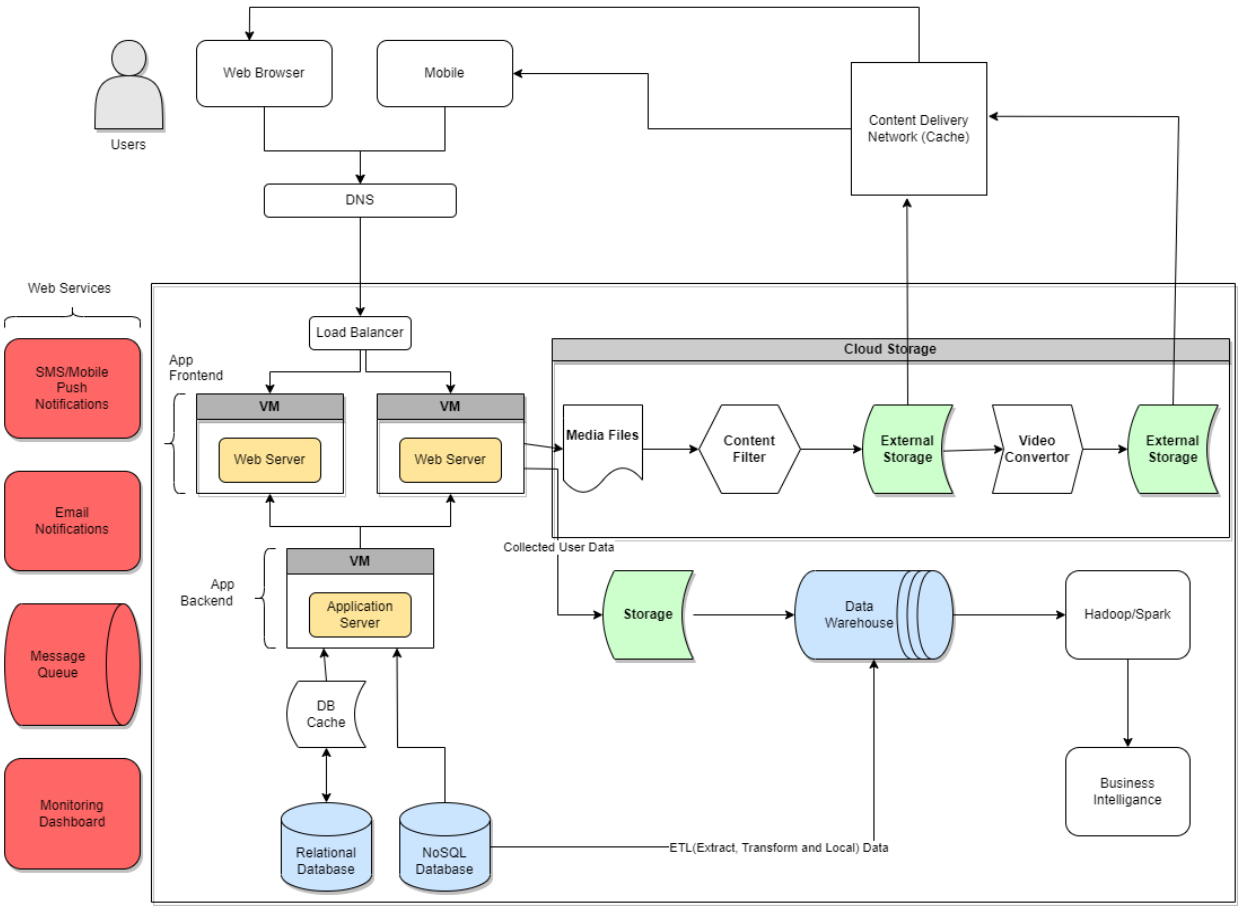
Επίπεδα αρχιτεκτονικής των διαδικτυακών εφαρμογών: Οι σύγχρονες διαδικτυακές εφαρμογές ακολουθούν επίπεδα αρχιτεκτονικής και αποτελούνται από αυτό της παρουσίασης, της επιχειρηματικής λογικής, της συντήρησης και της βάσης. Στις μικρότερες εφαρμογές υπάρχουν τρία επίπεδα λόγω μικρότερων αναγκών, έτσι το επίπεδο της επιχειρηματικής λογικής και της συντήρησης ενοποιούνται σε ένα.



Εικόνα 3.3 Επίπεδα αρχιτεκτονικής διαδικτυακών εφαρμογών

1. **Επίπεδο Παρουσίασης**: Χτίζεται κυρίως με τις τεχνολογίες HTML, CSS, JavaScript και τα frameworks που παρέχονται, έτσι ενεργοποιείται η επικοινωνία μεταξύ της διεπαφής και του περιηγητή για την διευκόλυνση της αλληλεπίδρασης με τον χρήστη.
2. **Επίπεδο Επιχειρηματικής Λογικής**: Ορίζει την επιχειρηματική λογική και τους κανόνες που πρέπει να διαχειρίζονται τα αιτήματα που λαμβάνει από τους περιηγητές, να εκτελεί αυτή τη λογική που συνδέεται με τα αιτήματα και στην συνέχεια να τα προωθεί και πάλι πίσω στο επίπεδο παρουσίασης.
3. **Επίπεδο Συντήρησης**: Ευθύνεται για την διατήρηση των δεδομένων, είναι στενά συνδεδεμένο με το επίπεδο λογικής και αλληλεπιδρά συνεχώς με την βάση δεδομένων για τα λαμβάνει/αποστέλλει δεδομένα.
4. **Επίπεδο Βάσης Δεδομένων**: Γνωστό και ώς επιπεδο υπηρεσίας δεδομένων, αποθηκεύει και εξασφαλίζει την ασφάλεια όλων των δεδομένων διαχωρίζοντας τα από τα επίπεδα λογικής και παρουσίασης.

Κάθε ένα από αυτά τα επίπεδα δουλεύουν απομονωμένα το ένα από το άλλο και επομένως και ανεξάρτητα μεταξύ τους. Παρόλα αυτά τα επίπεδα συνδέονται μεταξύ τους για να διασφαλίζουν την επικοινωνία και εύρυθμη λειτουργία τους ώστε να αλληλεπιδρούν με τον καλύτερο δυνατό τρόπο με τον τελικό χρήστη.



Εικόνα 3.4 Αρχιτεκτονικό διάγραμμα διαδικτυακής εφαρμογής

Στοιχεία αρχιτεκτονικής διαδικτυακών εφαρμογών

- **User Agent**: Οι περιηγητές (User agent) είναι βασικά εργαλεία που βοηθούν τους χρήστες να αλληλεπιδρούν με τους διακομιστές. Οι χρήστες πρακτικά αλληλεπιδρούν με τους διακομιστές μέσω ενός διαδικτυακού περιηγητή όπως Google Chrome, Firefox, Safari και εφαρμογές κινητών συσκευων για Android και iOS πλατφόρμες. Ουσιαστικά στέλνουν αιτήματα στον διακομιστή και λαμβάνουν απαντήσεις τις οποίες απεικονίζουν στον τελικό χρήστη.
- **Σύστημα Ονοματοδοσίας Διαδικτύου (DNS)**: Οι διακομιστές ονομάτων είναι σαν ένα βιβλίο διευθύνσεων για ιστότοπους. Κάθε φορά που ένας χρήστης στέλνει ένα

αίτημα στη διεύθυνση web χρησιμοποιώντας το πρόγραμμα περιήγησης, το πρόγραμμα περιήγησης χρησιμοποιεί το DNS για να βρει την πραγματική διεύθυνση του διακομιστή ιστού (Διεύθυνση IP) προτού μπορέσει να στείλει το αίτημα. Το πρόγραμμα περιήγησης πρέπει να βρει σε ποιον διακομιστή βρίσκεται ο ιστότοπος, ώστε να μπορεί να στείλει αιτήματα HTTP στο σωστό μέρος.

- **Load Blancer**: Το Load Balancer ασχολείται κυρίως με την οριζόντια κλιμάκωση. Κατευθύνει τα εισερχόμενα αιτήματα σε έναν από τους πολλαπλούς διακομιστές και ο εξισορροπητής φορτίου στη συνέχεια στέλνει την απάντηση που λαμβάνει από αυτούς τους διακομιστές στον χρήστη. Συνήθως, οι διακομιστές εφαρμογών Ιστού υπάρχουν με τη μορφή πολλαπλών αντιγράφων που αντικατοπτρίζονται μεταξύ τους για να παρέχουν συνέπεια και διαθεσιμότητα..
- **Web App Server**: Αυτό το στοιχείο επεξεργάζεται το αίτημα ενός χρήστη και στέλνει απαντήσεις σε αυτά τα αιτήματα που πιθανόν να περιλαμβάνουν και έγγραφα (HTML, JSON, XML) πίσω σε ένα πρόγραμμα περιήγησης. Για να εκτελεστεί αυτή η εργασία, συνήθως στηρίζεται σε υποδομές υποστήριξης, όπως βάση δεδομένων, διακομιστής προσωρινής μνήμης, ουρά εργασιών κλπ. Τουλάχιστον δύο διακομιστές, συνδεδεμένοι στο load balancer, καταφέρνουν να επεξεργάζονται τα αιτήματα του χρήστη.
- **Databases**: Η βάση δεδομένων παρέχει εργαλεία για την οργάνωση, την προσθήκη, την αναζήτηση, την ενημέρωση, τη διαγραφή και την εκτέλεση υπολογισμών. Στις περισσότερες περιπτώσεις, οι διακομιστές εφαρμογών Ιστού αλληλεπιδρούν απευθείας με τους διακομιστές εργασιών. Οι βασικότερες και πιο συχνά χρησιμοποιούμενες βάσης δεδομένων είναι MySQL, PostgreSQL, MongoDB, Microsoft SQL Server.
- **Caching Service**: Η υπηρεσία προσωρινής αποθήκευσης παρέχει αποθήκευση δεδομένων, η οποία επιτρέπει στην αποθήκευση και την αναζήτηση δεδομένων. Κάθε φορά που ένας χρήστης λαμβάνει κάποιες πληροφορίες από τον διακομιστή, τα αποτελέσματα αυτής της λειτουργίας πηγαίνουν στην κρυφή μνήμη. Έτσι, τα μελλοντικά αιτήματα επιστρέφουν πιο γρήγορα. Με λίγα λόγια, η προσωρινή αποθήκευση επιτρέπει να ανατρέξουμε στο προηγούμενο αποτέλεσμα για να κάνουμε έναν υπολογισμό πολύ πιο γρήγορο.
- **Services**: Οι υπηρεσίες Ιστού παρέχουν μια κοινή πλατφόρμα που επιτρέπει σε πολλαπλές εφαρμογές που είναι χτισμένες σε διάφορες γλώσσες προγραμματισμού να έχουν τη δυνατότητα να επικοινωνούν μεταξύ τους.Υπάρχουν δύο βασικοί τύποι διαδικτυακών υπηρεσιών:
    - Simple Object Access Protocol, **SOAP** web services

- Representational State Transfer, **REST** web services

Μοντέλα διαδικτυακών εφαρμογών

Υπάρχουν τέσσερα είδη διαδικτυακών μοντέλων και διαχωρίζονται βάση του αριθμού των υπηρεσιών και των βάσεων δεδομένων που χρησιμοποιούνται από μια διαδικτυακή εφαρμογή.[10] [11] [12]

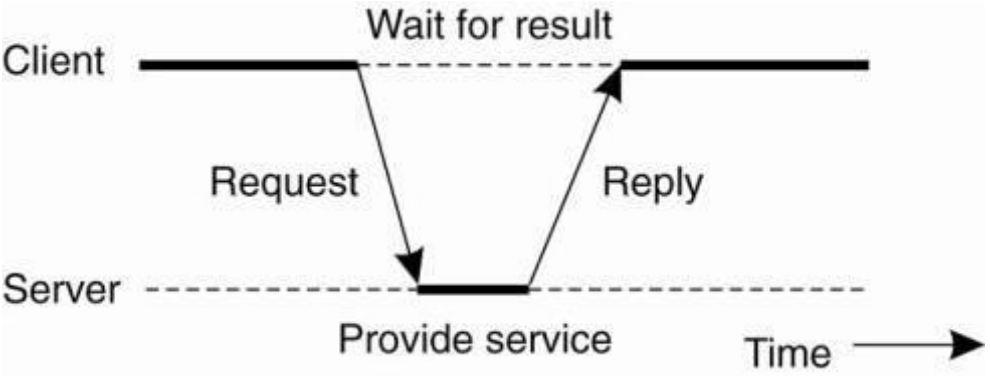
- **Ένας διακομιστής, μία βάση δεδομένων**
- **Πολλαπλοί διακομιστές, μία βάση δεδομένων**
- **Πολλαπλοί διακομιστές, πολλαπλές βάσεις δεδομένων**

3.3 Μοντέλο Πελάτη - Διακομιστή (Client - Server)

Στον σύγχρονο δικτυακό προγραμματισμό μεγάλο μέρος καταλαμβάνει το μοντέλο πελάτη-διακομιστή (client-server). Ένα είδος κλασσικής εφαρμογής client-server είναι ο δεύτερος να αποθηκεύει μεγάλες ποσότητες δεδομένων σε έναν server μεγάλων δυνατοτήτων, ενώ την ευθύνη για τον χειρισμό της λογικής και της διασύνδεσης με τον χρήστη του προγράμματος έχει το λογισμικό-πελάτης που εκτελείται σε σχετικά φθηνούς προσωπικούς υπολογιστές. Το μοντέλο client-server είναι ένα σύστημα στο οποίο το δίκτυο ενώνει διάφορους υπολογιστικούς πόρους, έτσι οι πελάτες μπορούν να ζητούν υπηρεσίες από έναν διακομιστή ο οποίος σερβίρει τις πληροφορίες. Οι υπηρεσίες που μπορεί να προσφέρει ένας διακομιστής ποικίλουν ανάλογα το είδος, για παράδειγμα μπορεί να είναι υπηρεσίες συστήματος αρχείων, βάσης δεδομένων κλπ. Το μοντέλο client-server βασίζεται στην λογική αίτησης/απάντησης (request/response), δηλαδή ο πελάτης στέλνει ένα μήνυμα αίτησης ζητώντας από τον εξυπηρετητή κάποια υπηρεσία, από την δική του πλευρά ο εξυπηρετητής εκτελεί τη διαδικασία και επιστρέφει τα δεδομένα που ζητήθηκαν ή ένα μήνυμα λάθους. Ο εξυπηρετητής εφόσον παραλάβει το αίτημα κάνει μια σειρά από ενέργειες και ανταποκρίνεται στον πελάτη, αυτές οι ενέργειες είτε ενεργοποιούνται άμεσα είτε μπαίνουν σε ουρά προς διεκπεραίωση. Η αρμοδιότητα του εξυπηρετητή είναι ακριβώς αυτό που λέει και ο όρος του να εξυπηρετεί συνεχώς τα αιτήματα των πελατών, να ετοιμάζει τα μηνύματα απάντησης με τα δεδομένα που ζητήθηκαν ή την πληροφορία αντίστοιχα ή αν δεν ήταν δυνατόν η διεκπεραίωση της με κάποια αιτιολογία.

Οι διακομιστές χωρίζονται σε έξι βασικές κατηγορίες

- Διακομιστές Εφαρμογών (Application servers)
- Διακομιστές Πληροφοριών (Data servers)
- Διακομιστές Υπολογισμών (Compute servers)
- Διακομιστές Βάσεων Δεδομένων (Database servers)
- Διακομιστές Πόρων ή Επικοινωνιών (Recourse or Communications servers)
- Διακομιστές Συναλλαγών (Transaction servers)
-



Εικόνα 3.5 Επικοινωνία πελάτη διακομιστή σε αναπαράσταση

(https://eclass.teicrete.gr/modules/document/file.php/TP183/%CE%98%CE%95%CE%A9%CE%A1%CE%99%CE%91/01.%CE%94%CE%B9%CE%B1%CF%86%CE%AC%CE%BD%CE%B5%CE%B9%CE%B5%CF%82/DS-1213E-slides/DS-L9b-05-Client-Server-Model.pdf)

Τα βήματα που ακολουθεί το μοντέλο για να αλληλεπιδρά client-server:

- Οι χρήστες εισάγουν το **URL** (Uniform Resource Locator) της ιστοσελίδας ή του αρχείου στον περιηγητή και στην συνέχεια αυτός το ζητά από τον DNS διακομιστή.
- Ο DNS διακομιστής αναζητά την διεύθυνση του διαδικτυακού διακομιστή.
- Ο DNS διακομιστής απαντά με την IP διεύθυνση του διαδικτυακού διακομιστή.
- Ο περιηγητής στη συνέχεια στέλνει το αίτημα του μέσω του πρωτοκόλλου HTTP/HTTPS στην διεύθυνση του διαδικτυακού διακομιστή που προηγουμένως είχε δώσει ο DNS διακομιστής.
- Ο διακομιστής μετά τις απαραίτητες διεργασίες του αποστέλλει τα απαραίτητα αρχεία της ιστοσελίδας.
- Ο περιηγητής στην συνέχεια αναλαμβάνει να απεικονίσει τα απαραίτητα αρχεία της ιστοσελίδας. Αυτή η διαδικασία γίνεται με την βοήθεια του **DOM** (Document Object Model) διερμηνέα, τον CSS διερμηνέα και το **JS Engine**.

Τα πλεονεκτήματα χρήσης του μοντέλου client-server:

- Κεντρικό σύστημα που συγκεντρώνει όλα τα δεδομένα σε ένα σημείο.
- Μείωση του κόστους συντήρησης.
- Δυνατότητα επαναφοράς δεδομένων.
- Η χωρητικότητα και υποστηριξη των πελατών/διακομιστών είναι διαχωρισμένη.

Τα μειονεκτήματα του μοντέλου client-server:

- Οι διακομιστές είναι ευάλωτοι σε επιθέσεις κορεσμού με αποτέλεσμα την άρνηση υπηρεσιών (DOS attacks).
- Οι πελάτες μπορεί να είναι επιρρεπής σε ιούς και ευπάθειες, όπως Trojans και Worms.
- Τα πακέτα δεδομένων ενδέχεται να τροποποιηθούν κατά τη μετάδοση.
- Το ηλεκτρονικό ψάρεμα ή η συλλογή διαπιστευτηρίων σύνδεσης ή άλλων χρήσιμων πληροφοριών του χρήστη είναι σύνηθες φαινόμενο, όπως και οι επιθέσεις MITM (Man in the Middle).

Ένα παράδειγμα που χρησιμοποιεί το μοντέλο client-server είναι το **FTP** (File Transfer Protocol). Το FTP χρησιμοποιεί διαφορετικά πρωτόκολλα εφαρμογών και διαφορετικό λογισμικό, αλλά εξακολουθεί να υιοθετεί τη βασική διάκριση σε διακομιστές FTP που στέλνουν αρχεία και πελάτες FTP που λαμβάνουν αρχεία. Επειδή πολλοί χρησιμοποιούν το FTP για να "*ανεβάζουν αρχεία*" από τον πελάτη στον διακομιστή, η μεταφορά δεδομένων συνήθως γίνεται προς μία μόνο κατεύθυνση. Παρόλα αυτά εξακολουθεί να ισχύει το μοντέλο client-server αφού ο πελάτης FTP ξεκινά τη σύνδεση και ο διακομιστής FTP ανταποκρίνεται.[8] [13]

Πρότυπα του Διαδικτύου

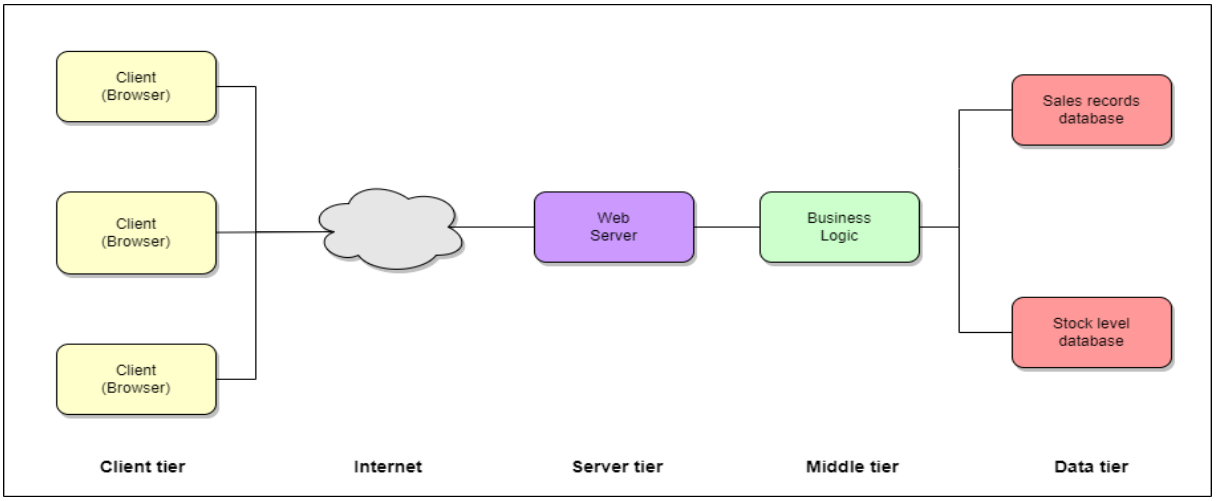
Υπάρχουν πολλοί οργανισμοί προτυποποίησης στον κόσμο, τα περισσότερα πρότυπα που σχετίζονται με τον δικτυακό προγραμματισμό και τα πρωτόκολλα του επιπέδου εφαρμογών παράγονται από δύο οργανισμούς: την Ομάδα Εργασίας Μελέτης Διαδικτύου (Internet Engineering Task Force, IETF) και την Κοινοπραξία Παγκόσμιου Ιστού (World Wide Web, W3C). Η IETF είναι ένας σχετικά ανεπίσημος, δημοκρατικός φορέας στον οποίο μπορεί να συμμετάσχει οποιοδήποτε ενδιαφερόμενο μέρος. Για τη δημιουργία των προτύπων της υιοθετεί την εξής λογική: "*πλειοψηφία , όχι ομοφωνία, και κώδικας που λειτουργεί στην πράξη*". Τείνει να βασίζεται σε λειτουργικές, πρακτικές υλοποιήσεις παρά να ορίζει προδιαγραφές γι' αυτές. Μεταξύ των προτύπων της συγκαταλέγονται τα TCP/IP, MIME (Multipurpose Internet Mail Extensions), Γενικές Επεκτάσεις Ταχυδρομείου Διαδικτύου και

SMTP. Στην αντίπερα όχθη, η W3C είναι μια κοινοπραξία κατασκευαστών/προμηθευτών ελεγχόμενη από τις εταιρίες-μέλη που πληρώνουν συνδρομή, η οποία απαγορεύει ρητά τη συμμετοχή φυσικών προσώπων. Κατά κύριο λόγο, η W3C ακολουθεί την τακτική να ορίζει τα πρότυπα πριν εμφανιστούν οι σχετικές υλοποιήσεις. Στο πρότυπα της περιλαμβάνονται τα HTTP, HTML (Hypertext Markup Language, Γλώσσα Σήμανσης Υπερκειμένου) και XML (eXtensible Markup Language, Επεκτάσιμη Γλώσσα Σήμανσης). [9]

Κατηγορίες αρχιτεκτονικής του μοντέλου client-server

Υπάρχουν τέσσερις κατηγορίες αρχιτεκτονικής και είναι οι εξής:

1. **Αρχιτεκτονική 1-επιπέδου**: αποτελείται από ένα απλό πρόγραμμα που εκτελείται σε έναν μόνο υπολογιστή χωρίς να απαιτείται πρόσβαση στο δίκτυο. Τα αιτήματα χρηστών δεν διαχειρίζονται κανένα πρωτόκολλο δικτύου, επομένως ο κώδικας είναι απλός και το δίκτυο απαλλάσσεται από την επιπλέον κίνηση.
2. **Αρχιτεκτονική 2-επιπέδων**: αποτελείται από τον πελάτη, τον διακομιστή και το πρωτόκολλο που συνδέει τα δύο επίπεδα. Ο κώδικας γραφικής διεπαφής χρήστη βρίσκεται στον κεντρικό υπολογιστή πελάτη και η λογική βρίσκεται στον κεντρικό υπολογιστή διακομιστή. Το GUI client-server είναι γραμμένο σε γλώσσες υψηλού επιπέδου.
3. **Αρχιτεκτονική 3-επιπέδων**: αποτελείται από ένα επίπεδο παρουσίασης, το οποίο είναι το επίπεδο διεπαφής χρήστη, το επίπεδο εφαρμογής, το οποίο είναι το επίπεδο υπηρεσίας που εκτελεί λεπτομερή επεξεργασία και το επίπεδο δεδομένων, το οποίο αποτελείται από έναν διακομιστή βάσης δεδομένων που αποθηκεύει πληροφορίες.
4. **Αρχιτεκτονική Ν-επιπέδων**: διαιρεί μια εφαρμογή σε λογικά επίπεδα, τα οποία διαχωρίζουν τις ευθύνες και διαχειρίζονται εξαρτήσεις, και φυσικές βαθμίδες, που εκτελούνται σε ξεχωριστά μηχανήματα, βελτιώνουν την επεκτασιμότητα και προσθέτουν καθυστέρηση από την πρόσθετη επικοινωνία δικτύου. Η αρχιτεκτονική Ν-επιπέδων μπορεί να είναι κλειστού επιπέδου, στο οποίο ένα επίπεδο μπορεί να επικοινωνεί μόνο με το επόμενο επίπεδο προς τα κάτω, ή ανοιχτού επιπέδου, στο οποίο ένα επίπεδο μπορεί να επικοινωνεί με οποιαδήποτε στρώματα κάτω από αυτό.[14] [15]



Εικόνα 3.6 Απεικόνιση της N-επιπέδων αρχιτεκτονικής

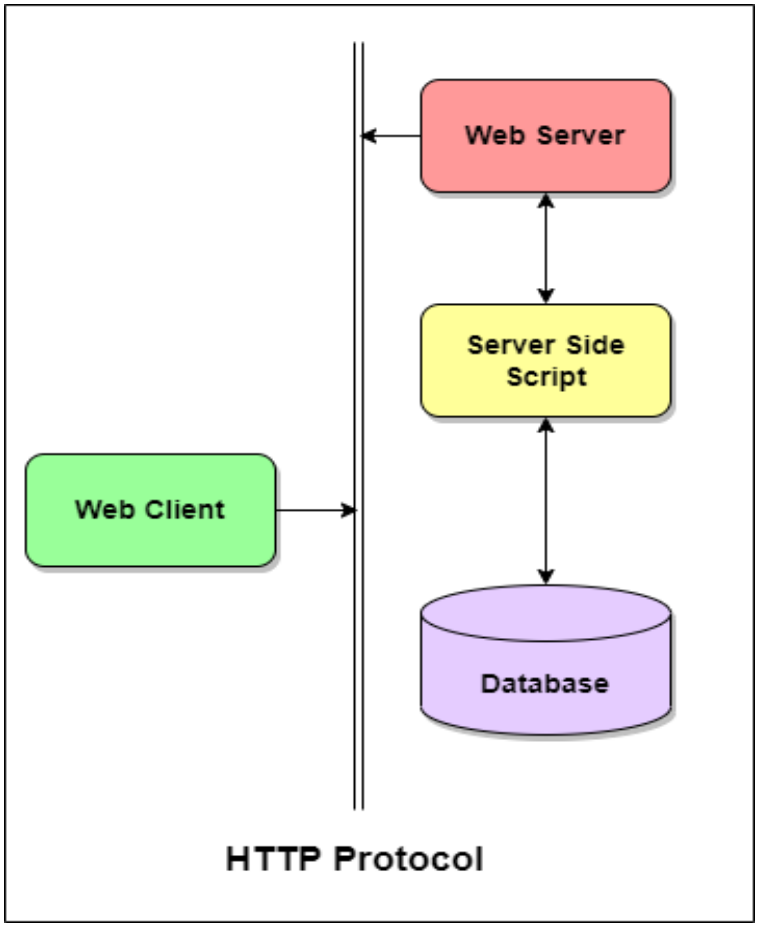
3.4 HTTP

Το HTTP (HyperText Transfer Protocol) αποτελεί ένα πρωτόκολλο σε επίπεδο εφαρμογής το οποίο είναι υπεύθυνο για την κατανομή και την συνεργασία μεταξύ συστημάτων και εφαρμογών. Το πρωτόκολλο αυτό αποτελεί την βάση ή αλλιώς το θεμέλιο για την επικοινωνία δεδομένων για το Διαδίκτυο από το 1990. Το HTTP είναι ένα γενικό πρωτόκολλο που δεν διατηρεί καταστάσεις, μπορεί να χρησιμοποιηθεί για άλλους σκοπούς με τη χρήση επεκτάσεων των μεθόδων αιτήματος του, κωδικών σφάλματος και κεφαλίδων. Πιο συγκεκριμένα, το HTTP είναι ένα πρωτόκολλο επικοινωνίας που βασίζεται στο TCP/IP και χρησιμοποιείται για να μεταφέρει HTML αρχεία, εικόνες, αποτελέσματα ερωτήσεων (queries) στο Διαδίκτυο. Η προκαθορισμένη πόρτα επικοινωνίας είναι η 80 και για τις ασφαλείς συνδέσεις η 443, φυσικά μπορούν να χρησιμοποιηθούν και άλλες πόρτες χωρίς κανένα πρόβλημα. Μέσω κάποιων προκαθορισμένων λειτουργιών παρέχει έναν σταθερό και καθορισμένο τρόπο επικοινωνίας μεταξύ υπολογιστικών μονάδων. Μέσω προδιαγραφών επιτρέπεται στου χρήστες να ζητούν δεδομένα από έναν διακομιστή καθώς και να λαμβάνουν τα δεδομένα αυτά με συγκεκριμένο τρόπο που καθορίζει τα δεδομένα, τις κεφαλίδες και ενδεχόμενα σφάλματα. Όταν επισκεπτόμαστε ιστότοπους μέσω ενός προγράμματος περιήγησης, ολόκληρη η σύνδεση πραγματοποιείται μέσω HTTP. Το HTTP είναι ένας από τους βασικούς άξονες του ιστού από τις αρχές της δεκαετίας του '90. Τις τελευταίες δεκαετίες, έχει εξελιχθεί για να γίνει πιο αποτελεσματικό και για αυτό αναπτύχθηκε το HTTP/2, το οποίο επιτρέπει στους πελάτες να φορτώνουν πόρους ταυτόχρονα αντί της ασύγχρονη τεχνολογίας που ίσχυε παλαιότερα. Αυτό έχει ως αποτέλεσμα μια τεράστια ώθηση απόδοσης. Το 2022, το

46% του ιστού χρησιμοποιεί HTTP/2. Τώρα, υπάρχουν ήδη συζητήσεις σχετικά με την υιοθέτηση του HTTP/3.

Βασικά χαρακτηριστικά

- **Το HTTP είναι ασύνδετο μεταξύ διαφορετικών κλήσεων**: Ο πελάτης ενός HTTP, δηλαδή αυτός που ξεκινά πρώτος την συνομιλία (Περιήγητης) αποστέλλει ένα αίτημα HTTP και αφού υποβληθεί το αίτημα, ο πελάτης-περιηγητής περιμένει την απάντηση. Ο διακομιστής με την σειρά του επεξεργάζεται το αίτημα που δέχτηκε και στέλνει μια απάντηση στον πελάτη και έτσι ολοκληρώνεται και αποσυνδέεται η σύνδεση. Έτσι, ο πελάτης και ο διακομιστής γνωρίζουν ο ένας τον άλλον μόνο κατά την τρέχουσα αίτηση και απόκριση. Για την αποστολή περαιτέρω αιτημάτων γίνονται νέες συνδέσεις, όπου ο πελάτης και ο διακομιστής εγκαθιδρύουν νέες συνδέσεις μεταξύ τους.
- **Το HTTP είναι ανεξάρτητο από τα μέσα**: Αυτό σημαίνει ότι οποιοσδήποτε τύπος δεδομένων μπορεί να αποσταλεί μέσω του HTTP, εφόσον τόσο ο πελάτης όσο και ο διακομιστής γνωρίζουν πώς να διαχειριστούν το περιεχόμενο δεδομένων. Απαιτείται τόσο ο πελάτης όσο και ο διακομιστής να καθορίσουν τον τύπο περιεχομένου χρησιμοποιώντας κατάλληλο τύπο MIME.
- **Το HTTP δεν διατηρεί καταστάσεις**: Το HTTP είναι ασύνδετο και είναι άμεσο αποτέλεσμα του HTTP ως πρωτόκολλου χωρίς κατάσταση. Ο διακομιστής και ο πελάτης γνωρίζουν ο ένας τον άλλον μόνο κατά τη διάρκεια ενός τρέχοντος αιτήματος. Για οποιαδήποτε νέο αίτημα χρειάζεται εκ νέου σύνδεση για την επικοινωνία client-server. Λόγω αυτής της φύσης του πρωτοκόλλου, ούτε ο πελάτης ούτε το πρόγραμμα περιήγησης μπορούν να διατηρήσουν πληροφορίες μεταξύ διαφορετικών αιτημάτων στις ιστοσελίδες - εφαρμογές.



Εικόνα 3.7 HTTP Αρχιτεκτονική

HTTP Αρχιτεκτονική: Το πρωτόκολλο HTTP είναι ένα πρωτόκολλο αιτήματος-απόκρισης που βασίζεται στην αρχιτεκτονική του μοντέλου client-server, όπου τα προγράμματα περιήγησης Ιστού, οι μηχανές αναζήτησης, κλπ λειτουργούν σαν πελάτες HTTP και ο διακομιστής Ιστού λειτουργεί ως διακομιστής. Ως πελάτης μπορεί να οριστεί ο υπολογιστής, αυτός στέλνει ένα HTTP αίτημα στον διακομιστή με τη μορφή μιας μεθόδου αιτήματος, ενός URI και μιας έκδοσης πρωτοκόλλου, ακολουθούμενο από ένα μήνυμα παρόμοιο με το MIME που περιέχει τροποποιητές αιτήματος, πληροφορίες πελάτη και πιθανό περιεχόμενο σώματος μέσω μιας σύνδεσης TCP/IP. Ως εξυπηρετητής μπορεί να οριστεί ο διακομιστής, αυτός αποκρίνεται με μια HTTP απόκριση κατάστασης, συμπεριλαμβανομένης της έκδοσης πρωτοκόλλου του μηνύματος και έναν κωδικό επιτυχίας ή σφάλματος, ακολουθούμενο από ένα μήνυμα τύπου MIME που περιέχει πληροφορίες διακομιστή, metadata οντοτήτων και ενδεχομένως ένα περιεχόμενο οντότητας.

Ένα αίτημα HTTP που έχει συνταχθεί σωστά περιέχει τα ακόλουθα στοιχεία:

1. Το **URI**, είναι μια γραμμή αιτήματος που αφορά την μέθοδο, το μονοπάτι που είναι το αναγνωριστικό πόρου και οι παράμετροι που ίσως έχει, καθώς και η έκδοση του πρωτοκόλλου HTTP (1, 2 η S για secure).
2. Το **Header**, είναι μια σειρά από κεφαλίδες HTTP ή πεδία κεφαλίδας. Οι κεφαλίδες HTTP γράφονται σε ένα μήνυμα για να παρέχουν στον παραλήπτη πληροφορίες σχετικά με το μήνυμα, τον αποστολέα και τον τρόπο με τον οποίο ο αποστολέας θέλει να επικοινωνήσει με τον παραλήπτη. Κάθε κεφαλίδα HTTP αποτελείται από ένα όνομα και μια τιμή. Οι προδιαγραφές του πρωτοκόλλου HTTP ορίζουν το τυπικό σύνολο κεφαλίδων HTTP και περιγράφουν τον τρόπο σωστής χρήσης τους. Τα μηνύματα HTTP μπορούν επίσης να περιλαμβάνουν κεφαλίδες επέκτασης, οι οποίες δεν αποτελούν μέρος των προδιαγραφών HTTP/1.1 ή HTTP/1.0.
3. Το **Body**, ένα σώμα μηνύματος, εφόσον χρειάζεται. Το κύριο περιεχόμενο οποιουδήποτε μηνύματος HTTP μπορεί να αναφέρεται ως σώμα μηνύματος ή σώμα οντότητας. Τεχνικά, το σώμα της οντότητας είναι το πραγματικό περιεχόμενο του μηνύματος. Το σώμα του μηνύματος περιέχει το σώμα της οντότητας, το οποίο μπορεί να βρίσκεται στην αρχική του κατάσταση ή μπορεί να κωδικοποιηθεί με κάποιο τρόπο για μεταφορά, όπως με τη διάσπαση σε κομμάτια (τεμαχισμένη κωδικοποίηση μεταφοράς). Το σώμα μηνύματος ενός αιτήματος μπορεί να αναφέρεται για ευκολία ως σώμα αιτήματος.

Μια απάντηση HTTP γίνεται από έναν διακομιστή σε έναν πελάτη. Ο στόχος της απάντησης είναι να παρέχει στον πελάτη τον πόρο που ζήτησε ή να ενημερώσει τον πελάτη ότι η ενέργεια που ζήτησε έχει πραγματοποιηθεί ή αλλιώς να ενημερώσει τον πελάτη ότι παρουσιάστηκε σφάλμα κατά την επεξεργασία του αιτήματός του. Μια απάντηση HTTP περιέχει:

1. Το **Status**, είναι μια γραμμή κατάστασης. Η γραμμή κατάστασης περιέχει την έκδοση του HTTP, το status code που είναι ένας τριψήφιος αριθμός που προσδιορίζει το αποτέλεσμα της αίτησης και το status text το οποίο είναι κείμενο αναγνώσιμο από τον άνθρωπο που συνοψίζει το νόημα του κωδικού κατάστασης.
2. Το **Header**, μια σειρά από κεφαλίδες HTTP ή πεδία κεφαλίδας. Οι κεφαλίδες HTTP για την απόκριση ενός διακομιστή περιέχουν πληροφορίες που μπορεί να χρησιμοποιήσει ένας πελάτης για να μάθει περισσότερα σχετικά με την απόκριση και σχετικά με τον διακομιστή που την έστειλε. Αυτές οι πληροφορίες μπορούν να βοηθήσουν τον πελάτη να εμφανίσει την απάντηση σε έναν χρήστη, να αποθηκεύσει

(ή να αποθηκεύσει προσωρινά) την απάντηση για μελλοντική χρήση και να υποβάλει περαιτέρω αιτήματα στον διακομιστή.

3. Το **Body**, είναι ένα σώμα μηνύματος, που συνήθως είναι απαραίτητο για απάντηση σε ένα επιτυχημένο αίτημα, το σώμα του μηνύματος περιέχει είτε τον πόρο που ζητά ο πελάτης είτε κάποιες πληροφορίες σχετικά με την κατάσταση της ενέργειας που ζητήθηκε από τον πελάτη. Για απάντηση σε ένα μη επιτυχές αίτημα, το σώμα του μηνύματος ενδέχεται να παρέχει περαιτέρω πληροφορίες σχετικά με τους λόγους του σφάλματος ή σχετικά με κάποια ενέργεια που πρέπει να κάνει ο πελάτης για να ολοκληρώσει το αίτημα με επιτυχία.

Η υποβολή αιτήματος HTTP περιλαμβάνει την αποστολή ενός μηνύματος στο διακομιστή λήψης σε συγκεκριμένη μορφή. Ο διακομιστής επιστρέφει μια απάντηση και ο πελάτης αναλαμβάνει δράση. Για παράδειγμα, μπορεί να φορτώσει πόρους ή να ανακατευθύνει εκ νέου σε άλλη σελίδα. Όταν λαμβάνετε ένα σφάλμα HTTP, είναι συνήθως επειδή ο διακομιστής δεν μπορεί να εκπληρώσει το αίτημά που του ζητήθηκε. Ο κωδικός σφάλματος που λαμβάνετε πρέπει να εξηγεί τι είδους σφάλμα προέκυψε και για ποιό λόγο. Οι πιο συνηθισμένες αιτίες σφαλμάτων HTTP περιλαμβάνουν την αδυναμία σύνδεσης με τον διακομιστή και εύρεσης των πόρων που έχουν ζητηθεί. Στο παρακάτω πίνακα μπορούμε να δούμε τις 5 κατηγορίες κωδικών που επιστρέφει η απάντηση ενός διακομιστή. αποτελείται από ένα τριψήφιο αριθμό όπου ο πρώτος αριθμός δηλώνει και την κατηγορία στην οποία ανήκει ο κωδικός, στην συνεχεια οι άλλοι δύο αριθμοί απευθύνονται στην σειρά με την οποία οργανώθηκαν οι κωδικοί στην συγκεκριμένη κατηγορία και συνοδεύονται από μια σύντομη περιγραφή για τον κωδικό.

S.N	**Κωδικός και Περιγραφή**
1	**1xx: Information** Δηλώνει ότι το αίτημα έχει παραληφθεί και η διαδικασία προχωρά κανονικά. (**100** Continue, **101** Switching Protocols)
2	**2xx: Success** Δηλώνει ότι η ενέργεια ήταν επιτυχής, κατανοητή και αποδεκτή. (**200** OK, **201** Created, **202** Accepted, κλπ)

3	**3xx: Redirection** Δηλώνει ότι πρέπει να γίνουν περαιτέρω ενέργειες για την διεκπεραίωση του αιτήματος. (**301** Moved Permanently, **304** Not Modified, κλπ)
4	**4xx: Client Error** Δηλώνει ότι το αίτημα περιλαμβάνει λανθασμένο συντακτικό και δεν μπορεί να ολοκληρωθεί. (**400** Bad Request, **401** Unauthorized, **402** Payment Required, **403** Forbidden, **404** Not Found, κλπ)
5	**5xx: Server Error** Δηλώνει ότι ο διακομιστής απέτυχε να διεκπεραιώσει ένα αποδεκτό αίτημα. (**500** Internal Server Error, **502** Bad Gateway, **503** Service Unavailable, κλπ)

Πίνακας 3.1 Κατηγορίες HTTP σφαλμάτων

Τι είναι και πως λειτουργεί ένα HTTP αίτημα

Καθώς το πρόγραμμα περιήγησής συνδέεται με τον διακομιστή και είτε ζητά έναν συγκεκριμένο πόρο είτε στέλνει δεδομένα σε αυτόν πραγματοποιεί ένα αίτημα HTTP. Υπάρχουν διάφοροι τύποι μεθόδων αιτήματος HTTP, οι οποίοι αλλάζουν εντελώς τον τύπο απόκρισης που λαμβάνετε από τον διακομιστή. Τα πιο συνηθισμένα είναι:

1. **GET**: Αυτή είναι η πιο συχνά χρησιμοποιούμενη μέθοδος αιτήματος HTTP. Ένα αίτημα GET ζητά από τον διακομιστή μια συγκεκριμένη πληροφορία ή πόρο. Όταν συνδέετε σε έναν ιστότοπο, το πρόγραμμα περιήγησής συνήθως στέλνει πολλά αιτήματα GET για να λάβει τα δεδομένα που χρειάζεται για τη φόρτωση της σελίδας.
2. **HEAD**: Με ένα αίτημα HEAD, λαμβάνονται μόνο οι πληροφορίες κεφαλίδας της σελίδας που θέλουμε να φορτωθούν. Μπορούμε να χρησιμοποιήσουμε αυτό το τύπο αιτήματος για να μάθουμε το μέγεθος ενός εγγράφου πριν το κατεβάσουμε χρησιμοποιώντας το GET.
3. **POST**: Το πρόγραμμα περιήγησής σας χρησιμοποιεί τη μέθοδο αιτήματος POST HTTP όταν χρειάζεται να στείλει δεδομένα στον διακομιστή. Για παράδειγμα, αν συμπληρώσουμε μια φόρμα επικοινωνίας σε έναν ιστότοπο και την υποβάλουμε, τότε χρησιμοποιείτε ένα αίτημα POST, ώστε ο διακομιστής να λάβει αυτές τις πληροφορίες.

4. **PUT**: Τα αιτήματα PUT είναι παρόμοιας σε λειτουργικότητα με τη μέθοδο POST. Ωστόσο, αντί να υποβληθούν εξ αρχής δεδομένα, χρησιμοποιείτε το αιτήματα PUT για να ενημερωθούν πληροφορίες που υπάρχουν ήδη στον τελικό διακομιστή.

Υπάρχουν και άλλοι τύποι αιτημάτων HTTP που μπορούμε να χρησιμοποιήσουμε, συμπεριλαμβανομένων των μεθόδων **DELETE**, **PATCH**, **CONNECT**, **TRACE** και **OPTIONS**.[16] [17] [18]

3.5 Web Services & RESTful

Το Internet είναι ένας τρόπος να συνδέονται εκατοντάδες χιλιάδες υπολογιστικές μηχανές διαφόρων τύπων που μπορούν να ανήκουν κιόλας σε πολλαπλά δίκτυα. Μια υπηρεσία Ιστού είναι μια τυποποιημένη μέθοδος για τη διάδοση μηνυμάτων μεταξύ εφαρμογών πελάτη και διακομιστή. Πιο συγκεκριμένα είναι μια λειτουργική μονάδα λογισμικού που προορίζεται να εκτελεί ένα συγκεκριμένο σύνολο λειτουργιών και να παρέχει λειτουργικότητα στον πελάτη που επικαλέστηκε την υπηρεσία. Μια διαδικτυακή υπηρεσία είναι ένα σύνολο ανοιχτών πρωτοκόλλων και προτύπων που επιτρέπουν την ανταλλαγή δεδομένων μεταξύ διαφορετικών εφαρμογών ή συστημάτων. Αν προσπαθήσουμε να δώσουμε έναν ορισμό για τις διαδικτυακές υπηρεσίες είναι: "*Μια διαδικτυακή υπηρεσία είναι μια συλλογή ανοιχτών πρωτοκόλλων και προτύπων που χρησιμοποιούνται για την ανταλλαγή δεδομένων μεταξύ εφαρμογών ή συστημάτων. Οι εφαρμογές λογισμικού είναι γραμμένες σε διάφορες γλώσσες προγραμματισμού και εκτελούνται σε διάφορες πλατφόρμες, μπορούν να χρησιμοποιούν υπηρεσίες Ιστού για την ανταλλαγή δεδομένων μέσω δικτύων υπολογιστών όπως το Διαδίκτυο με τρόπο παρόμοιο με την επικοινωνία μεταξύ διεργασιών σε έναν μόνο υπολογιστή. Αυτή η διαλειτουργικότητα (π.χ. μεταξύ JavaScript και Golang ή εφαρμογών Windows και Linux) οφείλεται στη χρήση ανοικτών προτύπων.*" [19]

Οι λειτουργίες που προσφέρει μια διαδικτυακή υπηρεσία:

- Διατίθεται μέσω Διαδικτύου ή δικτύων intranet.
- Τυποποιημένο σύστημα ανταλλαγής μηνυμάτων XML.
- Ανεξάρτητο από ένα λειτουργικό σύστημα ή γλώσσα προγραμματισμού.
- Αυτο-περιγραφή μέσω τυπικής γλώσσας XML.

Μια υπηρεσία Ιστού υποστηρίζει την επικοινωνία μεταξύ πολλών εφαρμογών με HTML, XML, WSDL, SOAP και άλλα ανοιχτά πρότυπα. Ένα RESTful API χρησιμοποιεί υπάρχουσες μεθοδολογίες HTTP που ορίζονται από το πρωτόκολλο RFC 2616.

Οι βασικοί τύποι των διαδικτυακών υπηρεσιών είναι:

- **XML-RPC** (Remote Procedure Call): αποτελεί το πιο βασικό πρωτόκολλο XML για την ανταλλαγή δεδομένων μεταξύ μιας μεγάλης ποικιλίας συσκευών σε ένα δίκτυο. Χρησιμοποιεί HTTP για γρήγορη και εύκολη μεταφορά δεδομένων και επικοινωνίας άλλων πληροφοριών από πελάτη σε διακομιστή.
- **UDDI** (Universal Description, Discovery and Integration): είναι ένα πρότυπο που βασίζεται σε XML για τη λεπτομέρεια, τη δημοσίευση και την ανακάλυψη υπηρεσιών web. Είναι βασικά ένα μητρώο Διαδικτύου για επιχειρήσεις σε όλο τον κόσμο. Στόχος είναι ο εξορθολογισμός των ψηφιακών συναλλαγών και του ηλεκτρονικού εμπορίου μεταξύ των συστημάτων της εταιρείας.
- **SOAP**: είναι ένα πρωτόκολλο υπηρεσίας Ιστού που βασίζεται σε XML για την ανταλλαγή δεδομένων και εγγράφων μέσω HTTP ή SMTP (Simple Mail Transfer Protocol). Επιτρέπει σε ανεξάρτητες διεργασίες που λειτουργούν σε διαφορετικά συστήματα να επικοινωνούν χρησιμοποιώντας XML.
- **REST**: το πρωτόκολλο αυτό παρέχει επικοινωνία και συνδεσιμότητα μεταξύ συσκευών και του διαδικτύου για εργασίες που βασίζονται σε APIs. Οι περισσότερες υπηρεσίες RESTful χρησιμοποιούν το HTTP ως πρωτόκολλο υποστήριξης. Το πρωτόκολλο αυτό επιλέχθηκε για την ανάπτυξη της διαδικτυακής μας εφαρμογής.

Πολλές φορές οι δύο έννοιες των διαδικτυακών υπηρεσιών και των APIs μπερδεύονται από τους χρήστες ενώ στην πραγματικότητα εργάζονται σε ξεχωριστό επίπεδο. Πιο συγκεκριμένα οι περισσότερες υπηρεσίες Ιστού παρέχουν ένα API, το οποίο, με το σύνολο των εντολών και των λειτουργιών του, χρησιμοποιείται για την ανάκτηση δεδομένων. Για παράδειγμα το Facebook παρέχει ένα API που εξουσιοδοτεί έναν προγραμματιστή να έχει πρόσβαση σε δημοσιεύσεις από έναν διακομιστή και στη συνέχεια συλλέγει δεδομένα σε μορφή JSON. Όλες οι υπηρεσίες ιστού μπορεί να είναι APIs, αλλά δεν μπορούν όλα τα APIs να είναι υπηρεσίες ιστού.

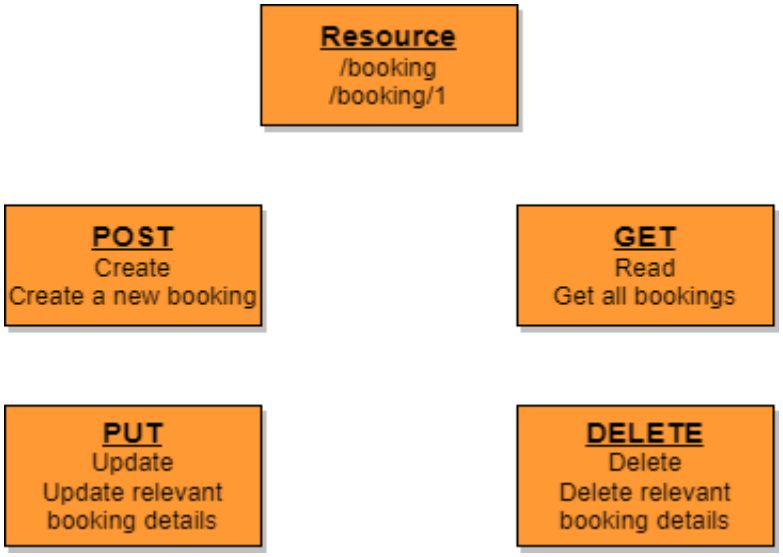
Το 2004, η κοινοπραξία Ιστού κυκλοφόρησε επίσης τον ορισμό ενός πρόσθετου προτύπου που ονομάζεται RESTful. Τα τελευταία χρόνια, αυτό το πρότυπο έχει γίνει αρκετά δημοφιλές και χρησιμοποιείται από τους πιο δημοφιλείς ιστοτόπους σε όλο τον κόσμο. Το REST είναι ένας τρόπος πρόσβασης σε πόρους που βρίσκονται σε ένα συγκεκριμένο περιβάλλον. Για παράδειγμα, θα μπορούσαμε να έχουμε έναν διακομιστή που θα μπορούσε να φιλοξενεί σημαντικά έγγραφα, εικόνες ή βίντεο, όλα αυτά αποτελούν παραδείγματα πόρων. Εάν ένας πελάτης, για παράδειγμα σε ένα πρόγραμμα περιήγησης ιστού χρειάζεται οποιονδήποτε από αυτούς τους πόρους, πρέπει να στείλει ένα αίτημα στον διακομιστή για πρόσβαση σε αυτούς τους πόρους. Παρακάτω έχουμε αναλυτικά παραδείγματα:

1. **Resources**: Το πρώτο βασικό στοιχείο είναι ο ίδιος ο πόρος. Ας υποθέσουμε ότι μια εφαρμογή Ιστού σε έναν διακομιστή έχει αρχεία πολλών κρατήσεων. Ας υποθέσουμε ότι η διεύθυνση URL της εφαρμογής Ιστού είναι **http://khakibee.com**. Τώρα για να αποκτήσουμε πρόσβαση σε έναν πόρο αρχείου κρατήσεων μέσω υπηρεσιών REST, μπορούμε με την εντολή **http://khakibee.com/booking/1**. Αυτή η εντολή λέει στον διακομιστή web να μας παράσχει τα στοιχεία της κράτησης της οποίας ο μοναδικός αριθμός είναι το 1.
2. **Request Verbs**: Αυτά περιγράφουν τι θέλουμε να κάνουμε με τον πόρο. Ένα πρόγραμμα περιήγησης βασιζόμενο στο HTTP χρησιμοποιεί ένα ρήμα **GET** για να δώσει εντολή στο τελικό σημείο που θέλει να λάβει δεδομένα. Ωστόσο, υπάρχουν πολλά άλλα διαθέσιμα ρήματα, όπως **POST**, **PUT** και **DELETE**. Έτσι, στην περίπτωση του παραδείγματος **http://khakibee.com/booking/1**, το πρόγραμμα περιήγησης ιστού χρησιμοποιεί το ρήμα GET επειδή θέλει να λάβει τις λεπτομέρειες του αρχείου κρατήσεων.
3. **Request Headers**: Αυτές είναι πρόσθετες οδηγίες που αποστέλλονται με το αίτημα. Αυτά μπορεί να καθορίσουν τον τύπο της απάντησης που απαιτείται ή τις λεπτομέρειες εξουσιοδότησης.
4. **Request Body**: Τα δεδομένα αποστέλλονται με το αίτημα, τα δεδομένα αυτά συνήθως αποστέλλονται στο αίτημα όταν υποβάλλεται ερώτημα POST στις υπηρεσίες web REST. Σε μια κλήση POST, ο πελάτης λέει στις υπηρεσίες web REST ότι θέλει να προσθέσει έναν πόρο στον διακομιστή. Ως εκ τούτου, το σώμα αιτήματος θα έχει τις λεπτομέρειες του πόρου που απαιτείται να προστεθεί στον διακομιστή.
5. **Response Body**: Αυτό είναι το κύριο σώμα της απάντησης. Έτσι, στο παράδειγμά μας RESTful API, εάν υποβάλλαμε ερώτημα στον διακομιστή ιστού μέσω του αιτήματος **http://khakibee.com/booking/1**, ο διακομιστής ιστού μπορεί να επιστρέψει ένα έγγραφο JSON/XML με όλα τα στοιχεία της κράτησης στο σώμα απάντησης.

6. **Response Status codes**: Αυτοί οι κωδικοί είναι οι γενικοί κωδικοί που επιστρέφονται μαζί με την απάντηση από τον διακομιστή ιστού. Ένα παράδειγμα είναι ο κωδικός **200** που συνήθως επιστρέφεται εάν δεν υπάρχει σφάλμα κατά την επιστροφή μιας απάντησης στον πελάτη.

Κλήση RESTful πόρων μέσω HTTP

Μέσω τον ρημάτων (verbs) (POST, GET, PUT, DELETE) γίνεται η επίκληση των μεθόδων. Πιο συγκεκριμένα όταν ένας χρήστης κάνει ένα αίτημα σε μία υπηρεσία διαδικτύου χρησιμοποιεί ένα από τα HTTP ρήματα για να περιγράψει την ενέργεια που θέλει να πραγματοποιηθεί.



Εικόνα 3.8 Απεικόνιση RESTful μεθόδων

Επιλογή του RESTful

Επιτρέπει σε εφαρμογές Ιστού που είναι χτισμένες σε διάφορες γλώσσες προγραμματισμού και να επικοινωνούν μεταξύ τους με τη βοήθεια των υπηρεσιών Restful, αυτές οι εφαρμογές Ιστού μπορούν να βρίσκονται σε διαφορετικά περιβάλλοντα, ορισμένες θα μπορούσαν να είναι σε Windows και άλλες σε Linux. Αλλά τελικά, ανεξάρτητα από το περιβάλλον, το τελικό αποτέλεσμα θα πρέπει να είναι πάντα το ίδιο, δηλαδή θα πρέπει να μπορούν να συνομιλούν μεταξύ τους. Οι RESTful διαδικτυακές υπηρεσίες προσφέρουν αυτήν την ευελιξία σε εφαρμογές που είναι κατασκευασμένες σε διάφορες γλώσσες προγραμματισμού και πλατφόρμες. Εάν μια εφαρμογή πελάτη θέλει να συνεργαστεί με ιστότοπους όπως το Facebook ή η Google, θα πρέπει πιθανόν να γνωρίζει ποιά είναι η γλώσσα στην οποία είναι χτισμένες οι πλατφόρμες αυτές, καθώς και σε ποια πλατφόρμα είναι κατασκευασμένα. Με

βάση αυτό, μπορούμε να γράψουμε τον κώδικα διεπαφής για την web εφαρμογή μας, αλλά αυτό θα μπορούσε να αποδειχθεί εφιάλτης. Το Facebook και η Google εκθέτουν τη λειτουργικότητά τους με τη μορφή υπηρεσιών web Restful. Αυτό επιτρέπει σε οποιαδήποτε εφαρμογή πελάτη να καλεί αυτές τις υπηρεσίες web μέσω REST. Ένα ακόμη προτέρημα του είναι ότι στις μέρες μας όλα πρέπει να λειτουργούν και σε φορητές συσκευές. Οι εφαρμογές μετακινούνται σιγά σιγά σε συστήματα που βασίζονται σε cloud, όπως στο Azure ή στο Amazon. Το Azure και το Amazon παρέχουν πολλά APIs που βασίζονται στην αρχιτεκτονική Restful. Ως εκ τούτου, οι εφαρμογές πρέπει να αναπτυχθούν με τέτοιο τρόπο ώστε να γίνονται συμβατές με το Cloud. Επειδή λοιπόν όλες οι αρχιτεκτονικές που βασίζονται στο Cloud λειτουργούν με την αρχή REST, είναι πιο λογικό οι υπηρεσίες web να προγραμματίζονται στην αρχιτεκτονική που βασίζεται σε υπηρεσίες REST για να κάνουν την καλύτερη χρήση των υπηρεσιών που βασίζονται στο Cloud.

RESTful αρχιτεκτονική

Η κατάσταση και η λειτουργικότητα χωρίζονται σε κατανεμημένους πόρους. Αυτό σημαίνει ότι κάθε πόρος πρέπει να είναι προσβάσιμος μέσω των κανονικών εντολών HTTP των GET, POST, PUT ή DELETE. Έτσι, εάν κάποιος ήθελε να πάρει ένα αρχείο από έναν διακομιστή, θα πρέπει να μπορεί να εκδώσει το αίτημα GET και να λάβει το αρχείο. Εάν θέλουν να τοποθετήσουν ένα αρχείο στον διακομιστή, θα πρέπει να μπορούν να εκδώσουν το αίτημα POST ή PUT. Και τέλος, αν ήθελαν να διαγράψουν ένα αρχείο από τον διακομιστή, μπορούν να εκδώσουν το αίτημα DELETE. [20]

3.6 HTML

Η HTML ή αλλιώς γλώσσα σήμανσης υπερκειμένου είναι μία τυπική γλώσσα σήμανσης (περιγραφική) για έγγραφα που έχουν σχεδιαστεί για να εμφανίζονται σε ένα πρόγραμμα περιήγησης στο διαδίκτυο δηλαδή δημιουργεί και διαρθρώνει περιεχόμενο στον Παγκόσμιο Ιστό. Μπορεί να υποστηρίζεται από τεχνολογίες όπως η CSS (Cascading Style Sheets) και γλώσσες προγραμματισμού όπως η Javascript. Ο φυλλομετρητής ιστού λαμβάνει έγγραφα HTML από έναν διακομιστή ιστού ή από τοπική αποθήκευση και αποδίδει πίσω τα έγγραφα σαν ιστοσελίδες πολυμέσων. Επί της ουσίας η HTML περιγράφει την δομή μιας ιστοσελίδας σημασιολογικά και επίσης περιλαμβάνει ενδείξεις για την εμφάνιση του εγγράφου. Η HTML παρέχει ένα μέσο για τη δημιουργία δομημένων εγγράφων υποδηλώνοντας σημασιολογία στο κείμενο όπως επικεφαλίδες, παραγράφους, λίστες, συνδέσμους, εισαγωγικά και άλλα

στοιχεία. Τα στοιχεία της HTML οριοθετούνται από ετικέτες, οι οποίες γράφονται με τη χρήση γωνιακών παρενθέσεων τα λεγόμενα **tags**. Ετικέτες όπως `<img />` και `<input />` εισάγουν άμεσα περιεχόμενο στη σελίδα. Άλλες ετικέτες όπως `<p>` περιβάλλουν και παρέχουν πληροφορίες σχετικά με το κείμενο του εγγράφου και μπορεί να περιλαμβάνουν άλλες ετικέτες ως υποστοιχεία. Τα προγράμματα περιήγησης δεν εμφανίζουν τις ετικέτες HTML αλλά τις χρησιμοποιούν για να ερμηνεύσουν το περιεχόμενο της σελίδας. Ένα παράδειγμα είναι η HTML5, που χρησιμοποιείται για την προβολή βίντεο και ήχου, κυρίως με τη χρήση του στοιχείου `<canvas>`, σε συνεργασία με τη Javascript. Η HTML δεν είναι μία κλασική γλώσσα προγραμματισμού και επί της ουσίας δεν αποτελεί γλώσσα προγραμματισμού καθώς δεν διαθέτει τα απαραίτητα χαρακτηριστικά ώστε να χαρακτηριστεί γλώσσα προγραμματισμού. Πιό συγκεκριμένα δεν επιτρέπει την αλλαγή ροής εκτέλεσης βάση συνθηκών και δεν περιλαμβάνει αναθέσεις. Είναι μία σχετικά απλή δομή και μπορεί να γραφτεί σε όλους τους επεξεργαστές κειμένων. Ό σκελετός που έχει κάθε HTML αρχείο είναι ο εξής:

```
<html lang="en">
  <head>    <meta charset="UTF-8" />
    <title>HTML</title>
  </head>
  <body> </body>
</html>
```

Ας κάνουμε μια ιστορική αναδρομή στην HTML (Hypertext Markup Language), αναπτύχθηκε στα τέλη της δεκαετίας του 1980 από τον Tim Berners-Lee, έναν Βρετανό επιστήμονα πληροφορικής, ο οποίος εφηύρε τον Παγκόσμιο Ιστό ενώ εργαζόταν στο CERN (Ευρωπαϊκός Οργανισμός Πυρηνικών Ερευνών). Η HTML είναι μια γλώσσα σήμανσης, πράγμα που σημαίνει ότι χρησιμοποιείται για να σχολιάσει κείμενο και άλλο περιεχόμενο με ετικέτες που καθορίζουν τον τρόπο με τον οποίο το περιεχόμενο θα πρέπει να εμφανίζεται και να μορφοποιείται. Η πρώτη έκδοση της HTML, γνωστή ως HTML 1.0, κυκλοφόρησε το 1993. Ήταν μια πολύ βασική γλώσσα, με λίγες μόνο ετικέτες για τη μορφοποίηση του κειμένου και την προσθήκη εικόνων στις ιστοσελίδες. Με την πάροδο των ετών, η HTML έχει εξελιχθεί και έχουν κυκλοφορήσει νέες εκδόσεις. Το 2014, το W3C (World Wide Web Consortium) κυκλοφόρησε την HTML5, η οποία είναι η πιο πρόσφατη έκδοση της HTML. Η HTML5 εισήγαγε μια σειρά νέων χαρακτηριστικών, συμπεριλαμβανομένης της δυνατότητας αναπαραγωγής αρχείων ήχου και βίντεο εγγενώς μέσα σε ιστοσελίδες, καθώς και της δυνατότητας δημιουργίας διαδραστικών γραφικών και κινούμενων σχεδίων. Η HTML

αποτελεί σημαντικό θεμέλιο του Παγκόσμιου Ιστού και συνεχίζει να εξελίσσεται και να βελτιώνεται καθώς προκύπτουν νέες τεχνολογίες και ανάγκες.

Πλεονεκτήματα της χρήσης της HTML:

- Είναι ένα ευρέως χρησιμοποιούμενο και καθιερωμένο πρότυπο, οπότε υπάρχει μια μεγάλη κοινότητα προγραμματιστών και πληθώρα πόρων.
- Η ανεκτικότητα της σε λάθη συγκριτικά με άλλες γλώσσες προγραμματισμού δίνουν ευελιξία και δεν έχουν σαν αποτέλεσμα την κατάρρευση της εφαρμογής.
- Είναι εύκολη στην εκμάθηση και τη χρήση, ακόμη και για αρχάριους.
- Υποστηρίζεται από όλα τα σύγχρονα προγράμματα περιήγησης ιστού, οπότε θα λειτουργεί με συνέπεια σε διαφορετικές πλατφόρμες και συσκευές.
- Διατίθεται δωρεάν και δεν απαιτείται η εγκατάσταση συγκεκριμένου λογισμικού καθώς μπορεί να συνταχθεί σε οποιονδήποτε κειμενογράφο και να εμφανιστεί σε οποιονδήποτε περιηγητή.

Μειονεκτήματα της χρήσης της HTML:

- Δεν είναι γλώσσα προγραμματισμού, επομένως δεν έχει τις πλήρεις δυνατότητες μιας γλώσσας προγραμματισμού όπως η JavaScript, ώς εκ τούτου δεν είναι τόσο αποτελεσματική για την κατασκευή σύνθετων και διαδραστικών εφαρμογών ιστού.
- Παρέχει προκαθορισμένα tags που παρόλες τις συνεχόμενες βελτιώσεις δεν καλύπτει την ανάγκη του χρήστη για δικά του tags.
- Η απλότητα και στατικότητα του μπορεί να δημιουργεί κενά ασφαλείας.
- Μπορεί να είναι δύσκολη η συντήρηση και η ενημέρωση μεγάλων βάσεων κώδικα HTML, καθώς δεν υπάρχει ενσωματωμένη υποστήριξη για πράγματα όπως η αρθρωματοποίηση και ο διαχωρισμός των προβλημάτων.

Συνολικά, η HTML είναι ένα πολύτιμο εργαλείο για την κατασκευή της δομής και του περιεχομένου μιας ιστοσελίδας και χρησιμοποιείται συχνά σε συνδυασμό με άλλες τεχνολογίες όπως η CSS και η JavaScript για τη δημιουργία πιο δυναμικών και διαδραστικών ιστοσελίδων. Έχοντας την ώς βάση θα σχεδιαστεί και δομηθεί και η δική μας διπλωματική τόσο στο κομμάτι της διαχειριστικής σελίδας αλλά και στο widget που θα απευθύνεται στους

πελάτες. Για την επίτευξη βέλτιστου αποτελέσματος θα γίνει συνδυασμός τεχνολογιών με χρήση της CSS καθώς και JavaScript, όπως θα αναλυθεί παρακάτω. [21]

3.7 CSS

Η **CSS** (Cascading Style Sheets) είναι μια γλώσσα που χρησιμοποιείται για να περιγράψει την εμφάνιση και τη μορφοποίηση ενός εγγράφου γραμμένο σε HTML (HyperText Markup Language). Χρησιμοποιείται για την παροχή ενός συνεπούς και οργανωμένου τρόπου διαμόρφωσης ιστοσελίδων. Η CSS αποτελεί το βασικό μέρος του σύγχρονου σχεδιασμού ιστοσελίδων και χρησιμοποιείται για τον έλεγχο της διάταξης, της γραμματοσειράς και τών χρωματισμών των ιστοσελίδων. Το όνομα cascading προέρχεται από το καθορισμένο σχήμα προτεραιότητας για τον προσδιορισμό του κανόνα στυλ που εφαρμόζεται εάν περισσότεροι από ένας κανόνες ταιριάζουν σε ένα συγκεκριμένο στοιχείο. Ο διαχωρισμός της μορφοποίησης και του περιεχομένου καθιστά επίσης εφικτή την παρουσίαση της ίδιας σελίδας σήμανσης με διαφορετικά στυλ για διαφορετικές μεθόδους απόδοσης, όπως στην οθόνη, στην εκτύπωση, με φωνή (μέσω προγράμματος περιήγησης που βασίζεται σε ομιλία ή αναγνώστη οθόνης) και σε συσκευές αφής που βασίζονται στη γραφή Braille. Η CSS διαθέτει επίσης κανόνες για εναλλακτική μορφοποίηση αν το περιεχόμενο προσπελαύνεται από κινητή συσκευή.

Η CSS έχει εξελιχθεί με την πάροδο του χρόνου, με νέες εκδόσεις να κυκλοφορούν περιοδικά για την προσθήκη νέων χαρακτηριστικών και τη βελτίωση της συμβατότητας. Η πιο πρόσφατη έκδοση των CSS είναι η CSS3, η οποία κυκλοφόρησε το 1999 και έκτοτε έχει ενημερωθεί αρκετές φορές. Για να χρησιμοποιήσουμε την CSS, μπορούμε να τα συμπεριλάβουμε στο αρχείο HTML, παραπέμποντας σε ένα εξωτερικό αρχείο στυλ ή ενσωματώνοντας τα απευθείας στο αρχείο HTML χρησιμοποιώντας ένα στοιχείο `<style>`. Ακολουθεί ένα παράδειγμα για το πώς μπορούμε να συνδέσουμε ένα εξωτερικό CSS αρχείο:

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

Ενσωματωση CSS σε HTML με τη χρήση ενός στοιχείου `<style>`:

```
<head>
  <style>
    h1 {
      color: red;
```

```
      font-size: 24px;
    }
  </style>
</head>
```

Πλεονεκτήματα της χρήσης CSS περιλαμβάνουν:

- Διαχωρισμός περιεχομένου και παρουσίασης
- Βελτιωμένη προσβασιμότητα
- Επαναχρησιμοποιήσιμα στυλ

Μειονεκτήματα της χρήσης CSS περιλαμβάνουν:

- Πολυπλοκότητα
- Ζητήματα συμβατότητας με προγράμματα περιήγησης
- Ζητήματα επιδόσεων

[22] [23]

3.7.1 Bootstrap (CSS Framework)

Η Bootstrap είναι ένα δωρεάν, ανοικτού κώδικα πλαίσιο ανάπτυξης ιστοσελίδων frontend που παρέχει ένα σύνολο εργαλείων για την κατασκευή mobile-first ιστοσελίδων και εφαρμογών ιστού. Βασίζεται σε HTML, CSS και JavaScript και δημιουργήθηκε από το Twitter. Η Bootstrap έχει σχεδιαστεί για να βοηθά τους προγραμματιστές να δημιουργούν γρήγορα και εύκολα οπτικά ελκυστικούς ιστότοπους που είναι συμβατοί με ένα ευρύ φάσμα σύγχρονων συσκευών και προγραμμάτων περιήγησης. Περιλαμβάνει ένα σύστημα πλέγματος, προ-σχεδιασμένα στοιχεία UI, όπως κουμπιά, φόρμες και μπάρες πλοήγησης, καθώς και διάφορα στυλ για την τυπογραφία, τους πίνακες και άλλα κοινά στοιχεία σχεδιασμού ιστού. Η Bootstrap χρησιμοποιείται ευρέως από προγραμματιστές ιστού επειδή είναι εύκολη στην εκμάθηση και τη χρήση και παρέχει συνεπή, επαγγελματική σχεδίαση για ιστότοπους. Είναι επίσης εξαιρετικά προσαρμόσιμη, επιτρέποντας στους προγραμματιστές να παρακάμπτουν τα προεπιλεγμένα στυλ και να δημιουργούν τα δικά τους σχέδια χρησιμοποιώντας τις παρεχόμενες κλάσεις και στυλ. Για να χρησιμοποιήσουμε τη Bootstrap σε μία εφαρμογή μπορούμε είτε να συνδεθούμε με το CDN (Content Delivery Network) της Bootstrap είτε να κατεβάσουμε τα πηγαία αρχεία της Bootstrap και να τα συμπεριλάβουμε στην εφαρμογή σας.

Βασικά χαρακτηριστικά της Bootstrap

- Η Bootstrap έχει σχεδιαστεί για να είναι mobile-first, δηλαδή δίνει προτεραιότητα στο σχεδιασμό και τη διάταξη ενός ιστότοπου για μικρά μεγέθη οθόνης και στη συνέχεια κλιμακώνεται σε μεγαλύτερα μεγέθη οθόνης. Αυτό συμβάλλει στη διασφάλιση της καλής εμφάνισης και της καλής λειτουργίας του ιστότοπου σε ένα ευρύ φάσμα συσκευών, συμπεριλαμβανομένων των smartphones, των tablets και των επιτραπέζιων υπολογιστών.
- Η Bootstrap περιλαμβάνει ένα σύστημα πλέγματος που επιτρέπει στους προγραμματιστές να δημιουργήσουν μια ανταποκρινόμενη (responsive) διάταξη για τον ιστότοπό τους. Το σύστημα πλέγματος αποτελείται από γραμμές και στήλες και τα στοιχεία μπορούν να τοποθετηθούν σε συγκεκριμένες στήλες για τη δημιουργία της επιθυμητής διάταξης. Το πλέγμα έχει σχεδιαστεί για να προσαρμόζει αυτόματα τη διάταξη του ιστότοπου με βάση το μέγεθος της οθόνης, ώστε ο ιστότοπος να φαίνεται σωστά σε διαφορετικές συσκευές και μεγέθη οθόνης.
- Η Bootstrap είναι εξαιρετικά προσαρμόσιμη, επιτρέποντας στους προγραμματιστές να παρακάμπτουν τα προεπιλεγμένα στυλ και να δημιουργούν τα δικά τους σχέδια χρησιμοποιώντας τις παρεχόμενες κλάσεις και στυλ.
- Η Bootstrap ενημερώνεται τακτικά με νέα χαρακτηριστικά και βελτιώσεις. Η πιο πρόσφατη έκδοση της Bootstrap είναι η έκδοση 5, η οποία κυκλοφόρησε τον Μάιο του 2021.

Επιλογή της Bootstrap

Η Bootstrap μπορεί να χρησιμοποιηθεί σε React JS εφαρμογές για τη γρήγορη κατασκευή εφαρμογών που ανταποκρίνονται στις ανάγκες κινητών συσκευών, τάμπλετ και σταθερών υπολογιστών. Παρέχει ένα σύνολο προ-σχεδιασμένων στοιχείων και στυλ UI που μπορούν εύκολα να προστεθούν σε ένα έργο React χρησιμοποιώντας τις κλάσεις της Bootstrap. Οι ανάγκες της εφαρμογής μας για επαγγελματική εμφάνιση αλλά και για ταχεία ανάπτυξη του MVP, μας οδήγησε σε αυτή την επιλογή. Ωστόσο, επειδή η εφαρμογή μας είχε ανάγκη για σχεδιαστικές ιδιαιτερότητες σχεδιάστηκαν και γράφτηκαν κανόνες καθαρής CSS ώστε να έχουμε το επιθυμητό αποτέλεσμα. [24]

3.8 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται συνήθως για τη δημιουργία διαδραστικών εφέ μέσα σε προγράμματα περιήγησης στο διαδίκτυο. Είναι μια

αντικειμενοστραφής γλώσσα που χρησιμοποιείται για τη δημιουργία δυναμικών και διαδραστικών ιστότοπων και διαδικτυακών εφαρμογών. Η JavaScript χρησιμοποιείται συχνά σε συνδυασμό με HTML και CSS για τη δημιουργία ιστότοπων και εφαρμογών. Η HTML χρησιμοποιείται για τη δόμηση του περιεχομένου ενός ιστότοπου, η CSS χρησιμοποιείται για τη διαμόρφωση του ιστότοπου και η JavaScript χρησιμοποιείται για την προσθήκη διαδραστικότητας και δυναμικής συμπεριφοράς. Η JavaScript μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός ευρέος φάσματος διαδραστικών εφέ, όπως:

- Κίνηση στοιχείων σε μια ιστοσελίδα.
- Επικύρωση εισόδου φόρμας.
- Δημιουργία διαδραστικών χαρτών και διαγραμμάτων.
- Προσθήκη συσκευών αναπαραγωγής ήχου και βίντεο σε μια ιστοσελίδα.
- Δημιουργία παιχνιδιών και άλλων διαδραστικών εφαρμογών.

Η JavaScript είναι μια υψηλού επιπέδου, δυναμικά τυποποιημένη γλώσσα προγραμματισμού που χρησιμοποιείται κυρίως στην ανάπτυξη ιστοσελίδων και εφαρμογών. Η JavaScript εκτελείται σε ένα πρόγραμμα περιήγησης ιστού και υποστηρίζεται από όλα τα σύγχρονα προγράμματα περιήγησης. Χρησιμοποιείται επίσης συνήθως στην πλευρά του διακομιστή μέσω της χρήσης περιβαλλόντων εκτέλεσης όπως η NodeJS. Υπάρχουν πολλά εργαλεία και πλαίσια διαθέσιμα για να βοηθήσουν τους προγραμματιστές να δημιουργήσουν εφαρμογές με JavaScript, συμπεριλαμβανομένων βιβλιοθηκών, πλαισίων και εργαλείων ανάπτυξης. Ορισμένα δημοφιλή εργαλεία και πλαίσια περιλαμβάνουν τα jQuery, React, Angular και Vue. Συνολικά, η JavaScript είναι μια ισχυρή και ευέλικτη γλώσσα που χρησιμοποιείται ευρέως στην ανάπτυξη ιστοσελίδων και εφαρμογών. Εξελίσσεται και βελτιώνεται συνεχώς και αποτελεί απαραίτητο εργαλείο για κάθε προγραμματιστή.

3.8.1 Διαδεδομένα Frameworks

Υπάρχουν πολλά δημοφιλή frameworks για την ανάπτυξη ιστοσελίδων και το ποιο είναι το "πιο δημοφιλές" μπορεί να ποικίλλει ανάλογα με τα κριτήρια που χρησιμοποιεί το συγκεκριμένο πλαίσιο στο οποίο μετράει τη δημοτικότητα τους. Ακολουθούν τα πιο δημοφιλή πλαίσια (frameworks):

- **Angular**: Η Angular είναι ένα ολοκληρωμένο πλαίσιο για την κατασκευή εφαρμογών μιας σελίδας (SPAs). Χρησιμοποιεί μια δηλωτική προσέγγιση για την κατασκευή UI και περιλαμβάνει ένα ισχυρό σύνολο εργαλείων για την κατασκευή σύνθετων, διαδραστικών εφαρμογών. Η Angular έχει σχεδιαστεί για να είναι ιδιαίτερα αρθρωτή

και χρησιμοποιεί μια αρχιτεκτονική βασισμένη σε συστατικά για να επιτρέπει στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα στοιχεία UI που μπορούν εύκολα να μοιραστούν και να επαναχρησιμοποιηθούν σε διάφορα μέρη μιας εφαρμογής.

- **React**: Η React είναι μια δημοφιλής βιβλιοθήκη για την κατασκευή διεπαφών χρήστη. Χρησιμοποιεί μια δηλωτική, βασισμένη σε στοιχεία προσέγγιση για την κατασκευή UI και έχει σχεδιαστεί για να είναι εύκολη στη χρήση και εξαιρετικά αποδοτική. Η React χρησιμοποιείται συχνά για την κατασκευή σύνθετων, διαδραστικών UI και χρησιμοποιείται ευρέως στην ανάπτυξη εφαρμογών μιας σελίδας.
- **Vue**: Η Vue.js είναι ένα προοδευτικά αναπτυσσόμενο πλαίσιο για την κατασκευή διεπαφών χρήστη. Έχει σχεδιαστεί για να είναι εύκολη στη χρήση και στην εκμάθηση της και επικεντρώνεται στην παροχή ενός απλού, διαισθητικού API για τη δημιουργία διαδραστικών UIs. Η Vue είναι ιδιαίτερα αρθρωτή και χρησιμοποιεί μια σύνταξη βασισμένη σε πρότυπα που καθιστά εύκολη την κατασκευή επαναχρησιμοποιήσιμων στοιχείων UI.
- **jQuery**: Η jQuery είναι μια βιβλιοθήκη JavaScript που έχει σχεδιαστεί για να διευκολύνει την εργασία με HTML, CSS και JavaScript. Είναι δημοφιλής επειδή παρέχει ένα ευρύ φάσμα χρήσιμων λειτουργιών και χαρακτηριστικών που μπορούν εύκολα να ενσωματωθούν σε έργα ιστού.

3.8.2 Επιλογή της React

Η React είναι μια βιβλιοθήκη JavaScript για την κατασκευή διεπαφών χρήστη, αναπτύχθηκε από το Facebook και χρησιμοποιείται συχνά για την κατασκευή εφαρμογών μιας σελίδας και εφαρμογών για κινητά τηλέφωνα. Η React μας επιτρέπει να δημιουργούμε επαναχρησιμοποιήσιμα στοιχεία UI. Χρησιμοποιεί ένα εικονικό DOM (μια ελαφριά αναπαράσταση στη μνήμη του πραγματικού DOM) για τη βελτίωση της απόδοσης μειώνοντας τον αριθμό των πράξεων χειρισμού του DOM που απαιτούνται για την ενημέρωση του UI. Η React ακολουθεί ένα δηλωτικό στυλ προγραμματισμού, το οποίο σημαίνει ότι περιγράφουμε την επιθυμητή κατάσταση του UI και η React αναλαμβάνει την ενημέρωση του πραγματικού DOM ώστε να ταιριάζει με αυτή την κατάσταση, αυτό διευκολύνει τη συλλογιστική του κώδικα και τη συγγραφή συντηρήσιμων εφαρμογών. Η React χρησιμοποιεί μια επέκταση σύνταξης που ονομάζεται JSX, η οποία μας επιτρέπει να γράφουμε κώδικα που μοιάζει με HTML στα αρχεία JavaScript. Αυτό μπορεί να κάνει τον κώδικα πιο συνοπτικό και πιο ευανάγνωστο, ειδικά όταν συνδυάζεται με την ικανότητα του JSX να ενσωματώνει HTML σε τεθλασμένες αγκύλες (tags).

Τα βασικά χαρακτηριστικά που μας παρέχει η React είναι τα εξής

- **Εικονικό DOM**: Η React χρησιμοποιεί ένα εικονικό DOM (Document Object Model) για τη βελτιστοποίηση των ενημερώσεων στο UI. Όταν η κατάσταση ενός στοιχείου αλλάζει, η React συγκρίνει τη νέα έκδοση του στοιχείου με την προηγούμενη έκδοση και ενημερώνει μόνο τα τμήματα του DOM που έχουν αλλάξει. Αυτό μπορεί να κάνει τις εφαρμογές React πιο αποδοτικές και αποτελεσματικές, ειδικά για εφαρμογές με μεγάλο όγκο δεδομένων ή συχνές ενημερώσεις.
- **Δηλωτική σύνταξη**: Η React χρησιμοποιεί ένα δηλωτικό συντακτικό, το οποίο σημαίνει ότι οι προγραμματιστές περιγράφουν πώς θα πρέπει να μοιάζει το UI, αντί να καθορίζουν ρητά τον τρόπο ενημέρωσης του UI. Αυτό μπορεί να διευκολύνει την κατανόηση και την αποσφαλμάτωση των εφαρμογών και μπορεί επίσης να διευκολύνει τη δημιουργία επαναχρησιμοποιήσιμων στοιχείων UI.
- **Δημοτικότητα και υποστήριξη από την κοινότητα**: Η React είναι μια ευρέως χρησιμοποιούμενη και δημοφιλής βιβλιοθήκη και διαθέτει μια μεγάλη και ενεργή κοινότητα προγραμματιστών. Αυτό σημαίνει ότι υπάρχουν πολλοί διαθέσιμοι πόροι και εργαλεία για να βοηθήσουν τους προγραμματιστές να δημιουργήσουν εφαρμογές με την React και είναι εύκολο να βρουν βοήθεια και υποστήριξη όταν χρειάζεται.
- **Επαναχρησιμοποιήσιμα στοιχεία**: Όπως αναφέρθηκε προηγουμένως, ένα από τα κύρια πλεονεκτήματα της React είναι η αρχιτεκτονική της που βασίζεται σε συστατικά, η οποία επιτρέπει στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα συστατικά UI που μπορούν εύκολα να μοιραστούν και να επαναχρησιμοποιηθούν σε διάφορα μέρη μιας εφαρμογής. Αυτό μπορεί να είναι ιδιαίτερα χρήσιμο για την κατασκευή μιας σελίδας διαχείρισης, καθώς μπορεί να απαιτείται μεγάλος αριθμός διαφορετικών στοιχείων UI, όπως φόρμες, πίνακες και κουμπιά. Χτίζοντας αυτά τα στοιχεία ως επαναχρησιμοποιήσιμα στοιχεία, μπορεί να είναι ευκολότερη η συντήρηση και η ενημέρωση της σελίδας με την πάροδο του χρόνου.
- **Απεικόνιση από την πλευρά του διακομιστή**: Η React μπορεί να χρησιμοποιηθεί για την απόδοση σελίδων στην πλευρά του διακομιστή, γεγονός που μπορεί να είναι επωφελές για το SEO και την απόδοση. Όταν μια σελίδα διαχείρισης αποδίδεται στον διακομιστή, μπορεί να ευρετηριαστεί εύκολα από τις μηχανές αναζήτησης και μπορεί επίσης να φορτωθεί γρηγορότερα για τους χρήστες.
- **Διαχείριση δεδομένων**: Η React περιλαμβάνει ένα σύνολο εργαλείων για τη διαχείριση της ροής δεδομένων μέσα σε μια εφαρμογή, συμπεριλαμβανομένης της χρήσης κατάστασης. Αυτό μπορεί να είναι ιδιαίτερα χρήσιμο για την κατασκευή μιας σελίδας διαχείρισης, καθώς μπορεί να περιλαμβάνει την εμφάνιση και την ενημέρωση

μεγάλου όγκου δεδομένων. Με τη χρήση των εργαλείων διαχείρισης δεδομένων της React, μπορεί να είναι ευκολότερο να διατηρείται το UI σε συγχρονισμό με τα υπόλοιπα δεδομένα.

Συνολικά, η React είναι μια ισχυρή και δημοφιλής βιβλιοθήκη που είναι κατάλληλη για την κατασκευή σύνθετων, διαδραστικών διεπαφών χρήστη, συμπεριλαμβανομένων και σελίδων διαχείρισης. Η αρχιτεκτονική του που βασίζεται σε συστατικά, οι δυνατότητες απόδοσης από την πλευρά του διακομιστή και τα εργαλεία διαχείρισης δεδομένων το καθιστούν μια ισχυρή επιλογή για αυτόν τον τύπο εφαρμογών. [25] [26]

3.8.3 Jest και React Testing-library

Το Jest είναι ένα πλαίσιο δοκιμών JavaScript που χρησιμοποιείται ευρέως για τον έλεγχο εφαρμογών JavaScript. Είναι γρήγορο, εύκολο στη χρήση και διαθέτει ένα μεγάλο οικοσύστημα από πρόσθετα και επεκτάσεις που μπορούν να χρησιμοποιηθούν για την επέκταση της λειτουργικότητας του. Η βιβλιοθήκη Testing-library είναι μια βιβλιοθήκη βοηθητικών συναρτήσεων που μας βοηθά να γράφουμε δοκιμές που εστιάζουν στη συμπεριφορά της εφαρμογής και όχι στις λεπτομέρειες υλοποίησής της. Ενθαρρύνει τη συγγραφή δοκιμών που είναι εύκολα αναγνώσιμες και κατανοητές και που είναι λιγότερο εύθραυστες και περισσότερο συντηρήσιμες με την πάροδο του χρόνου. Μαζί, το Jest και το Testing-library μπορούν να χρησιμοποιηθούν για τη δημιουργία μιας ισχυρής και αποτελεσματικής εγκατάστασης δοκιμών για εφαρμογές JavaScript. Το Jest παρέχει την υποδομή δοκιμών και το περιβάλλον εκτέλεσης, ενώ το Testing-library παρέχει τις βοηθητικές λειτουργίες και την καθοδήγηση για τη συγγραφή αποτελεσματικών δοκιμών. [27] [28]

Πλεονεκτήματα που μας προσφέρουν:

- Ταχύτητα
- Ευκολία στη χρήση
- Μεγάλο οικοσύστημα
- Χρησιμοποιούνται ευρέως

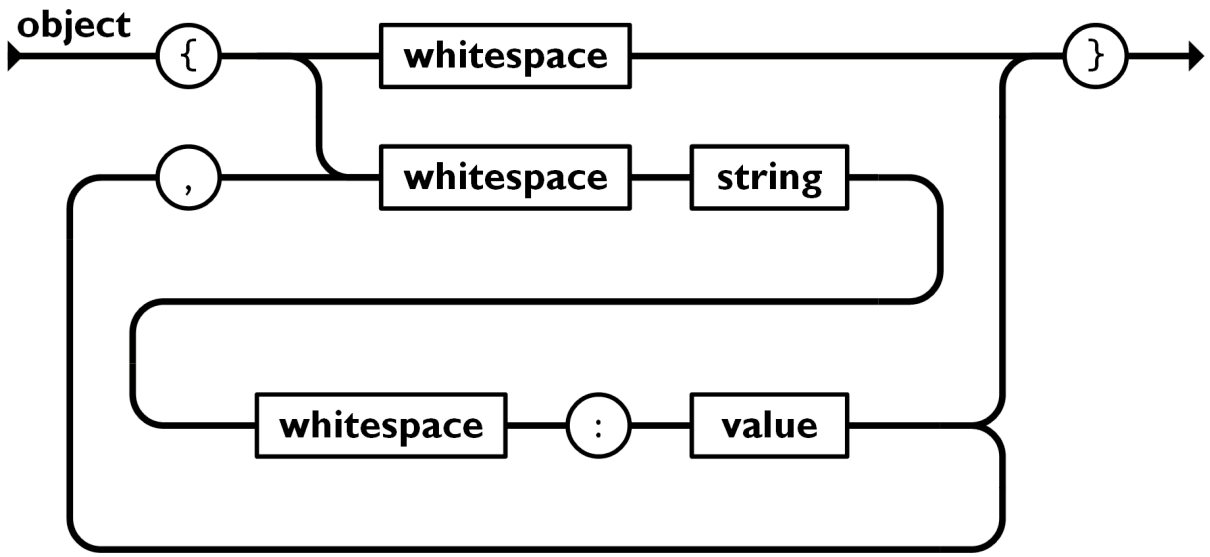
Μειονεκτήματα των παραπάνω εργαλείων:

- Περιορίζεται σε JavaScript
- Μπορεί να απαιτούν ρυθμίσεις

3.9 JSON

Το JSON είναι μία μορφή δεδομένων που σημαίνει JavaScript Object Notation, πρόκειται για μια ελαφριά μορφή ανταλλαγής δεδομένων που είναι ευανάγνωστο και εύκολα επεξεργάσιμο για τους ανθρώπους αλλά κυρίως για τις υπολογιστικές μηχανές να αναλύουν και να παράγουν τέτοιας μορφής δεδομένα. Βασίζεται σε ένα υποσύνολο του προτύπου γλώσσας προγραμματισμού JavaScript ECMA-262 3rd Edition, Δεκέμβριος 1999. Το JSON είναι μια μορφή κειμένου που είναι πλήρως ανεξάρτητη από τη γλώσσα προγραμματισμού, αλλά χρησιμοποιεί συμβάσεις που είναι γνωστές από την οικογένεια γλωσσών C, συμπεριλαμβανομένων των C++, C#, Java, JavaScript, Python, κλπ και αυτές οι ιδιότητες καθιστούν το JSON μια ιδανική γλώσσα ανταλλαγής δεδομένων. Το JSON αποτελείται από ζεύγη κλειδιών-τιμών, τα κλειδιά είναι συμβολοσειρές και οι τιμές μπορεί να είναι συμβολοσειρές, αριθμοί, booleans, πίνακες ή άλλα αντικείμενα JSON. Το JSON χρησιμοποιείται συχνά για τη ανταλλαγή δεδομένων μεταξύ ενός διακομιστή και μιας διαδικτυακής εφαρμογής ή μεταξύ διαφορετικών APIs. Το JSON χρησιμοποιείται επίσης σε πολλά άλλα μέρη, όπως αρχεία ρυθμίσεων και βάσεις δεδομένων NoSQL.

Μία από τις πιό σύνηθες μορφές που παίρνει το JSON είναι όταν ένα αντικείμενο είναι ένα μη ταξινομημένο σύνολο ζευγών ονόματος/τιμής. Ένα αντικείμενο αρχίζει με **{** αριστερή αγκύλη και τελειώνει με **}** δεξιά αγκύλη. Κάθε όνομα ακολουθείται από **:** άνω και κάτω τελεία και τα ζεύγη ονόματος/τιμής χωρίζονται με **,** κόμμα. Μια συνηθισμένη επίσης μορφή έκφρασης των πινάκων με JSON είναι όταν ένας πίνακας είναι μια διατεταγμένη συλλογή τιμών. Ένας πίνακας αρχίζει με **[** αριστερή τετραγωνισμένη παρένθεση και τελειώνει με **]** δεξιά τετραγωνισμένη παρένθεση και οι τιμές χωρίζονται με **,** κόμμα. [28]



Εικόνα 3.9 Δομή αντικειμένου σε JSON
(https://www.json.org/img/object.png)

3.10 Golang

Η Go, γνωστή και ώς Golang είναι μια στατικά τυποποιημένη γλώσσα σύνταξης που είναι παρόμοια με την οικογένεια γλωσσών όπως η C και η C++ αλλά με πολλά πρόσθετα χαρακτηριστικά που ξεπερνούν τα προβλήματα και την πολυπλοκότητα των παραπάνω γλωσσών. Η Go είναι γνωστή για την ταυτόχρονη χρήση της (concurrency) και την αποδοτική χρήση της μνήμης και των πόρων, καθιστώντας την κατάλληλη για δικτυακά και κατανεμημένα συστήματα. Η Go είναι μια γλώσσα που μεταγλωττίζεται, αυτό σημαίνει ότι μεταφράζεται σε κώδικα μηχανής που μπορεί να εκτελεστεί απευθείας από τη CPU ενός υπολογιστή. Αυτό κάνει τα προγράμματα που είναι γραμμένα σε Go γρήγορα και αποδοτικά, αλλά σημαίνει επίσης ότι πρέπει να μεταγλωτίζονται εκ νέου για διαφορετικά λειτουργικά συστήματα και αρχιτεκτονικές. Ένα από τα βασικά χαρακτηριστικά της Go είναι η ενσωματωμένη υποστήριξη ταυτόχρονης εκτέλεσης, η οποία επιτρέπει την ταυτόχρονη εκτέλεση πολλαπλών διεργασιών. Η Go χρησιμοποιεί ένα μοντέλο ταυτόχρονης εκτέλεσης που ονομάζεται "goroutines", τα οποία είναι ελαφριά νήματα που εκτελούνται στον ίδιο χώρο διευθύνσεων με το κύριο πρόγραμμα. Η διαχείριση των goroutines γίνεται από το περιβάλλον εκτέλεσης της Go, γεγονός που καθιστά εύκολη τη συγγραφή κώδικα ταυτόχρονης εκτέλεσης χωρίς την ανάγκη άλλων πρωτοκόλλων συγχρονισμού. Η Go διαθέτει επίσης έναν ενσωματωμένο συλλέκτη σκουπιδιών, ο οποίος απελευθερώνει αυτόματα τη μνήμη που δεν χρειάζεται πλέον. Αυτό εξαλείφει την ανάγκη για χειροκίνητη διαχείριση της μνήμης, η οποία μπορεί να αποτελέσει πηγή σφαλμάτων και τρωτών σημείων ασφαλείας σε άλλες γλώσσες.

Η τυπική βιβλιοθήκη της Go παρέχει ένα ευρύ φάσμα λειτουργιών, συμπεριλαμβανομένης της υποστήριξης για δικτύωση, κρυπτογράφηση και συμπίεση. Παρέχει επίσης ισχυρά εργαλεία για τη δημιουργία εφαρμογών και υπηρεσιών ιστού, συμπεριλαμβανομένου του πακέτου net/http για τη δημιουργία διακομιστών και πελατών HTTP. Η Go είναι επίσης κατάλληλη για την κατασκευή κατανεμημένων συστημάτων μεγάλης κλίμακας, καθώς έχει μικρό αποτύπωμα και είναι εύκολο να αναπτυχθεί. Αυτό έχει οδηγήσει στην αυξανόμενη δημοτικότητά της στον κλάδο, ιδίως στον τομέα του Cloud, του DevOps και των Microservices.

Ιστορική αναδρομή στην δημιουργία της Golang

Η Go αναπτύχθηκε το 2007 από την Google και η βασική ιδέα ήταν να απλοποιηθεί η διαδικασία ανάπτυξης λογισμικού με μια γλώσσα σε σχέση με αυτές που χρησιμοποιούνταν εκείνη την εποχή. Οι έμπειροι προγραμματιστές Ken Thompson (εφευρέτης του UNIX και της C), Rob Pike (συγγραφέας του UTF 8 και του UNIX format) και Robert Griesemer ήταν οι άνθρωποι πίσω από την ανάπτυξη της Go. Η ευρέως χρησιμοποιούμενη C++ είναι μια εξαιρετικά πολύπλοκη γλώσσα και έχει ως αποτέλεσμα οι ομάδες ανάπτυξης να είναι λιγότερο παραγωγικές και να σπαταλούσαν περισσότερο χρόνο για τη διόρθωση σφαλμάτων. Η Go σχεδιάστηκε για να είναι απλή στη χρήση και ιδανική για τη δημιουργία σύνθετων συστημάτων προγραμματισμού. Αυτό επέτρεψε σε μεγάλες ομάδες ανάπτυξης λογισμικού να συνεργάζονται και να επιλύουν αποτελεσματικά πολύπλοκα έργα.

Η χρήση της Golang ενδείκνυται ιδιαίτερα για την ανάπτυξη

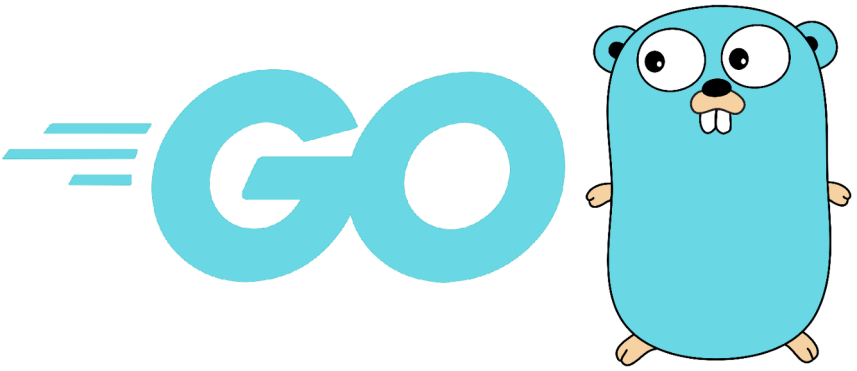
- Διαδικτυακών εφαρμογών.
- Υπηρεσίες διαδικτύου και νέφους.
- DevOps και αξιοπιστία ιστοσελίδων.
- Διεπαφές γραμμής εντολών.

Η Go είναι μια σχετικά νέα γλώσσα προγραμματισμού ανοιχτού κώδικα. Η Google τη σχεδίασε για να καλύψει τις ανάγκες των προγραμματιστών που εργάζονται σε έργα μεγάλης κλίμακας. Αν και ενσωματώνει έννοιες από άλλες γλώσσες όπως η C, η Go έχει μοναδικά χαρακτηριστικά όσον αφορά τις δυνατότητες ανάπτυξης εφαρμογών. Είναι περισσότερο προσανατολισμένη στη δημιουργία διακομιστών και backend συστημάτων, αλλά ως γλώσσα προγραμματισμού γενικού σκοπού, μπορεί να υλοποιήσει σχεδόν τα πάντα. Η Go είναι πολύ ισχυρή όσον αφορά την ταυτόχρονη χρήση, είναι ευκολότερο να δημιουργήσουμε εφαρμογές που αξιοποιούν αποτελεσματικά τις σύγχρονες δικτυωμένες μηχανές και τους πολλαπλούς

πολυ-πυρήνες, διατηρώντας ταυτόχρονα καθαρό κώδικα προγραμματισμού. Η Go είναι ιδανική για την ανάπτυξη ιστοσελίδων με τις βιβλιοθήκες της, τη σταθερή απόδοση, την ασφάλεια και την καθαρή σύνταξη.

Τα βασικότερα χαρακτηριστικά και πλεονεκτήματα της Golang είναι τα εξής:

- **Δυαδικό εκτελέσιμο αρχείο**: Ένα από τα πιο σημαντικά και ανεκτίμητα χαρακτηριστικά της Go είναι η μεταγλώττιση του κώδικα σε ένα δυαδικό αρχείο. Αυτό πρακτικά σημαίνει ότι δεν υπάρχει ανάγκη για κάποιο διερμηνευτή και συνεπώς ενισχύει την απόδοση και οργάνωση του έργου.
- **Αυτόματος συλλέκτης σκουπιδιών**: Η Go είναι μια γλώσσα υψηλότερου επιπέδου με αυτόματη διαχείριση μνήμης χωρίς την ανάγκη παρέμβασης από τους προγραμματιστές. Αυτό διευκολύνει τις ομάδες προγραμματισμού να επικεντρωθούν σε πιο σημαντικά πράγματα και παράλληλα απομακρύνει τον κίνδυνο σφαλμάτων από την διαχείριση σφαλμάτων από προγραμματιστές και μείωσης της παραγωγικότητας τους.
- **Ανεπτυγμένες τεχνικές ταυτόχρονης χρήσης**: Η ύπαρξη Goroutines, Channels, Mutexes, WatiGroups της Go ενθαρρύνουν πρότυπα που επιτρέπουν σε τμήματα του κώδικα να συνομιλούν αποτελεσματικά και παράλληλα μεταξύ τους. Τα Goroutines είναι σαν φθηνά εικονικά νήματα που μπορούν να αλληλεπιδρούν με πραγματικά νήματα. Η Go έχει μια εξαιρετικά βελτιωμένη ροή εργασίας σε σύγκριση με την ταυτόχρονη χρήση σε μια ασύγχρονη πλατφόρμα όπως η NodeJS.
- **Ταχύτητα**: Η Go είναι αδιαμφισβήτητα μία από τις πιο γρήγορες γλώσσες προγραμματισμού αυτήν την στιγμή, στο κομμάτι ανάπτυξης εφαρμογών του backend κομματιού.
- **Δομή και μινιμαλισμός**: Κατά την ανάπτυξη κώδικα σε Go δεν υπάρχει η ανάγκη εγκατάστασης τρίτων εργαλείων και βιβλιοθηκών για την στοίχιση και δόμηση του κώδικα, όπως για παράδειγμα στη Javascript με το Prettier.js. Επιπλέον η συγγραφή κώδικα σε Go αποτελεί μια απλοϊκή διαδικασία καθώς ο μινιμαλισμός και η απλότητα επικρατούν ακόμη και σε πολύπλοκες συναρτήσεις.



Εικόνα 3.10 Λογότυπο και Μασκότ (Gopher) της Golang
(https://nepcodex.com/wp-content/uploads/2019/07/Golang-700x395.png)

Μερικοί πολλοί γνωστοί κολοσσοί που χρησιμοποιούν την Go για την ανάπτυξη ολόκληρης ή στοιχείων της εφαρμογής τους είναι οι Google, Dropbox, Uber, Alibaba, American Express, 1Password και SoundCloud. [30] [31]

3.11 Τεχνολογία ώθησης (SSE)

Το **SSE** (Server-Sent Events) είναι μια τεχνική που επιτρέπει σε ένα πρόγραμμα περιήγησης (πελάτη) να λαμβάνει αυτόματες ενημερώσεις από έναν διακομιστή, όπως δεδομένα συμβάντων βασισμένα σε κείμενο, μέσω μιας τυπικής σύνδεσης HTTP. Ο στόχος του SSE είναι να παρέχει στο πρόγραμμα περιήγησης έναν ομαλό τρόπο λήψης δεδομένων από το διακομιστή χωρίς να χρειάζεται να τα ζητήσει. Πρόκειται για μια προδιαγραφή που ορίζει τον τρόπο με τον οποίο οι διακομιστές αρχίζουν να στέλνουν δεδομένα στους πελάτες μετά την εγκαθίδρυση μιας σύνδεσης πελάτη-διακομιστή. Για να βελτιστοποιήσουν τη ροή μεταξύ των προγραμμάτων περιήγησης, χρησιμοποιούν ένα API JavaScript που ονομάζεται EventSource για να εγγραφούν στην πηγή συμβάντων και να μεταδώσουν μηνύματα ή συνεχείς ενημερώσεις σε έναν πελάτη. Κατά τη διάρκεια της αρχικοποίησης συμβάντος, η πηγή συμβάντος θα παρέχεται στο API EventSource. Αυτό θα ελέγχει τη σύνδεση με την πηγή και θα στέλνει αυτόματα τις αλλαγές στους πελάτες. Αυτή η έννοια καθιστά την εργασία με δεδομένα πραγματικού χρόνου εξαιρετικά αποδοτική, καθώς χρησιμοποιεί μόνο μία σύνδεση HTTP μακράς διάρκειας. Το SSE είναι ένα πρότυπο που περιγράφει τον τρόπο με τον οποίο οι διακομιστές μπορούν να ξεκινήσουν τη μετάδοση δεδομένων προς τους

πελάτες, μόλις δημιουργηθεί μια αρχική σύνδεση με τον πελάτη. Παρέχει μια αποδοτική ως προς τη μνήμη υλοποίηση της ροής XHR. Σε αντίθεση με μια ακατέργαστη σύνδεση XHR, η οποία αποθηκεύει ολόκληρη τη ληφθείσα απάντηση μέχρι τη διακοπή της σύνδεσης, μια σύνδεση SSE μπορεί να απορρίπτει επεξεργασμένα μηνύματα χωρίς να συσσωρεύονται όλα στη μνήμη. Το SSE μεταδίδει δεδομένα σε κωδικοποιημένο κείμενο UTF-8.

Τα πλεονεκτήματα του SSE είναι τα εξής

- Απλοποιημένο πρωτόκολλο.
- Μεταφορά μέσω απλού HTTP αντί για ένα προσαρμοσμένο πρωτόκολλο.
- Ενσωματωμένη υποστήριξη για επανασύνδεση και event-id.
- Αποφεύγονται προβλήματα με τείχη προστασίας που κάνουν επιθεώρηση πακέτων.

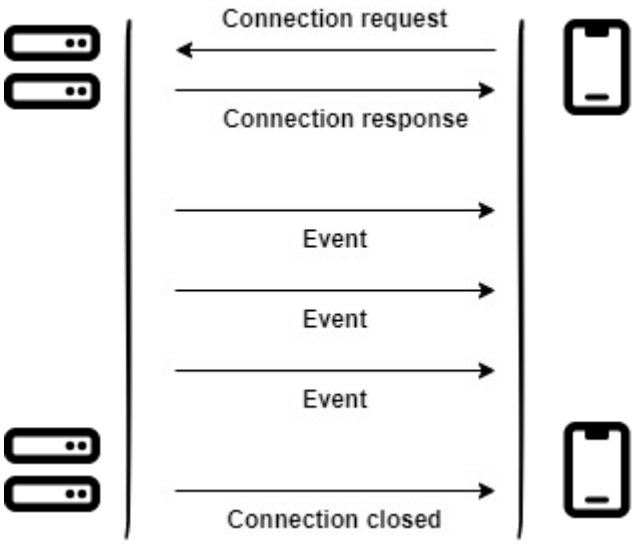
Όπως απεικονίζεται στην **εικόνα 3.12**, το SSE είναι μία μονόδρομη επικοινωνία, επομένως επιτρέπει την αποστολή δεδομένων μόνο από τον διακομιστή προς τον πελάτη. Το SSE έχει σχεδιαστεί για να είναι πιο αποδοτικό από τη μακροχρόνια αμφίδρομη επικοινωνία και περιλαμβάνει ορισμένα χαρακτηριστικά όπως:

- **Αυτόματη επανασύνδεση**: εάν ένας πελάτης χάσει τη σύνδεση με την πηγή συμβάντων, η επανασύνδεση επαναλαμβάνεται αυτόματα μετά από ορισμένο χρονικό διάστημα (η διάρκεια του οποίου μπορεί να προσαρμοστεί).
- **Αναγνωριστικά συμβάντος**: σε κάθε αποστελλόμενο συμβάν μπορεί να αποδοθεί ένα μοναδικό αναγνωριστικό.
- **Η δυνατότητα αποστολής αυθαίρετων συμβάντων**.

Περιορισμοί χρήσης του SSE

- Περιορίζεται σε UTF-8 και δεν υποστηρίζει δυαδικά δεδομένα.
- Όταν δεν χρησιμοποιείται μέσω του HTTP/2, η προσέγγιση των συμβάντων που αποστέλλονται από τον διακομιστή υπόκειται σε περιορισμούς όσον αφορά τον μέγιστο αριθμό ανοιχτών συνδέσεων. Αυτό μπορεί να είναι ιδιαίτερα οδυνηρό όταν ανοίγετε διάφορες καρτέλες, καθώς το όριο είναι ανά πρόγραμμα περιήγησης και έχει οριστεί σε πολύ χαμηλό αριθμό (6).
- Δεν διαθέτει εγγενή υποστήριξη του προγράμματος περιήγησης. Ωστόσο, υπάρχουν διαθέσιμες λύσεις με poly-fill (αντιγραφή ενός API) σε Javascript που προσομοιώνει

τη λειτουργικότητα του SSE για την επίλυση αυτού του προβλήματος. Όλα τα σύγχρονα προγράμματα περιήγησης υποστηρίζουν γεγονότα που αποστέλλονται από τον διακομιστή: Firefox 6+, Google Chrome 6+, Opera 11.5+, Safari 5+, Microsoft Edge 79+.



Εικόνα 3.11 Διαδικασία επικοινωνίας διακομιστή-πελάτη μέσω SSE

Υπάρχουν πολλές εφαρμογές όπου η αποστολή δεδομένων από τον πελάτη δεν είναι απαραίτητη και προτεινόμενη χρήση είναι το SSE. Το SSE είναι ιδιαίτερα χρήσιμο σε ενημερώσεις κατάστασης, ροές ειδήσεων κοινωνικών μέσων, ειδοποιήσεις push, ενημερωτικά δελτία κλπ. Με βάση αυτά συνεπώς έγινε η επιλογή του SSE ώστε να μπορεί να ενημερώνεται ανά πάσα στιγμή η εφαρμογή μας τόσο στη διαχειριστική σελίδα όσο και στο widget των πελατών για αλλαγές στην διαθεσιμότητα των κρατήσεων. Μέσω του SSE αποφεύχθηκε η χρήση της τεχνολογίας του WebSocket που παρόλο που ικανοποιεί πλήρως τις απαιτήσεις μας, προσθέτει αχρείαστη πολυπλοκότητα στην υλοποίηση της εφαρμογής.[32]

3.12 PostgreSQL

Η PostgreSQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) ανοικτού κώδικα. Είναι γνωστή για την επεκτασιμότητα, την αξιοπιστία και τις επιδόσεις της και χρησιμοποιείται συχνά για μεγάλους και πολύπλοκους φόρτους εργασίας δεδομένων. Υποστηρίζει μεγάλη ποικιλία τύπων δεδομένων και χαρακτηριστικά βελτιστοποίηση

επιδόσεων, τα οποία είναι διαθέσιμα σε βάσεις δεδομένων όπως η Oracle και ο SQL Server. Η PostgreSQL είναι επίσης διαθέσιμη σχεδόν για κάθε λειτουργικό σύστημα με την τελευταία σταθερή της έκδοση. Αυτή τη στιγμή εκδόθηκε η τελευταία έκδοση της που είναι η PostgreSQL 15, υποστηρίζοντας ένα πολύ μεγάλο εύρος χαρακτηριστικά όπως η ταχύτητα, οι πολύπλοκες επερωτήσεις, η ασφάλεια δεικτοδότηση και φυσικά σε όλα αυτά μπορούμε να συγκαταλέξουμε και την πολύ μεγάλη κοινότητα της που έχει αναπτυχθεί γύρω από την PostgreSQL.

Τεχνικά χαρακτηριστικά της PostgreSQL:

- **DBMS ανοιχτού κώδικα**: Μόνο η PostgreSQL παρέχει επιχειρησιακές επιδόσεις και λειτουργίες μεταξύ των σημερινών Open Source DBMS με απεριόριστες δυνατότητες ανάπτυξης. Επίσης, οι χρήστες της PostgreSQL μπορούν να συμμετέχουν άμεσα στην κοινότητα και να δημοσιεύουν και να μοιράζονται τις δυσχέρειες και τα σφάλματα.
- **Συναρτήσεις**: Οι συναρτήσεις SQL που ονομάζονται "Store Procedure" μπορούν να χρησιμοποιηθούν για περιβάλλον διακομιστή. Επίσης, υποστηρίζουν γλώσσες παρόμοιες με την PL/SQL στην Oracle, όπως PL/pgSQL, PL/Python, PL/Perl, C/C++ και PL/R.
- **ACID και συναλλαγές**: Η PostgreSQL υποστηρίζει ACID (Ατομικότητα, Συνέπεια, Απομόνωση, Ανθεκτικότητα).
- **Ποικίλες τεχνικές ευρετηρίων**: Η PostgreSQL δεν παρέχει μόνο τεχνικές δεικτών δέντρων B+, αλλά και διάφορα είδη τεχνικών όπως το GIN (Generalized Inverted Index) και το GiST (Generalized Search Tree), κ.λπ.
- **Ευέλικτη αναζήτηση κειμένου**: Η αναζήτηση πλήρους κειμένου είναι διαθέσιμη κατά την αναζήτηση συμβολοσειρών με την εκτέλεση της διανυσματικής λειτουργίας και της αναζήτησης συμβολοσειρών.
- **Ποικίλα είδη αντιγραφής**: Η PostgreSQL υποστηρίζει μια ποικιλία μεθόδων αντιγραφής όπως Streaming Replication , Slony-I και cascading.
- **Διαφοροποιημένες λειτουργίες επέκτασης**: Η PostgreSQL υποστηρίζει διάφορα είδη τεχνικών για την αποθήκευση γεωγραφικών δεδομένων, όπως PostGIS, Key-Value Store και DBLink. [33]

3.13 Json Web Token (JWT)

Το Json Web Token (JWT) είναι ένα ανοιχτό πρότυπο (RFC 7519) που ορίζει έναν ασφαλή τρόπο μετάδοσης πληροφοριών σε μορφή JSON μεταξύ δύο μερών. Οι πληροφορίες αυτές είναι ψηφιακά υπογεγραμμένες, γεγονός που επιτρέπει την επανάληψη τους και την εξασφάλιση της αξιοπιστίας τους. Τα JWTs μπορούν να υπογραφούν με χρήση μυστικού κλειδιού (με αλγόριθμο HMAC) ή δημόσιου/ιδιωτικού κλειδιού (με RSA ή ECDSA). Τα υπογεγραμμένα tokens επιτρέπουν την επαλήθευση της ακεραιότητας των πληροφοριών που περιέχουν, ενώ τα κρυπτογραφημένα tokens κρύβουν τις πληροφορίες από άλλα μέρη. Η χρήση δημόσιου/ιδιωτικού κλειδιού εξασφαλίζει επίσης ότι μόνο το μέρος που διαθέτει το ιδιωτικό κλειδί μπορεί να υπογράψει τα tokens. Αν εξετάσουμε τα σενάρια και το πότε μπορούμε να χρησιμοποιήσουμε τον JWT, έχουμε τις εξής δύο περιπτώσεις:

- **Εξουσιοδότηση**: Αυτό είναι το πιο συνηθισμένο σενάριο για τη χρήση JWT. Μόλις ο χρήστης συνδεθεί σε μία εφαρμογή θα λάβει σαν απάντηση ένα JWT, κάθε επόμενη αίτηση θα περιλαμβάνει το JWT, επιτρέποντας στον χρήστη να έχει πρόσβαση σε διαδρομές, υπηρεσίες και πόρους που επιτρέπονται με αυτό το token. Το Single Sign On είναι ένα χαρακτηριστικό που χρησιμοποιεί ευρέως το JWT στις μέρες μας, λόγω της μικρής επιβάρυνσης του και της δυνατότητας του να χρησιμοποιείται εύκολα σε διαφορετικούς τομείς.
- **Ανταλλαγή πληροφοριών**: Τα JWTs είναι ένας καλός τρόπος ασφαλούς μετάδοσης πληροφοριών μεταξύ των μερών. Επειδή τα JWTs μπορούν να υπογραφούν χρησιμοποιώντας ζεύγη δημόσιων/ιδιωτικών κλειδιών μπορούμε να είμαστε σίγουροι ότι οι αποστολείς είναι αυτοί που λένε ότι είναι. Επιπλέον, καθώς η υπογραφή υπολογίζεται χρησιμοποιώντας την επικεφαλίδα και το ωφέλιμο φορτίο, μπορούμε επίσης να επαληθεύσουμε ότι το περιεχόμενο δεν έχει αλλοιωθεί.

Σε μια διαδικτυακή εφαρμογή, τα JWTs χρησιμοποιούνται συνήθως για να επιβεβαιώσουν την ταυτότητα των χρηστών και να μεταφέρουν πληροφορίες σχετικά με αυτούς ανάμεσα στον πελάτη και τον διακομιστή. Ο διακομιστής δημιουργεί ένα JWT για το χρήστη και το αποστέλλει στον πελάτη μετά τη σύνδεσή του. Στη συνέχεια, ο πελάτης συμπεριλαμβάνει το JWT στην επικεφαλίδα "Authorization" στα επόμενα αιτήματα που αποστέλλει στον διακομιστή. Ο διακομιστής μπορεί να χρησιμοποιήσει το JWT για να επιβεβαιώσει την ταυτότητα του χρήστη και να του παραχωρήσει πρόσβαση σε προστατευμένους πόρους. Τα JWT επίσης χρησιμοποιούνται συχνά για την υλοποίηση της ενιαίας σύνδεσης (SSO) σε πολλαπλές εφαρμογές. Είναι σημαντικό να αποθηκεύονται τα JWT σε HttpOnly και Secure

cookies για να αποτραπούν επιθέσεις XSS και άλλων ειδών επιθέσεων. Τέλος, το μυστικό κλειδί που χρησιμοποιείται για την υπογραφή του JWT θα πρέπει να διατηρείται μυστικό.

Η δομή ενός Json Web Token στη συμπαγή και ολοκληρωμένη μορφή του περιλαμβάνει τρία διαφορετικά κομμάτια που χωρίζονται μεταξύ τους με τελεία (.).

- **Επικεφαλίδα**: Η επικεφαλίδα αποτελείται συνήθως από δύο μέρη, τον τύπο του συμβόλου που είναι JWT και τον αλγόριθμο υπογραφής που χρησιμοποιήθηκε, όπως για παράδειγμα HMAC, SHA256 ή RSA.

```
{
      "alg": "HS256",
      "typ": "JWT"
}
```

- **Ωφέλιμο φορτίο**: Αυτό το μέρος περιλαμβάνει ισχυρισμούς οι οποίοι είναι δηλώσεις σχετικά με μία οντότητα (συνήθως με τον χρήστη) και πρόσθετα δεδομένα. Υπάρχουν τρεις τύποι ισχυρισμοί και είναι οι καταχωρημένοι, οι δημόσιοι και οι ιδιωτικοί. Οι καταχωρημένοι ισχυρισμοί είναι ένα σύνολο προκαθορισμένων ισχυρισμών που συνιστώνται ώστε να παρέχεται ένα σύνολο χρήσιμων λειτουργικών ισχυρισμών. Μερικά παραδείγματα είναι το iss (issuer), exp (expiration time), sub (subject) και aud (audience). Οι δημόσιοι ισχυρισμοί μπορούν να οριστούν κατά βούληση από όσους χρησιμοποιούν JWT. Προκειμένου όμως να αποφευχθούν συγκρούσεις θα πρέπει να ορίζονται στο IANA JSON Web Token Registry ή να ορίζονται ως URI που περιέχει ένα namespace ανθεκτικό στις συγκρούσεις.

```
{
      "sub": "1234567890",
      "name": "Dimitris Rakas",
      "admin": true
}
```

- **Υπογραφή**: Για να δημιουργήσουμε το τρίτο μέρος αυτό της υπογραφής πρέπει να πάρουμε την κωδικοποιημένη επικεφαλίδα, το κωδικοποιημένο ωφέλιμο φορτίο, ένα μυστικό, τον αλγόριθμο που καθορίζεται στην επικεφαλίδα και να το υπογράψουμε. Για παράδειγμα, αν θέλουμε να χρησιμοποιήσουμε τον αλγόριθμο HMAC SHA256, η υπογραφή θα δημιουργηθεί με τον ακόλουθο τρόπο:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
```

```
    secret)
```

Η υπογραφή χρησιμοποιείται για να επαληθεύσει ότι το μήνυμα δεν άλλαξε στην πορεία και, στην περίπτωση των tokens που έχουν υπογραφεί με ιδιωτικό κλειδί, μπορεί επίσης να επαληθεύσει ότι ο αποστολέας του JWT είναι αυτός που λέει ότι είναι.

Συνθέτοντας και τα παραπάνω μέρη μαζί έχουμε τρεις συμβολοσειρές Base64Url που χωρίζονται μεταξύ τους με τελείες και μπορούν εύκολα να περάσουν σε περιβάλλοντα HTML και HTTP, ενώ είναι πιο σταθερά σε σχέση με πρότυπα που βασίζονται σε XML, όπως το SAML. Παρακάτω έχουμε ένα παράδειγμα ενός JWT που έχεις κωδικοποιήσει την παραπάνω επικεφαλίδα, το ωφέλιμο φορτίο και έχει υπογραφεί με ένα μυστικό κλειδί. [34]

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpbWl0cmlzIFJha2FzIiwiYWRtaW4iOnRydWV9.TCx4I0hajY_ljKgxGo5UXqhZLkPYs94fGDMsJfSUS94
```

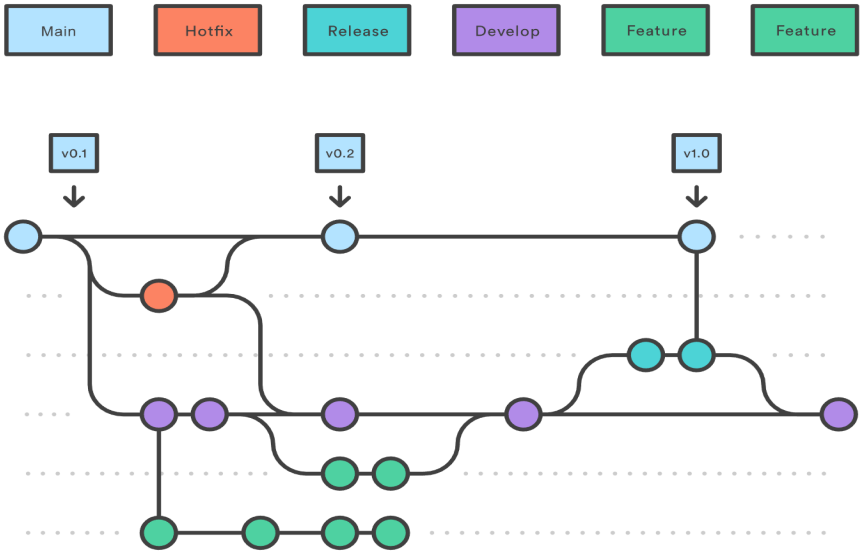
3.14 Version Control

Ο έλεγχος εκδόσεων αφορά την παρακολούθηση και διαχείριση των αλλαγών στον κώδικα λογισμικού. Τα συστήματα ελέγχου εκδόσεων αποτελούν εργαλεία λογισμικού που βοηθούν τις ομάδες λογισμικού να διαχειρίζονται τις αλλαγές στον πηγαίο κώδικα με την πάροδο του χρόνου, προκειμένου να εργάζονται ταχύτερα και εξυπνότερα. Αυτά τα συστήματα είναι ιδιαίτερα χρήσιμα για τις ομάδες DevOps, καθώς βοηθούν στη μείωση του χρόνου ανάπτυξης και την αύξηση της αποτελεσματικότητας. Εάν προκύψει κάποιο λάθος, οι προγραμματιστές μπορούν να επιστρέψουν σε προηγούμενες εκδόσεις του κώδικα για να βοηθήσουν στη διόρθωσή του και να μειώσουν την αναστάτωση στην ομάδα. Το σύστημα ελέγχου εκδόσεων επιτρέπει σε πολλούς χρήστες να συνεργάζονται σε ένα έργο και να ανατρέχουν σε προηγούμενες εκδόσεις του έργου, να συγχωνεύουν αλλαγές και να διατηρούν ιστορικό της ανάπτυξής του. Τα πιο γνωστά συστήματα ελέγχου εκδόσεων περιλαμβάνουν το Git, το Mercurial και το Subversion. Οι προγραμματιστές λογισμικού που εργάζονται σε ομάδες γράφουν συνεχώς νέο πηγαίο κώδικα και αλλάζουν τον υπάρχοντα πηγαίο κώδικα. Ο κώδικας για ένα έργο, μια εφαρμογή ή ένα στοιχείο λογισμικού είναι συνήθως οργανωμένος σε μια δομή φακέλων ή σε ένα "δέντρο αρχείων". Ένας προγραμματιστής της ομάδας μπορεί να εργάζεται σε ένα νέο χαρακτηριστικό, ενώ ένας άλλος προγραμματιστής διορθώνει ένα άσχετο σφάλμα αλλάζοντας κώδικα, κάθε

προγραμματιστής μπορεί να κάνει τις αλλαγές του σε διάφορα μέρη του δέντρου αρχείων. Το λογισμικό ελέγχου εκδόσεων αποτελεί ουσιαστικό μέρος της καθημερινής εργασίας των προγραμματιστών και αποτέλεσε φυσικά και αντικείμενο στην ανάπτυξη της διπλωματικής μας εργασίας καθώς χρειάστηκε και τα δύο άτομα της διπλωματικής να αναπτύσουν κομμάτια του κώδικα σε διαφορετικά σημεία του δέντρου χωρίς ο ένας να παρεμποδίζει την εργασία του άλλου και παράλληλα να αποφευγονται λάθη ανατρέχοντας στο ιστορικό.

Μακράν, το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων στον κόσμο σήμερα είναι το Git. Το Git είναι ένα ώριμο, ενεργά συντηρούμενο έργο ανοικτού κώδικα που αναπτύχθηκε αρχικά το 2005 από τον Linus Torvalds, τον διάσημο δημιουργό του πυρήνα του λειτουργικού συστήματος Linux. Ένας εντυπωσιακός αριθμός έργων λογισμικού βασίζεται στο Git για τον έλεγχο εκδόσεων, συμπεριλαμβανομένων εμπορικών έργων καθώς και έργων ανοικτού κώδικα. Έχοντας μια κατανεμημένη αρχιτεκτονική, το Git είναι ένα παράδειγμα ενός **DVCS** (Distributed Version Control System). Αντί να υπάρχει μόνο ένα ενιαίο μέρος για το πλήρες ιστορικό εκδόσεων του λογισμικού, όπως συνηθίζεται στα κάποτε δημοφιλή συστήματα ελέγχου εκδόσεων όπως το CVS ή το Subversion. Στο Git το αντίγραφο εργασίας του κώδικα κάθε προγραμματιστή είναι επίσης ένα αποθετήριο που μπορεί να περιέχει το πλήρες ιστορικό όλων των αλλαγών. [35] Ορισμένα βασικά χαρακτηριστικά του Git περιλαμβάνουν:

- **Κατανεμημένος έλεγχος εκδόσεων**
- **Διακλάδωση και συγχώνευση**
- **Ιστορικό**
- **Συνεργασία**



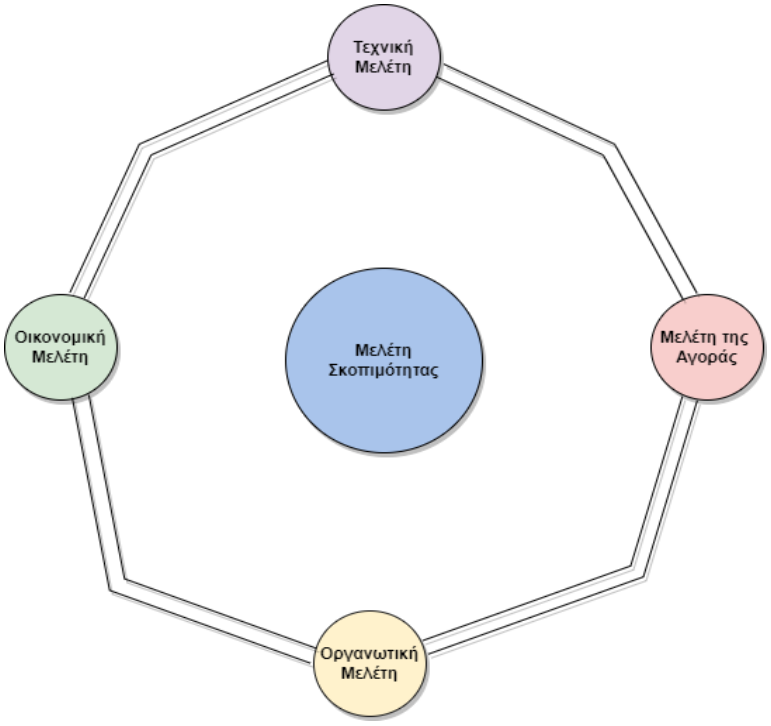
Εικόνα 3.12 Αποτύπωση δέντρου διακλαδώσεων ελέγχων σε διαφορετικά στάδια ανάπτυξης

Κεφάλαιο 4ο

Προδιαγραφές του Συστήματος Κρατήσεων

4.1 Μελέτη Σκοπιμότητας

Η μελέτη σκοπιμότητας για το σύστημα κρατήσεων μας είναι ένα σημαντικό βήμα για να διαπιστώσουμε τις προκλήσεις αλλά και την ανάγκη ανάπτυξης ενός τέτοιου προϊόντος και κατά πόσο καλύπτει τις ανάγκες των χρηστών. Πιο συγκεκριμένα θα περιλαμβάνει την αξιολόγηση των δυνατοτήτων επιτυχίας του συστήματος και τον προορισμό του κατά πόσον αποτελεί βιώσιμη επιλογή. Οι βασικότεροι παράγοντες που θα εξετάσουμε και αποτελούν δομικά στοιχεία μιας μελέτης σκοπιμότητας περιλαμβάνουν:



Εικόνα 4.1 Πεδία μελέτης σκοπιμότητας

1. **Ζήτηση στην αγορά**: Έπειτα και από την COVID εποχή η ανάγκη για συστήματα κρατήσεων είναι επιτακτική, καθώς προσφέρει στους πελάτες την ευκολία να πραγματοποιούν κρατήσεις από οπουδήποτε και οποτεδήποτε χωρίς την άμεση

επικοινωνία με την επιχείρηση. Η αγορά των χώρων ψυχαγωγίας αυτή τη στιγμή διαθέτει δύο κεντρικοποιημένες πλατφόρμες κρατήσεων, εμείς καλούμαστε να διαφοροποιηθούμε από αυτές μέσω των δυνατοτήτων και της καινοτομίας που θα εντάξουμε στο σύστημα μας.

2. **Ανταγωνισμός**: Οι κύριοι ανταγωνιστές στον χώρο των δωματίων απόδρασης προσφέρουν online κρατήσεις μέσω δικών τους πλατφορμών και εφαρμογών για κινητά. Οι πλατφόρμες αυτές κλειδώνουν όμως τις επιχειρήσεις σε μία συγκεκριμένη υπηρεσία χωρίς να δίνουν την δυνατότητα για διασύνδεση με τρίτους παρόχους και την εξατομίκευση που θέλουν οι επιχειρήσεις για να προβάλουν την ποιότητα των υπηρεσιών τους. Μερικά χαρακτηριστικά του δικού μας συστήματος είναι η δυνατότητα εναλλαγής του προγράμματος και η πολλαπλή διάθεση των υπηρεσιών τους σε ιδιωτικά κανάλια.
3. **Τεχνική σκοπιμότητα**: Το σύστημα κρατήσεων μας θα πρέπει να αναπτυχθεί ώς ανταποκρινόμενος ιστότοπος (responsive), χρησιμοποιώντας έναν συνδυασμό από τεχνολογίες όπως HTML, CSS και JavaScript για το frontend και ένα backend διακομιστή που τρέχει σε λειτουργικό σύστημα Linux. Το σύστημα κρατήσεων μας θα πρέπει να ενσωματώνεται με την υπάρχουσα ιστοσελίδα της επιχειρήσεις ώστε να παρέχει τους χώρους ψυχαγωγίας, την ενημέρωση της διαθεσιμότητας και τη διαχείριση των κρατήσεων.

Σε αυτήν την μελέτη αποκλείσαμε την εξέταση του οικονομικού κόστους και της βιωσιμότητας του με χρηματικά δεδομένα στην αγορά καθώς δεν αποτελεί αντικείμενο επεξήγησης στην ανάπτυξη του συστήματος κρατήσεων μας στα πλαίσια της διπλωματικής εργασίας. Συνολικά, η μελέτη σκοπιμότητας υποδηλώνει ότι το δικό μας σύστημα κρατήσεων έχει τη δυνατότητα να είναι επιτυχημένο, υπό την προϋπόθεση ότι είναι σε θέση να διαφοροποιηθεί από τους ανταγωνιστές και να ανταποκριθεί στις τεχνικές απαιτήσεις.

4.2 Απαιτήσεις Λογισμικού

Προκειμένου να πραγματοποιηθεί μια σωστή και τεκμηριωμένη παραγωγή λογισμικού χρειάζονται κάποια αρχικά στάδια τα οποία είναι η ανάλυση των απαιτήσεων χρήστη και του συστήματος. Ξεκινώντας από το θεωρητικό κομμάτι οι απαιτήσεις ενός συστήματος ορίζονται ως οι περιγραφές των υπηρεσιών που παρέχονται από το σύστημα και οι λειτουργικοί περιορισμοί του. Οι απαιτήσεις επί της ουσίας αντανακλούν τις ανάγκες των πελατών για ένα σύστημα που βοηθάει για την επίλυση κάποιων προβλημάτων. Η διαδικασία του εντοπισμού,

της ανάλυσης, της τεκμηρίωσης, του ελέγχου αυτών των υπηρεσιών και των περιορισμών ονομάζεται τεχνολογία απαιτήσεων (requirements engineering). Για την διάκριση τους, θα χρησιμοποιήσουμε τον όρο απαιτήσεις χρήστη (user requirements) για τις αφηρημένες απαιτήσεις υψηλού επιπέδου, και απαιτήσεις συστήματος (system requirements) για τη λεπτομερή περιγραφή του τι πρέπει να κάνει το σύστημα.

1. Οι **απαιτήσεις χρήστη** είναι δηλώσεις σε φυσική γλώσσα και διαγράμματα των υπηρεσιών που αναμένεται να παρέχει το σύστημα και των περιορισμών κάτω από τους οποίους πρέπει να λειτουργεί.
2. Οι **απαιτήσεις συστήματος** περιγράφουν με λεπτομέρειες τις λειτουργίες, τις υπηρεσίες και τους λειτουργικούς περιορισμούς του συστήματος. Το έγγραφο των απαιτήσεων συστήματος θα πρέπει να είναι ακριβές προκειμένου να υλοποιηθεί. Μπορεί να αποτελεί μέρος της σύμβασης μεταξύ του αγοραστή του συστήματος και των κατασκευαστών του λογισμικού.

Στην δική μας περίπτωση η ανάπτυξη μιας εφαρμογής πρέπει να αναπτυχθεί με γνώμονα την κάλυψη όλων των βασικών λειτουργικών και μη λειτουργικών απαιτήσεων που οφείλει ένα σύστημα κρατήσεων να διαθέτει. Λίγα λόγια για τις λειτουργικές και μη απαιτήσεις μιας εφαρμογής σε θεωρητικό επίπεδο πριν εμβαθύνουμε στο σύστημα κρατήσεων μας είναι:

1. **Λειτουργικές απαιτήσεις** (Functional requirements): πρόκειται για δηλώσεις που ορίζουν ποιες υπηρεσίες θα πρέπει να παρέχει το σύστημα, πώς θα πρέπει να αντιδρά σε συγκεκριμένες εισόδους, και πως θα πρέπει να συμπεριφέρεται σε συγκεκριμένες καταστάσεις. Σε μερικές περιπτώσεις, οι λειτουργικές απαιτήσεις μπορούν επίσης να δηλώνουν ρητά τι δεν θα πρέπει να κάνει το σύστημα.
2. **Μη λειτουργικές απαιτήσεις** (Non-functional requirements): Πρόκειται για περιορισμούς στις υπηρεσίες ή τις λειτουργίες που προσφέρει το σύστημα. Περιλαμβάνουν χρονικούς περιορισμούς, περιορισμούς της διαδικασίας ανάπτυξης και πρότυπα. Οι μη λειτουργικές απαιτήσεις συχνα έχουν εφαρμογή στο σύστημα ώς σύνολο, και δεν αφορούν μεμονωμένα χαρακτηριστικά ή υπηρεσίες του.
3. **Απαιτήσεις πεδίου** (Domain requirements): Πρόκειται για απαιτήσεις που προέρχονται από το πεδίο εφαρμογής του συστήματος και αντανακλούν χαρακτηριστικά και περιορισμούς αυτού του πεδίου. Μπορεί να είναι λειτουργικές ή μη λειτουργικές απαιτήσεις.

Έχοντας σαν οδηγό τα παραπάνω ερχόμαστε να αναλύσουμε το αντικείμενο και εν συνεχεία το σκοπό της εφαρμογής μας. Το λογισμικό αναπτύχθηκε έτσι ώστε να μπορεί να προσφέρει σε πολλαπλούς ιδιοκτήτες χώρων ψυχαγωγίας έναν αποτελεσματικό τρόπο για την διαχείριση των κρατήσεων τους ψηφιακά, με εύκολη πρόσβαση μέσω του διαδικτύου, χωρίς να χρειάζεται να κατεβάζουν κάποιο λογισμικό η να εμπλέκονται με περίπλοκες εγκαταστάσεις.

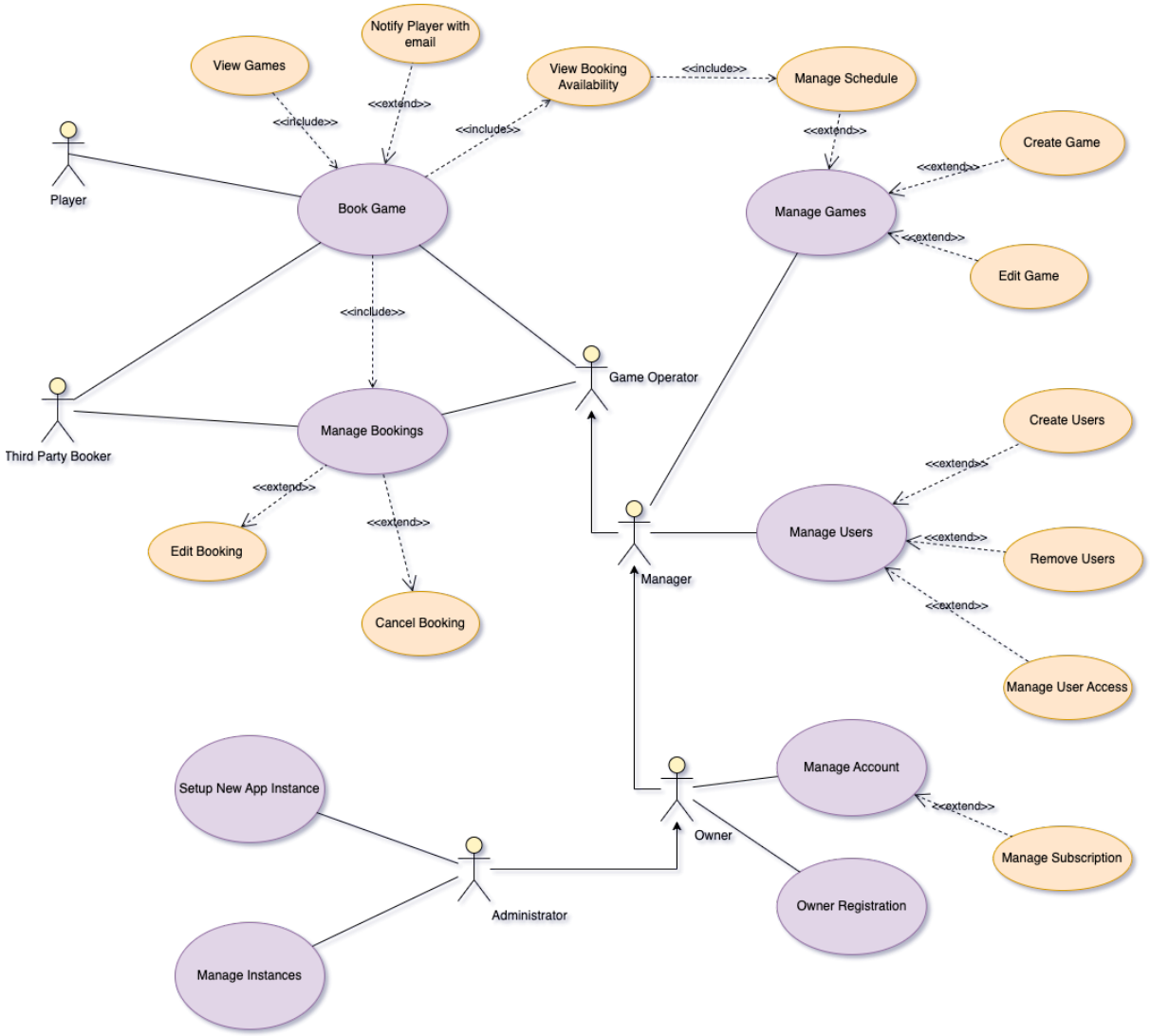
Ανάλυση απαιτήσεων χρήστη

Η απαιτήσεις ενός συστήματος λογισμικού περιγράφουν τι πρέπει να κάνει το σύστημα και ορίζουν περιορισμούς που αφορούν τη λειτουργία και την υλοποίηση του. Μέσω της συμφωνίας των χρηστών και των αναλυτών-τεχνικών διαμορφώνεται ένα σύνολο απαιτήσεων που θα καθιστούν την εφαρμογή χρηστική και ώριμη προς ανάπτυξη. Στο δικό μας σύστημα οι απαιτήσεις που κρίθηκαν απαραίτητες ώστε να μπορούν να ικανοποιηθούν τόσο η πλευρά της επιχείρησης αλλα και του διανομέα του λογισμικού. Το λογισμικό θα μπορεί να συντηρείτε και να διανέμεται εύκολα. Ο διαχειριστής της επιχείρησης θα μπορεί να ορίζει τους χώρους ψυχαγωγίας και την διαθεσιμότητα τους, να κάνει νέες κρατήσεις καθώς και να επεξεργάζεται τις υπάρχουσες σε περιπτώσεις αλλαγών η ακυρώσεων. Επίσης η επιχείρηση θα μπορεί να επιτρέπει στους πελάτες να κάνουν μόνοι τους κράτηση διαδικτυακά αλλά και να δίνει πρόσβαση σε τρίτες υπηρεσίες κρατήσεων για να πετυχαίνει μεγαλύτερη πληρότητα. Στην **εικόνα 4.2** φαίνεται η δομή που θα πρέπει να έχει το σύστημα όπως προκύπτει από τις παραπάνω λειτουργίες.

Εικόνα 4.2 Αρχιτεκτονικό μοντέλο συστήματος

Συνεχίζοντας την προσέγγιση μας ανθρωποκεντρικά και διαμορφώνοντας τους ρόλους και τα χαρακτηριστικά των χρηστών που θα διαθέτει η εφαρμογή μας όπως προκύπτουν από το διάγραμμα περιπτώσεων χρήσης (use case) **εικόνα 4.3** έχουμε ξεχωρίσει τεσσάρων ειδών χρήστες, ο **διαχειριστής συστήματος** (master administrator), ο **διαχειριστής** (admin), ο **χρήστης τρίτων υπηρεσιών** (third party booker) και ο **πελάτης**.

- Ο **χρήστης διαχείρισης συστήματος** (Master Administrator) είναι ο πιο προιμοδοτημένος χρήστης της εφαρμογής καθώς διαθέτει τα δικαιώματα και τη δυνατότητα για πιο συστημικές ενέργειες. Αυτές είναι η εγκατάσταση ενός αντιγράφου της εφαρμογής (Instance) για κάθε νέο πάροχο υπηρεσιών (Owner) και η διαχείριση της εφαρμογής, καθώς και όλες οι ενέργειες που μπορεί να κάνει ο διαχειριστής.
- Ο **χρήστης διαχειριστής** (Admin) αποτελεί τον χρήστη της διαχειριστικής σελίδας του συστήματος κρατήσεων. Χωρίζεται σε τρεις διαφορετικούς ρόλους βάση δικαιωμάτων και ιεραρχίας τους. Αυτοί είναι, ιεραρχόντας τους από τον πιο προιμοδοτημενο στον λιγότερο, ο **ιδιοκτήτης** (Owner), ο **διαχειριστής** (Manager) και ο **χειριστής του παιχνιδιού** (Game Operator/Master). Οι πιο ψηλά στην ιεραρχία χρήστες θα κληρονομούν και όλες τις δυνατότητες τον χαμηλότερων. Ο ιδιοκτήτης θα μπορεί να διαχειρίζεται το λογαριασμό της επιχείρησης στο σύστημα κρατήσεων που έχει πληρώσει ως υπηρεσία, να ολοκληρώνει τις συνδρομές που απαιτούνται για την πρόσβαση στο λογισμικο και να ξεκινάει την αρχική εκδήλωση ενδιαφέροντος της επιχείρησης για το σύστημα κρατήσεων. Ο διαχειριστής έχει τη δυνατότητα αρχικά να δημιουργεί και να διαχειρίζεται τους χρήστες του αντιγράφου της εφαρμογής της επιχείρησης που ανήκει. Επίσης θα έχει τη δυνατότητα της δημιουργία και διαχείρισης των χώρων ψυχαγωγίας εκ μέρους της επιχείρησης καθώς και να θέτει το ωράριο λειτουργίας και την διαθεσιμότητα του κάθε χώρου. Τέλος ο χειριστής του παιχνιδιού θα μπορεί να πραγματοποιήσει και να διαχειριστεί τις κρατήσεις καθώς και να παρακολουθεί την εύρυθμη λειτουργία των ραντεβού που πραγματοποιείται από τους τελικούς χρήστες.
- Ο **χρήστης τρίτων υπηρεσιών** (Third party booker) θα έχει την δυνατότητα να παρακολουθεί τις κρατήσεις των χώρων ψυχαγωγίας που έχουν πραγματοποιηθεί καθώς και να διαχειρίζεται όσες έρχονται από το δικό του σύστημα. Μέσα στο πλαίσιο της διαχείρισης θα έχει την δυνατότητα φυσικά να πραγματοποιεί και αυτός κρατήσεις, να βλέπει τα δωμάτια και την διαθεσιμότητα τους.
- Ο **πελάτης** θα έχει τη δυνατότητα να πραγματοποιεί κρατήσεις μέσω ενός περιηγητή, θα δύναται να βλέπει σε πραγματικό χρόνο την διαθεσιμότητα των χώρων ψυχαγωγίας ανά ημέρα και ώρα και να αποφασίζει πότε επιθυμεί να πραγματοποιήσει μία κράτηση.



Εικόνα 4.3 Διάγραμμα περιπτώσεων χρήσης των χρηστών και των δυνατοτήτων τους

Ειδικές απαιτήσεις συστήματος

Οι απαιτήσεις ενός συστήματος χωρίζονται σε δύο βασικές κατηγορίες που είναι οι λειτουργικές και μη λειτουργικές απαιτήσεις. Ξεκινώντας με την πρώτη κατηγορία οι λειτουργικές απαιτήσεις ενός συστήματος περιγράφουν τι θα πρέπει να κάνει το σύστημα. Όταν εκφράζονται ως απαιτήσεις χρήστη, οι απαιτήσεις συνήθως περιγράφονται με αρκετά αφηρημένο τρόπο. Όταν όμως είναι λειτουργικές απαιτήσεις συστήματος πρέπει να περιγράφουν με λεπτομέρειες τη λειτουργία του συστήματος, τις εισόδους και τις εξόδους του καθώς και τις εξαιρέσεις. Ο βασικός κανόνας για τις απαιτήσεις συστήματος είναι ότι οι προδιαγραφές των λειτουργικών απαιτήσεων ενός συστήματος θα πρέπει να είναι και πλήρεις και συνεπείς. **Πληρότητα** (Completeness) σημαίνει ότι θα πρέπει να έχουν οριστεί όλες οι υπηρεσίες που απαιτούνται από το χρήστη. **Συνέπεια** (Consistency) σημαίνει ότι οι απαιτήσεις δεν πρέπει να έχουν ορισμούς που αντιφάσκουν.

Λειτουργικές απαιτήσεις

- Ο πελάτης θα πρέπει να είναι σε θέση να πραγματοποιεί κρατήσεις για το επιθυμητό δωμάτιο και ώρα διαλέγοντας από τη διαθεσιμότητα που του παρουσιάζεται μέσω της ιστοσελίδας της ενδιαφερόμενης επιχείρησης.
- Ένας τρίτος πάροχος θα είναι σε θέση να λαμβάνει από το σύστημα μας τη διαθεσιμότητα για τα δωμάτια και τα ωράρια, παράλληλα θα μπορεί να δημιουργεί κρατήσεις καθώς και να διαγράψει ή τροποποιήσει της κρατήσεις που προέρχονται από αυτόν μέσω ειδικής πρόσβασης στο σύστημα της κάθε επιχείρησης.
- Η διαχειριστική σελίδα θα πρέπει να παρέχει στον χρήστη με τα κατάλληλα δικαιώματα την δυνατότητα για δημιουργία επιπλέον χρηστών της διαχειριστικής σελίδας καθώς και να προσαρμόζει τα δικαιώματα τους. Οι παραπάνω ενέργειες όσων αφορά τα δικαιώματα των χρηστών θα μπορούν φυσικά να τροποποιηθούν ή και να διαγραφούν οι χρήστες. Με άλλα λόγια θα δίνεται η δυνατότητα διαχείρισης των χρηστών από τον ανώτερο διαχειριστή αλλά και άλλες ενέργειες όπως είναι η πληρωμή της συνδρομητικής χρήσης του λογισμικού κρατήσεων.
- Το σύστημα κρατήσεων θα πρέπει να επιτρέπει την δημιουργία, τροποποίηση ή και διαγραφή υπηρεσιών. Πιο συγκεκριμένα θα μπορούν να δημιουργηθούν νέα δωμάτια απόδρασης ή να τροποποιηθούν τα υπάρχοντα και να οριστούν τα κατάλληλα χαρακτηριστικά που πρέπει να τα συνοδεύουν όπως ο αριθμός παικτών και η διάρκεια. Μέσα στο πλαίσιο της τροποποίησης τους θα δύναται να γίνει και απενεργοποίηση των υπηρεσιών πλην της διαγραφής έως ότου ο διαχειριστής χρειαστεί να επαναφέρει την συγκεκριμένη υπηρεσία.
- Το σύστημα κρατήσεων θα πρέπει να επιτρέπει την δημιουργία και τροποποίηση προγραμμάτων λειτουργίας. Πιο συγκεκριμένα θα μπορούν να δημιουργηθούν νέα προγράμματα λειτουργία των δωματίων ή να τροποποιηθούν τα υπάρχοντα και να οριστούν τα κατάλληλα χαρακτηριστικά που πρέπει να τα συνοδεύουν όπως η ημερομηνία έναρξης ισχύος και τα ωράρια λειτουργίας. Μέσα στο πλαίσιο της τροποποίησης τους θα δύναται να καθορίζεται η σειριακή ακολουθία των προγραμμάτων λειτουργίας ώστε το ένα να ακολουθεί το άλλο και να διασφαλιστεί η εύρυθμη λειτουργία των δωματίων. Το πρόγραμμα λειτουργίας φυσικά εναπόκειται σε κάποιους περιορισμούς που είναι η διάρκεια κάθε ραντεβού.
- Το σύστημα κρατήσεων θα πρέπει να επιτρέπει την δημιουργία, τροποποίηση ή και διαγραφή κρατήσεων. Πιο συγκεκριμένα θα μπορούν να δημιουργηθούν νέες κρατήσεις σε κάποιο υφιστάμενο δωμάτιο ή να τροποποιηθεί μία κράτηση και να οριστούν τα νέα κατάλληλα χαρακτηριστικά που πρέπει να την συνοδεύουν όπως ημερομηνία κράτησης, άτομα και τα στοιχεία επικοινωνίας του ατόμου που

πραγματοποιεί την κράτηση. Μέσα στο πλαίσιο της τροποποίησης τους θα δύναται να μετατοπίσει το ραντεβού μόνο κάποιος χρήστης της διαχειριστικής σελίδας βάση της διαθεσιμότητας. Ο τελικός χρήστης δεν θα έχει αυτόνομα τη δυνατότητα τροποποίησης της κράτησης για να αποφευχθούν προβλήματα συχνών ακυρώσεων.

- Το σύστημα κρατήσεων θα πρέπει να ενημερώνει τους χρήστες αλλά και τον διαχειριστή του δωματίου όταν γίνεται μία αλλαγή στις κρατήσεις του δωματίου. Πιο συγκεκριμένα όταν ένας χρήστης πραγματοποιήσει μία κράτηση και αυτή επιβεβαιωθεί από το σύστημα, θα πρέπει να αποσταλεί ενημερωτικό μήνυμα τόσο στον χρήστη που πραγματοποίησε την κράτηση όσο και στον διαχειριστή για τη νέα αυτή κράτηση. Αντίστοιχη ενημέρωση θα πρέπει να λάβουν και όταν γίνει κάποια τροποποίηση σε μία υφιστάμενη κράτηση ή διαγραφή της.

Οι μη λειτουργικές απαιτήσεις, όπως υπονοεί το όνομα τους, είναι απαιτήσεις που δεν αφορούν άμεσα τις συγκεκριμένες λειτουργίες που παρέχονται από το σύστημα. Μπορεί να σχετίζονται με τις ανακύπτουσες ιδιότητες του συστήματος, όπως η αξιοπιστία, ο χρόνος απόκρισης και ο αποθηκευτικός χώρος. Πιο αναλυτικά μπορεί να καθορίζουν την απόδοση, την προστασία από εξωτερικούς κινδύνους, τη διαθεσιμότητα και άλλες ανακύπτουσες ιδιότητες του συστήματος. Οι μη λειτουργικές απαιτήσεις προκύπτουν από τις ανάγκες των χρηστών, από περιορισμούς του προϋπολογισμού, από πολιτικές της εταιρείας, από την ανάγκη διαλειτουργικότητας με άλλα συστήματα λογισμικού ή υλικού, ή από εξωτερικούς παράγοντες όπως οι κανονισμοί προστασίας ή η νομοθεσία περί προσωπικού απορρήτου. Ένα επιπλέον στοιχείο που πρέπει να αναφέρουμε εδώ είναι ότι οι μη λειτουργικές απαιτήσεις συχνά έρχονται σε αντίθεση και αλληλεπιδρούν με άλλες λειτουργικές ή μη λειτουργικές απαιτήσεις. Οι τύποι των μη λειτουργικών απαιτήσεων είναι οι εξής:

- Απαιτήσεις προϊόντος
- Εταιρικές απαιτήσεις
- Εξωτερικές απαιτήσεις

Μη λειτουργικές απαιτήσεις

- Η προβολή των διαθέσιμων δωματίων και ωραρίων θα πρέπει να παρέχεται μέσω ενός widget που θα χρησιμοποιεί τεχνολογίες που επιτρέπουν την συμβατότητα με κάθε υπολογιστική μονάδα και τεχνολογία ανάπτυξης της εκάστοτε σελίδας προβολής της επιχείρησης. Πιο συγκεκριμένα θα γίνει χρήση HTML, CSS και Javascript

τεχνολογιών, που αντιλαμβάνεται κάθε περιηγητής αλλά και μπορεί να προσαρμοστεί εύκολα σε κάθε είδους σελίδα προβολής όπως CMS ιστοσελίδες. Στο παραπάνω περιορισμό μπορούμε να προσθέσουμε και τις ελάχιστες δυνατές απαιτήσεις μιας υπολογιστικής μονάδας για να τρέξει ένας περιηγητή.

- Εν συνεχεία της χρήσης περιηγητή θα πρέπει να διασφαλιστεί η τήρηση χρήσης συγκεκριμένων πρωτοκόλλων ασφαλείας για εύρυθμη και αξιόπιστη λειτουργία του συστήματος, τέτοια πρωτόκολλα είναι η χρήση HTTPS, CSP και X-XSS-Protection.
- Το σύστημα κρατήσεων μας εφόσον λαμβάνει, διαχειρίζεται και φυλάσσει προσωπικά δεδομένα των χρηστών οφείλει σύμφωνα με την ευρωπαϊκή νομοθεσία να τηρεί κάποιους κανόνες για αυτά τα δεδομένα και την ασφάλεια τους. Ο νέος Γενικός Κανονισμός Προστασίας Δεδομένων (GDPR - 2016/679) της ΕΕ αποτελεί ένα πλαίσιο που καθορίζουν εξ αρχής τον σχεδιασμό της εφαρμογής με τρόπο τέτοιο ώστε να προστατεύουν τα προσωπικά δεδομένα των χρηστών καθώς και αυτοί να έχουν τη δυνατότητα της διαγραφής τους όποτε ζητηθεί. Επιπρόσθετα θα πρέπει να προστατεύει τα δεδομένα των χρηστών από άτομα μη εξουσιοδοτημένης πρόσβασης.
- Το σύστημα θα πρέπει να είναι σε θέση να διαχειρίζεται τυχόν αυξανόμενο αριθμό κρατήσεων καθώς η επιχείρηση αναπτύσσεται. Δηλαδή να έχει γίνει η απαραίτητη πρόβλεψη έτσι ώστε να μπορεί αν χρειαστεί να επεκταθεί και να υποστηρίξει μεγάλο όγκο χρηστών.
- Το σύστημα θα πρέπει να είναι εύχρηστο, με σαφές και διαισθητικό περιβάλλον εργασίας.
- Το σύστημα θα πρέπει να είναι διαθέσιμο και λειτουργικό ανά πάσα στιγμή, με υψηλό ποσοστό διαθεσιμότητας.
- Το σύστημα θα πρέπει να προστατεύει τους χρήστες κατά την εισαγωγή δεδομένων στο σύστημα όπως για παράδειγμα συμπλήρωση φορμών. Με αυτόν τον τρόπο δεν θα εισάγουν λανθασμένα δεδομένα και παράλληλα το σύστημα θα ενημερώνει τους χρήστες με κατάλληλα μηνύματα για ενδεχόμενα λάθη.

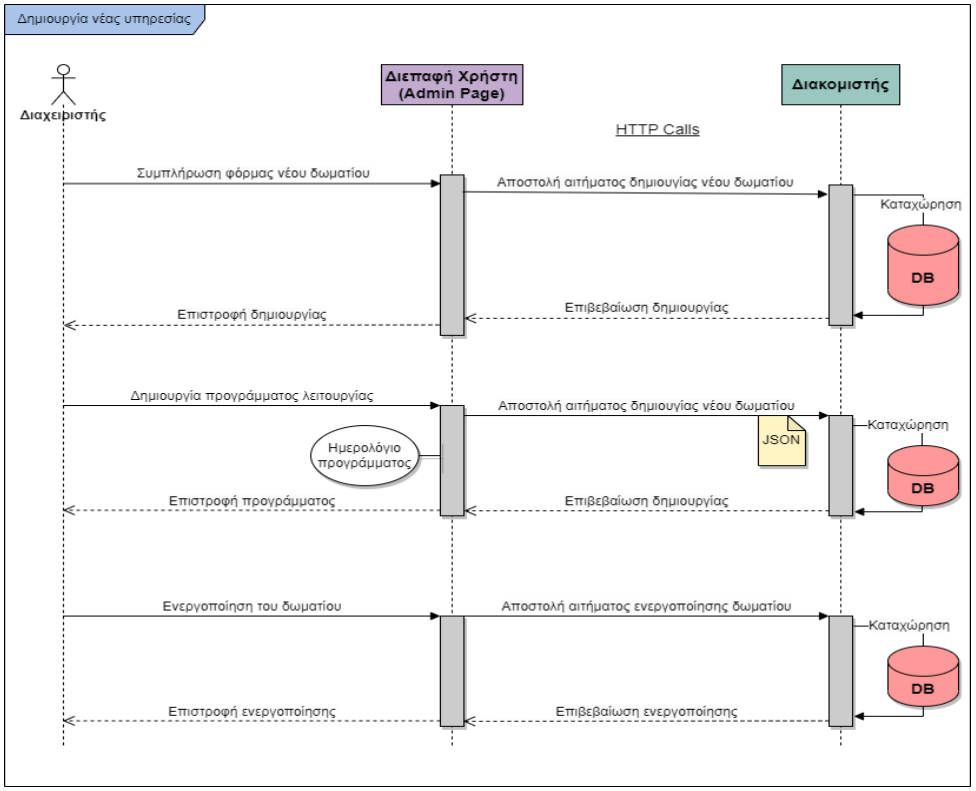
4.3 Απαιτήσεις Συστήματος

Οι απαιτήσεις συστήματος είναι επεκτάσεις των απαιτήσεων χρήστη οι οποίες χρησιμοποιούνται από τους μηχανικούς λογισμικού ώς σημείο εκκίνησης για το σχεδιασμό του συστήματος. Περιλαμβάνουν επιπλέον λεπτομέρειες και επεξηγούν πως το σύστημα θα πρέπει να παρέχει τις απαιτήσεις χρήστη. Αν εξετάσουμε τις απαιτήσεις από την πλευρά του συστήματος μπορέσουμε να βγάλουμε στην επιφάνεια τα σημεία που είναι απαραίτητα για την υλοποίηση της κάθε λειτουργίας του συστήματος μας (λειτουργικές και μη λειτουργικές)

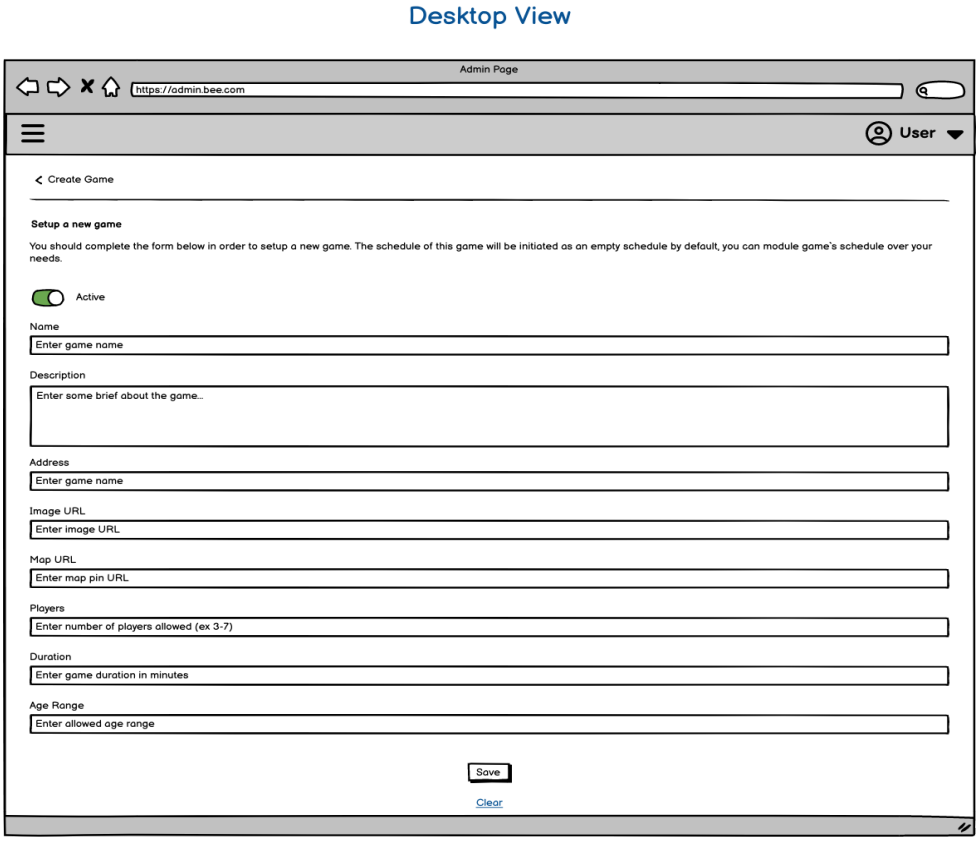
έτσι ώστε να ικανοποιούνται οι απαιτήσεις χρήστη και κατ' επέκταση και συστήματος ακολουθώντας φυσικά και το διάγραμμα περιπτώσεων χρήσης (εικόνα 4.3).

Δημιουργία και διαχείριση υπηρεσιών

Η βασικότερη λειτουργία που αποτελεί και τη θεμελιώδη λειτουργία του συστήματος κρατήσεων μας είναι οι υπηρεσίες που προσφέρονται και στην προκειμένη περίπτωση είναι δωμάτια απόδρασης. Η δημιουργία και διαχείριση των δωματίων θα γίνεται μέσω της διαχειριστικής σελίδας που θα έχουν πρόσβαση μόνο διαπιστευμένοι χρήστες με τα κατάλληλα δικαιώματα. Πιο συγκεκριμένα οι χρήστες που θα μπορούν να προβούν σε τέτοιες ενέργειες θα είναι ο διαχειριστής (Manager), αυτός θα πρέπει να δημιουργεί ένα δωμάτιο εξ αρχής, να το ενεργοποιεί ή απενεργοποιεί, να τροποποιεί τα χαρακτηριστικά του καθώς και να εφαρμόζει σε αυτό ένα ή περισσότερα προγράμματα λειτουργίας. Ξεκινώντας να αναλύσουμε αυτά τα χαρακτηριστικά, ο διαχειριστής θα δημιουργήσει ένα νέο δωμάτιο μέσω μιας φόρμας (εικόνες 4.5, 4.6) στη διαχειριστική σελίδα του συστήματος, συμπληρώνοντας τα δεδομένα που είναι απαραίτητα για την δημιουργία ενός νέου δωματίου τα οποία είναι η ονομασία του, η διάρκεια του σε λεπτά, υποστηριζόμενος αριθμός παικτών κλπ. Αφού συμπληρωθούν τα απαραίτητα πεδία της φόρμας θα αποσταλεί το αίτημα προς διεκπεραίωση από το αντίστοιχο API του διακομιστή. Το API θα αναλάβει την διεκπεραίωση του αιτήματος και θα απαντήσει επιτυχώς εφόσον όλα έχουν πάει καλά ή με κάποιο μήνυμα λάθους σε διαφορετική περίπτωση. Θεωρούμε ότι η διεκπεραίωση του αιτήματος ήταν επιτυχής. Η διαχειριστική σελίδα αναλαμβάνει να παρουσιάσει αυτές τις αλλαγές στον χρήστη με κατάλληλα μηνύματα και ενέργειες και να έχουμε το αποτέλεσμα στην οθόνη του χρήστη. Στην συνέχεια ο διαχειριστής καλείται να φτιάξει ένα πρόγραμμα λειτουργίας για το δωμάτιο ώστε να έχει υπόσταση και να μπορεί να γίνει ανοιχτό προς το κοινό. Έπειτα από την δημιουργία του προγράμματος μέσα από το εργαλείο που παρέχεται για εύκολη διαχείριση και δημιουργια ωραρίων πάνω σε έναν ημερολογιακό καμβά, ακολουθείται η παραπάνω διαδικασία αποστολής του αιτήματος για αποθήκευση του προγράμματος από το αντίστοιχο API. Κατόπιν και αυτής της επιτυχημένης ενέργειας ο διαχειριστής όταν κρίνει πλέον ότι η επιχείρηση είναι έτοιμη να διαθέσει το δωμάτιο προς το κοινό ενεργοποιεί το δωμάτιο μέσω ενός κουμπιού διακόπτη, αφού πρώτα λάβει την θετική απάντηση από τον διακομιστή για το αίτημα του. Έτσι το δωμάτιο πλέον είναι προσβάσιμο διαδικτυακά προς το κοινό και έτοιμο να δεχτεί κρατήσεις. Παρακάτω (εικόνα 4.4)παρουσιάζονται οι αλληλεπιδράσεις του χρήστη με το σύστημα για την δημιουργία/διαχείριση ενός δωματίου σε διάγραμμα ακολουθίας.



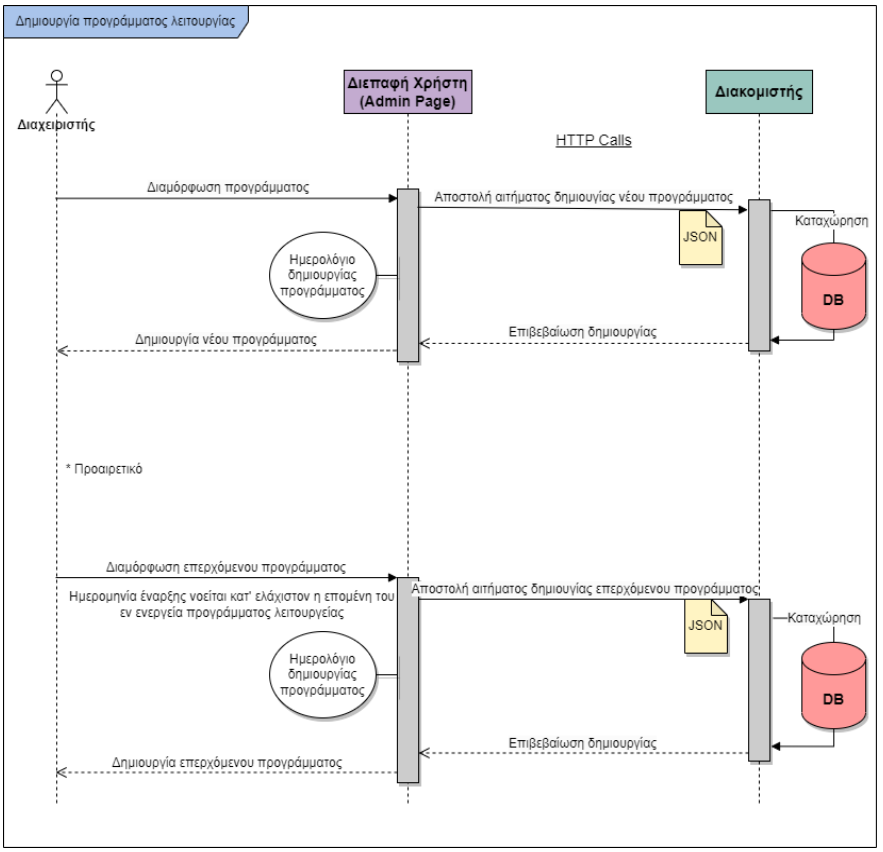
Εικόνα 4.4 Διάγραμμα ακολουθίας για την δημιουργία υπηρεσίας



Εικόνα 4.5 Προσχέδιο φόρμας για τη δημιουργία ενός νέου δωματίου (Desktop view)

Δημιουργία και διαχείριση ωραρίων

Η διαμόρφωση των ωραρίων λειτουργίας των δωματίων που θα υποστηρίζει το σύστημα κρατήσεων μας είναι μία εξίσου σημαντική λειτουργία καθώς είναι ο τρόπος να οργανώνει, διαχωρίζει και διαθέτει τα ωράρια στο κοινό. Πέραν της δημιουργίας ενός ωραρίου ή και της επεξεργασίας του υφιστάμενου ωραρίου εξίσου σημαντική κρίνεται και η ύπαρξη ωραρίων σειριακής ακολουθίας ώστε να καθορίζουν τα ωράρια βάση των αναγκών της επιχείρησης αλλά και της ζήτησης από το κοινό. Η δημιουργία και η διαχείριση των ωραρίων λειτουργίας του συστήματος κρατήσεων θα γίνεται μέσω της διαχειριστικής σελίδας και θα μπορεί να την πραγματοποιήσει ο διαχειριστής των δωματίων. Μέσω ενός κατάλληλα διαμορφωμένου ημερολογιακού καμβά θα μπορεί ο διαχειριστής να ορίσει το προκαθορισμένο εβδομαδιαίο ωράριο ενός δωματίου σύμφωνα με τις ανάγκες του. Το ωράριο θα εναπόκειται σε κάποιους περιορισμούς όπως η διάρκεια ενός ραντεβού, δύο ραντεβού δεν θα μπορούν να είναι το ένα μετά το άλλο χωρίς χρονικό κενό. Βάση αυτών των περιορισμών θα δημιουργείται, θα διαμορφώνεται και θα αποστέλλεται ένα αίτημα στον διακομιστή του συστήματος για την διεκπεραίωση του αιτήματος μας. Κατόπιν θετικής απάντησης από το αντίστοιχο API θα μπορούν να διατίθενται αυτά τα ωράρια στο κοινό για την πραγματοποιήση κρατήσεων.



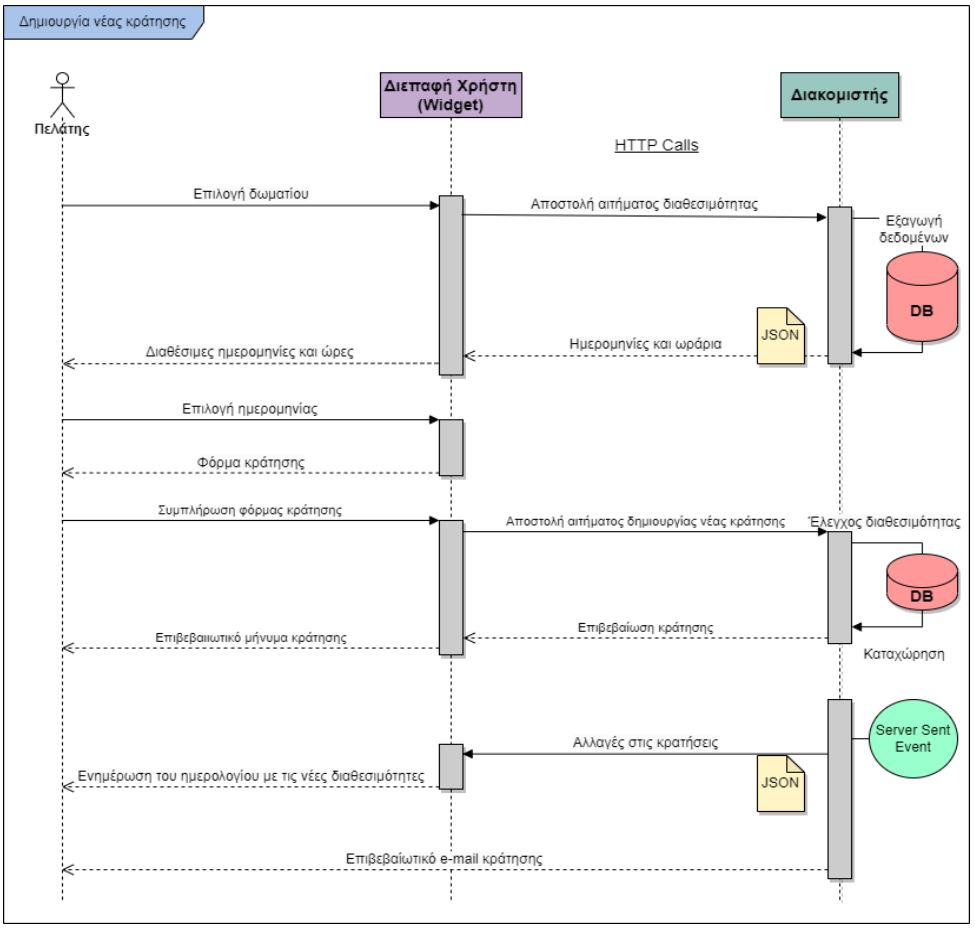
Εικόνα 4.6 Διάγραμμα ακολουθίας για την δημιουργία ωραρίων

Όπως βλέπουμε και στο διάγραμμα ακολουθίας 4.6 παρακάτω το επόμενο βήμα είναι προαιρετικό αλλά πολύ σημαντικό για τις επιχειρήσεις καθώς αποτελεί ένα σοβαρό πρόβλημα που αντιμετωπίζουν τα συστήματα κρατήσεων μαζί με την αλλαγή των ωραρίων. Έπειτα από μελέτη για την επίλυση αυτού του προβλήματος σχεδιάστηκε η χρήση μελλοντικών σειριακών προγραμμάτων λειτουργίας ενός δωματίου. Με αυτόν τον τρόπο ο διαχειριστής θα μπορεί να ορίζει το επερχόμενο πρόγραμμα λειτουργίας χωρίς να δημιουργούνται συγκρουόμενα ωράρια με αποτέλεσμα τα συχνά λάθη, έλλειψη ελέγχου και δυσαρέσκεια των πελατών. Όσων αφορά την δημιουργία και διαχείριση του επερχόμενου ωραρίου δεν διαφέρει σε τίποτα από την παραπάνω διαδικασία με την μόνη διαφορά τον καθορισμό της ημερομηνίας έναρξης ισχύος του νέου ωραρίου στο δωμάτιο. Η ημερομηνία αυτή θα πρέπει τουλάχιστον να είναι η επόμενη μέρα από την δημιουργία του επερχόμενου ωραρίου.

Δημιουργία και διαχείριση κρατήσεων

Η δημιουργία και η διαχείριση κρατήσεων αποτελεί το αντικείμενο ολόκληρου του συστήματος μας, καθώς είναι ο βασικός σκοπός της δημιουργίας του. Οι κρατήσεις μπορούν να πραγματοποιηθούν από τριών ειδών χρήστες, αυτοί είναι οι πελάτες, οι τρίτοι πάροχοι και οι διαχειριστές ενός δωματίου. Η πρώτη κατηγορία που θα εξετάσουμε είναι οι πελάτες από τους οποίους θα πραγματοποιείται και ο μεγαλύτερος όγκος κρατήσεων. Για την πραγματοποιήσει των κρατήσεων από τους πελάτες κατασκευάστηκε μία διεπαφή πελάτη (θα την αποκαλούμε στο εξής widget) για να παρέχει την ευκολία της προσαρμογής και τοποθέτησης σε διαφόρων ειδών ιστοσελίδες. Μέσω αυτής ο πελάτης θα πλοηγείται και θα μπορεί να δει τα διαθέσιμα δωμάτια ψυχαγωγίας της εκάστοτε επιχείρησης. Αφού επιλέξει ποιο δωμάτιο επιθυμεί πραγματοποιείται μια HTTP επικοινωνία με το αντίστοιχο API ώστε να φέρει τα απαραίτητα δεδομένα για αυτό το δωμάτιο, που είναι οι διαθέσιμες ημέρες και ώρες που είναι ελεύθερες. Στην συνέχεια ο πελάτης επιλέγει την ημερομηνία που επιθυμεί και το widget του σερβίρει μια φόρμα προς συμπλήρωση ώστε να συλλεξει το σύστημα μας τα απαραίτητα στοιχεία για την κράτηση. Τα βασικότερα στοιχεία είναι το ονοματεπώνυμο, ο αριθμός των παικτών, το κινητό τους τηλέφωνο και το ηλεκτρονικό ταχυδρομείο για περαιτέρω επικοινωνία και ενημέρωση σχετικά με την κράτηση. Στην συνέχεια αυτά τα δεδομένα αποστέλλονται στον διακομιστή για επεξεργασία, καταχώρηση και τελικά την επιβεβαίωση της κράτησης. Το μήνυμα της επιβεβαίωσης στέλνεται από τον διακομιστή στο widget το οποίο παρουσιάζει στον πελάτη το αποτέλεσμα της ενέργειας του. Παράλληλα με αυτά τα βήματα ενεργοποιούνται και 2 άλλες λειτουργίες του συστήματος μας που είναι η ενημέρωση του πελάτη με επιβεβαιωτικό μήνυμα στο ηλεκτρονικό ταχυδρομείο που

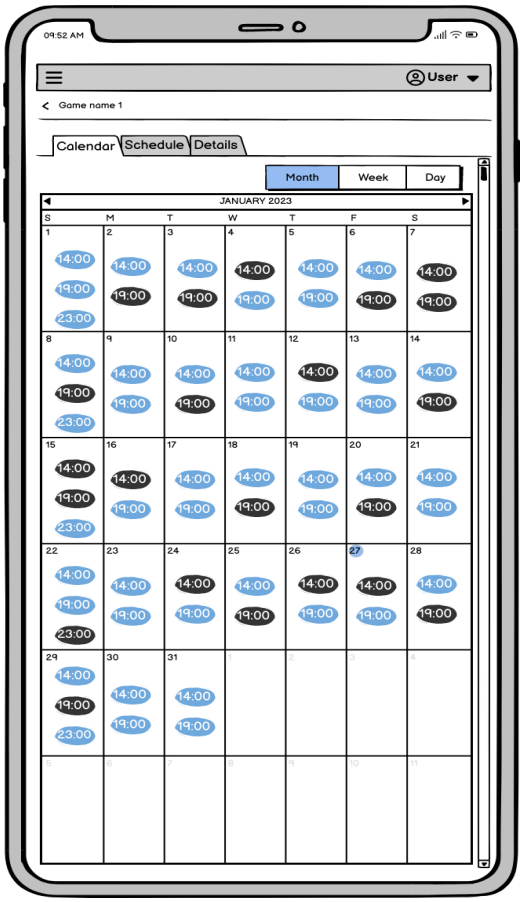
καταχώρησε και τις λεπτομέρειες σχετικά με το δωμάτιο ψυχαγωγίας που επέλεξε. Η δεύτερη λειτουργία είναι η ενημέρωση όλων όσων “ακούνε” τον διακομιστή μας και συνεπώς την βάση δεδομένων του συστήματος μας για τυχόν αλλαγές στις κρατήσεις. Αυτό επιτυγχάνεται με την μέθοδο του Server Sent Event και θα αναλυθεί εκτενέστερα σε παρακάτω κεφάλαιο, η λειτουργία όμως αυτή μας εξασφαλίζει την άμεση ενημέρωση όλων όσων κοιτούν το συγκεκριμένο δωμάτιο και την έγκυρη παροχή δεδομένων σχετικά με την διαθεσιμότητα του δωματίου. Όλες αυτές οι ενέργειες έχουν αποτυπωθεί και για τους τρεις τύπους χρηστών στα παρακάτω διαγράμματα ακολουθίας 4.7 και 4.8.



Εικόνα 4.7 Διάγραμμα ακολουθίας για την δημιουργία κράτησης από πελάτη

Η ίδια λογική ακολουθείται και στην δεύτερη περίπτωση όπου κράτηση πραγματοποιεί ο διαχειριστής του δωματίου που επί της ουσίας πραγματοποιεί μία κράτηση εξ ονόματος ενός πελάτη (τηλεφωνική κράτηση) με την μόνη διαφορά ότι αντί για χρήση του widget γίνεται χρήση της διαχειριστικής σελίδας. Στον διαχειριστή επίσης δίνεται η δυνατότητα της επεξεργασίας των κρατήσεων όσων αφορά την ημερομηνία και τα λοιπά στοιχεία.

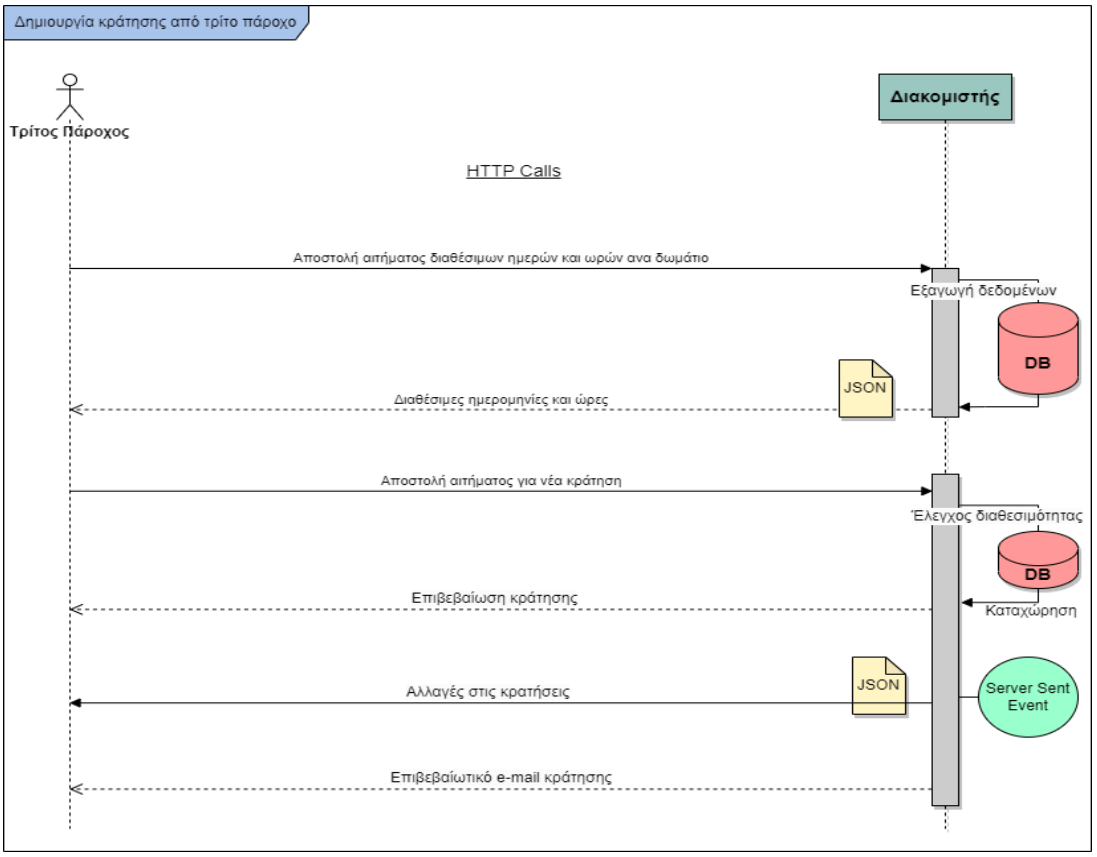
Επιλέχθηκε αυτή η δυνατότητα να μην δοθεί στους πελάτες για αποφυγεί συχνών ακυρώσεων και αλλαγών, μόνον οι διαχειριστές του δωματίου θα κάνουν χρήση της επεξεργασίας εφόσον αυτή είναι απαραίτητη. Η επεξεργασία δεν διαφέρει σε πολλά με την δημιουργία μιας κράτησης, γίνεται χρήση των HTTP μεθόδων και των κατάλληλων APIs. Παρέχεται επίσης ένα ειδικά διαμορφωμένο ημερολόγιο με πολλές δυνατότητες για την παρουσιαση στον διαχειριστή του δωματίου των κρατησεων του ώστε να κάνει καλύτερη διαχείριση τους με την μικρότερη δυνατή προσπάθεια.



Εικόνα 4.8 Σχεδιασμός ημερολογίου κρατήσεων διαχειριστικής σελίδας

Η τρίτη και τελευταία περίπτωση δημιουργίας κράτησεων είναι από τρίτους παρόχους. Τρίτοι πάροχοι θα έχουν την δυνατότητα να κοιτούν τα διαθέσιμα δωμάτια και μαζί με τις διαθεσιμότητες να πραγματοποιούν κρατήσεις. Οι ενέργειες αυτές και η επικοινωνία των δύο συστημάτων θα πραγματοποιείται μέσω των αντίστοιχων APIs τα οποία θα γίνονται διαθέσιμα προς συνεργαζόμενους τρίτους παρόχους μέσω ενός αναγνωριστικού (Token). Η λειτουργία αυτή θα δίνει την δυνατότητα στο σύστημα κρατήσεων μας να προβάλλεται και σε άλλες πλατφόρμες ώστε να αποκομίζει από εκεί οφείλοι για την δημοτικότητα της επιχείρησης

και την περαιτέρω προσέλκυση πελατών. Η επικοινωνία των δύο συστημάτων θα γίνεται μέσω των APIs και χρήση JSON για την μεταφορά δεδομένων και στην συνέχεια οι τρίτοι πάροχοι θα αναλαμβάνουν αυτόνομα την διαδικασία παρουσίασης της πληροφορίας για την συγκεκριμένη επιχείρηση στο σύστημα τους. Η διαδικασία επικοινωνίας δεν διαφέρει σε τίποτα με την παραπάνω που περιγράφηκε και αποτυπώνεται στο παρακάτω διάγραμμα.

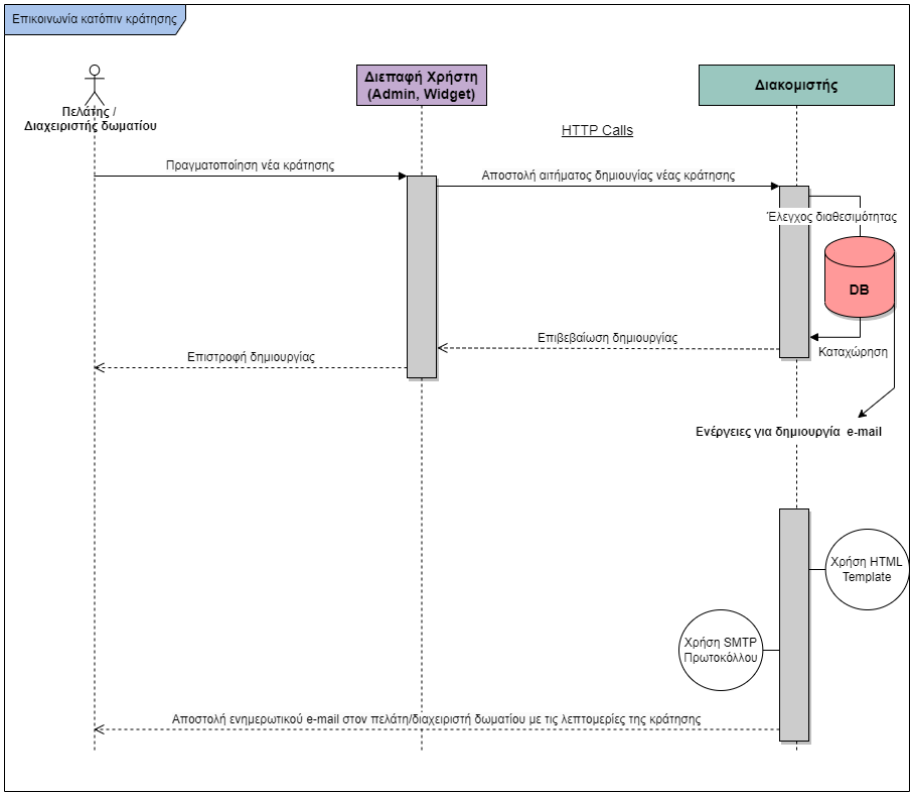


Εικόνα 4.9 Διάγραμμα ακολουθίας για την δημιουργία κράτησης από τρίτο πάροχο

Αποστολή αυτόματων ενημερωτικών μηνυμάτων

Η επικοινωνία του συστήματος κρατήσεων μας με τους χρήστες του είναι μία πολύ σημαντική και βασική λειτουργία καθώς προσωποποιεί μία διαδικασία αλλά και παρέχει άμεση ενημέρωση για την εξέλιξη των αιτημάτων των χρηστών. Όπως βλέπουμε και στο διάγραμμα ακολουθίας 4.9 και 4.10 η επικοινωνία του συστήματος με τους πελάτες και τους διαχειριστές των δωματίων γίνεται μέσω ηλεκτρονικού ταχυδρομείου. Κατόπιν μιας επιτυχημένης κράτησης ενεργοποιείται αυτόματα και η λειτουργία της επικοινωνίας τόσο με τον πελάτη αλλά και τον διαχειριστή του δωματίου. Το σύστημα μας σε επίπεδο διακομιστή πραγματοποιεί ενέργειες για την παρασκευεί των emails μέσω ενός προσχεδιασμένου

προτύπου (HTML template) αρχικά για την περίπτωση νέων κρατήσεων αντλώντας τα απαραίτητα δεδομένα για την συγκεκριμένη κράτηση από την βάση δεδομένων. Στην συνέχεια μέσω της χρήσης του πρωτοκόλλου SMTP αποστέλλει τα ενημερωτικά emails στον πελάτη που πραγματοποίησε την κράτηση, πιο συγκεκριμένα στον λογαριασμό που καταχώρησε κατά την διάρκεια της κράτησης και στο ηλεκτρονικό ταχυδρομείο του διαχειριστή του δωματίου. Το περιεχόμενο αυτών των emails είναι καθαρά ενημερωτικό και παρέχει τις πληροφορίες που αφορούν τις λεπτομέρειες της κράτησης όπως ημερομηνία, αριθμός παικτών, τηλέφωνο επικοινωνίας του χρήστη κλπ, καθώς και τηλέφωνα επικοινωνίας της επιχείρησης αν απαιτηθεί περαιτέρω επικοινωνία. Από πλευράς διαχειριστή οι πληροφορίες αφορούν τη νέα κράτηση για την άμεση ενημέρωση του ώστε να υπάρχει καλύτερη οργάνωση του χρόνου για την προετοιμασία υποδοχής της νέας ομάδας στο δωμάτιο ψυχαγωγίας.



Εικόνα 4.10 Διάγραμμα ακολουθίας για την αποστολή email κράτησης

Στην περίπτωση που μία κράτηση για κάποιο λόγο τροποποιηθεί ή ακυρωθεί αποστέλλονται επίσης ενημερωτικά emails για τον καλύτερο χειρισμό από πλευράς διαχειριστή αλλά και την ενημέρωση/υπενθύμιση του πελάτη για τις τελευταίες αλλαγές της κράτησης. Η διαδικασία δεν διαφέρει σε πολλά σημεία κατόπιν επιβεβαίωσης των αλλαγών από τον διακομιστή, είτε

πρόκειται για τροποποίηση της κράτησης είτε ακύρωση της, σε αυτές τις περιπτώσεις φυσικά χρησιμοποιείται διαφορετικό πρότυπο (HTML template) και εισάγονται τα απαραίτητα δεδομένα της κράτησης που τροποποιήθηκε, δηλαδή νέα ημερομηνία και ώρα, αλλαγή στον αριθμό των παικτών κλπ. Στην περίπτωση της ακύρωσης η οποία υπενθυμίζουμε ότι πραγματοποιείται μόνο από τον διαχειριστή του δωματίου αποστέλλεται ενημερωτικό μήνυμα στον πελάτη για την ακύρωση της κράτησης και ένα μήνυμα με σκοπό την μελλοντική επαναπροσεγγίσει του πελάτη από την επιχείρηση. [36]

Κεφάλαιο 5ο

Σχεδίαση και Υλοποίηση βάσης δεδομένων

5.1 Μοντελοποίηση βάσης δεδομένων

Η βάση δεδομένων που επιλέχθηκε για το σύστημα κρατήσεων είναι η PostgreSQL, με έκδοση 11.18, η οποία είναι μια σχεσιακή βάση και η αποθήκευση των δεδομένων γίνεται με τη μορφή πινάκων. Στις σχεσιακές βάσεις δεδομένων κάποια από τα δεδομένα των πινάκων υποδεικνύουν δεδομένα σε άλλους πίνακες και με αυτόν τον τρόπο οι πίνακες συσχετίζονται. Ο σχεδιασμός της βάσης δεδομένων είναι πολύ κρίσιμος για την ανάπτυξη της εφαρμογής και γι αυτό αποτελείται από κάποια στάδια όπως θα αναλύσουμε και παρακάτω όπως είναι η μοντελοποίηση, τα διαγράμματα και την υλοποίηση.

5.1.1 Οντότητες και γνωρίσματα

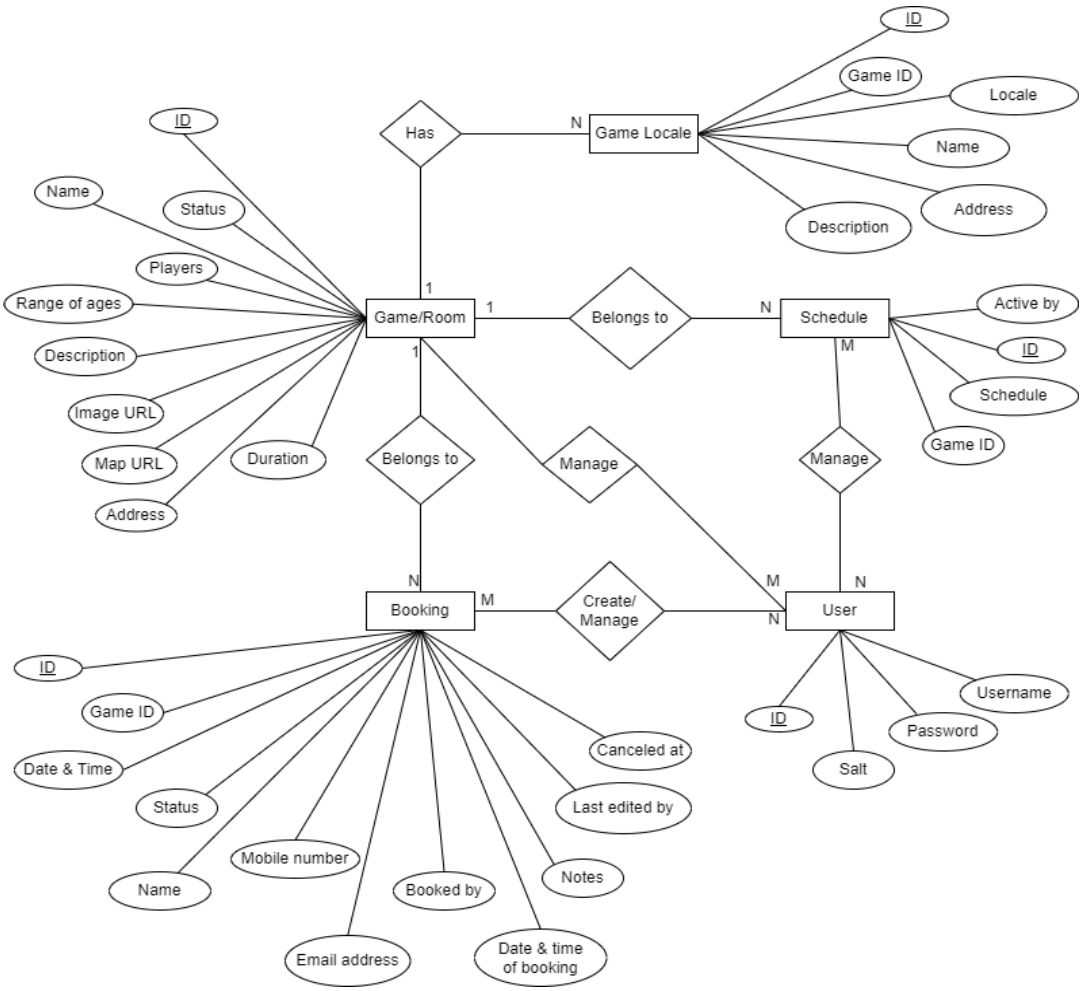
Η βάση δεδομένων του συστήματος περιλαμβάνει 5 βασικές οντότητες-πίνακες για την υλοποίηση του ελάχιστου βιώσιμου προϊόντος (MVP). Αυτές οι οντότητες είναι τα δωμάτια/παιχνίδια (πίνακας **gme**), οι γλώσσες για τα δωμάτια/παιχνίδια (πίνακας **gme_lcl**), το πρόγραμμα (πίνακας **schdl**), οι κρατήσεις (πίνακας **bkng**) και οι χρήστες του συστήματος (πινακας **usr**). Ξεκινώντας να αναλύουμε κάθε οντότητα της βάσης δεδομένων μας και των πεδίων τους θα εξετάσουμε τα δωμάτια/παιχνίδια (πίνακας **gme**). Για τα δωμάτια/παιχνίδια αποθηκεύονται δεδομένα όπως ο μοναδικός αριθμός του (gme_id), η κατάσταση του (status), το URL της εικόνας (img_url), το στίγμα του χάρτη (map_url), το εύρος των παικτών (plrs), η ελάχιστη επιτρεπόμενη ηλικία (age_rng), το όνομα του δωματίου/παιχνιδιού (nme), μια περιγραφή για το δωμάτιο/παιχνίδι (descr), η διεύθυνση (addr) και η διάρκεια του δωματίου/παιχνιδιού σε λεπτά (dur). Στην συνέχεια ο πίνακας που συνδέεται άμεσα με τον παραπάνω είναι οι υποστηριζόμενες γλώσσες των δωματίων (πίνακας **gme_lcl**). Για αυτόν τον πίνακα αποθηκεύονται δεδομένα όπως ο μοναδικός αριθμός του (gme_lcl_id), ο μοναδικός αριθμός του συνδεδεμένου δωματίου/παιχνιδιού (gme_id), η υποστηριζόμενη γλώσσα (lcl), το όνομα του δωματίου/παιχνιδιού (nme), η περιγραφή του (descr), η διεύθυνση (addr). Τα παραπάνω πεδία που επαναλαμβάνονται με τον πρώτο πίνακα

περιλαμβάνουν περιεχόμενο με την υποστηριζόμενη γλώσσα του πεδίου lcl. Στον επόμενο πίνακας της βάσης δεδομένων του συστήματος κρατήσεων είναι ένας πολύ κρίσιμος πίνακας και αυτός είναι το πρόγραμμα των δωματίων/παιχνιδιών (πίνακας **schdl**). Για αυτόν τον πίνακα αποθηκεύονται δεδομένα όπως ο μοναδικός αριθμός του (schdl_id), ο μοναδικός αριθμός του συνδεδεμένου δωματίου/παιχνιδιού (gme_id), το εβδομαδιαίο πρόγραμμα (schedule) και η ημερομηνία ενεργοποίησης (active_by). Στην συνέχεια έχουμε έναν εξίσου σημαντικό πίνακα που αποτελεί και την βασική ιδέα πάνω στην οποία στηρίχθηκε όλη η εφαρμογή και ο πίνακας αυτός είναι οι κρατήσεις των δωματίων/παιχνιδιών (πίνακας **bkng**). Για αυτόν τον πίνακα αποθηκεύονται δεδομένα όπως ο μοναδικός αριθμός του (bkng_id), ο μοναδικός αριθμός του συνδεδεμένου δωματίου/παιχνιδιού (gme_id), η κατάσταση της κράτησης (status), η ημερομηνία και ώρα της κράτησης (dte), το όνομα του παίκτη της κράτησης (nme), ο τηλεφωνικός αριθμός του παίκτη της κράτησης (mob_num), το ηλεκτρονικό ταχυδρομείο του παίκτη της κράτησης (email_addr), η ημερομηνία και ώρα που πραγματοποιήθηκε η κράτηση (tmstmp), η ημερομηνία και ώρα που μπορεί να πραγματοποιηθεί ακύρωση της κράτησης (canceled_at), σημειώσεις (notes), ποιος πραγματοποίησε τη κράτηση (bked_by), ποιος χρήστης του συστήματος έκανε τελευταίος αλλαγές στην υπάρχουσα κράτηση (lst_edted_by). Ο τελευταίος πίνακας της βάσης δεδομένων μας είναι οι χρήστες του συστήματος και οι πελάτες της εφαρμογής (πίνακας **usr**). Για αυτόν τον πίνακα αποθηκεύονται δεδομένα όπως ο μοναδικός αριθμός του (usr_id), το όνομα χρήση (usrnme), ο κωδικός του χρήστη (psswrd), το αναγνωριστικό-salt ασφαλείας του κωδικού (salt).

5.1.2 Διάγραμμα Οντοτήτων Συσχετίσεων

Το διάγραμμα Οντοτήτων Συσχετίσεων (ERD) είναι μια αφηρημένη και εννοιολογική μέθοδος αναπαράστασης δεδομένων και σχέσεων μεταξύ δεδομένων μιας βάσης δεδομένων. Εισήχθει από τον Peter Chen το 1976 σε μια δημοσίευση που θεωρείται σήμερα ανάμεσα στις σημαντικότερες μεθόδους και με μεγαλύτερο αντίκτυπο στην ανάπτυξη λογισμικού. Χρησιμοποιείται για το σχεδιασμό και τη μοντελοποίηση βάσεων δεδομένων με εννοιολογικό τρόπο, πριν από την πραγματική υλοποίηση της βάσης δεδομένων. Το μοντέλο ER αποτελείται από οντότητες (ή αντικείμενα) που αναπαρίστανται με ορθογώνια, και τις σχέσεις μεταξύ αυτών των οντοτήτων, που αναπαρίστανται με γραμμές που συνδέουν τα ορθογώνια. Οι σχέσεις μπορεί να είναι διαφόρων τύπων, όπως μία προς μία, μία προς πολλές και πολλές προς πολλές. Κάθε οντότητα περιγράφεται από ένα σύνολο χαρακτηριστικών που αναπαρίσταται με ένα οβάλ σχήμα και συνδέεται με το ορθογώνιο της οντότητας με μια γραμμή. Το μοντέλο ER χρησιμοποιείται για την αναπαράσταση της δομής των δεδομένων

και των σχέσεων μεταξύ διαφορετικών στοιχείων δεδομένων και αποτελεί χρήσιμο εργαλείο για τον σχεδιασμό και την κατανόηση των βάσεων δεδομένων. Το μοντέλο αυτό βοηθάει στην απεικόνιση αντικειμένων και συσχετίσεων και αποτελεί δημοφιλή μεθοδολογία ανάπτυξης του αρχικού σχεδιασμού της βάσης δεδομένων. Κάποια δεδομένα των πινάκων (συνήθως μοναδικοί αριθμοί) υποδεικνύουν δεδομένα σε άλλους πίνακες και με αυτόν τον τρόπο οι πίνακες συσχετίζονται μεταξύ τους. Η λογική αναπαράσταση αυτής της δομής αποδίδεται με το διάγραμμα Οντοτήτων - Συσχετίσεων. Στην παρακάτω εικόνα γίνεται παρουσίαση του διαγράμματος για την διαδικτυακή σελίδα που αναπτύξαμε και φαίνονται τα βασικά βήματα για την σχεδίαση της βάσης δεδομένων για τον σύστημα κρατήσεων. Τα βήματα αυτά συμπεριλαμβάνουν αρχικά, τον προσδιορισμό του σκοπού που θα έχει η βάση δεδομένων και ο προσδιορισμός των πινάκων και των πεδίων που χρειάστηκαν. Επιπλέον γίνεται καθορισμός των πρωτευόντων κλειδιών και οι συσχετίσεις μεταξύ των πινάκων. Κατόπιν του αρχικού σχεδιασμού υπάρχουν περιθώρια βελτίωσης και ανακατασκευής όπου απαιτηθεί ανάλογα με τις ανάγκες - απαιτήσεις που προκύπτουν κατά την αναπτυξη.



Εικόνα 5.1 Διάγραμμα Οντοτήτων - Συσχετίσεων αρχικού σχεδιασμού

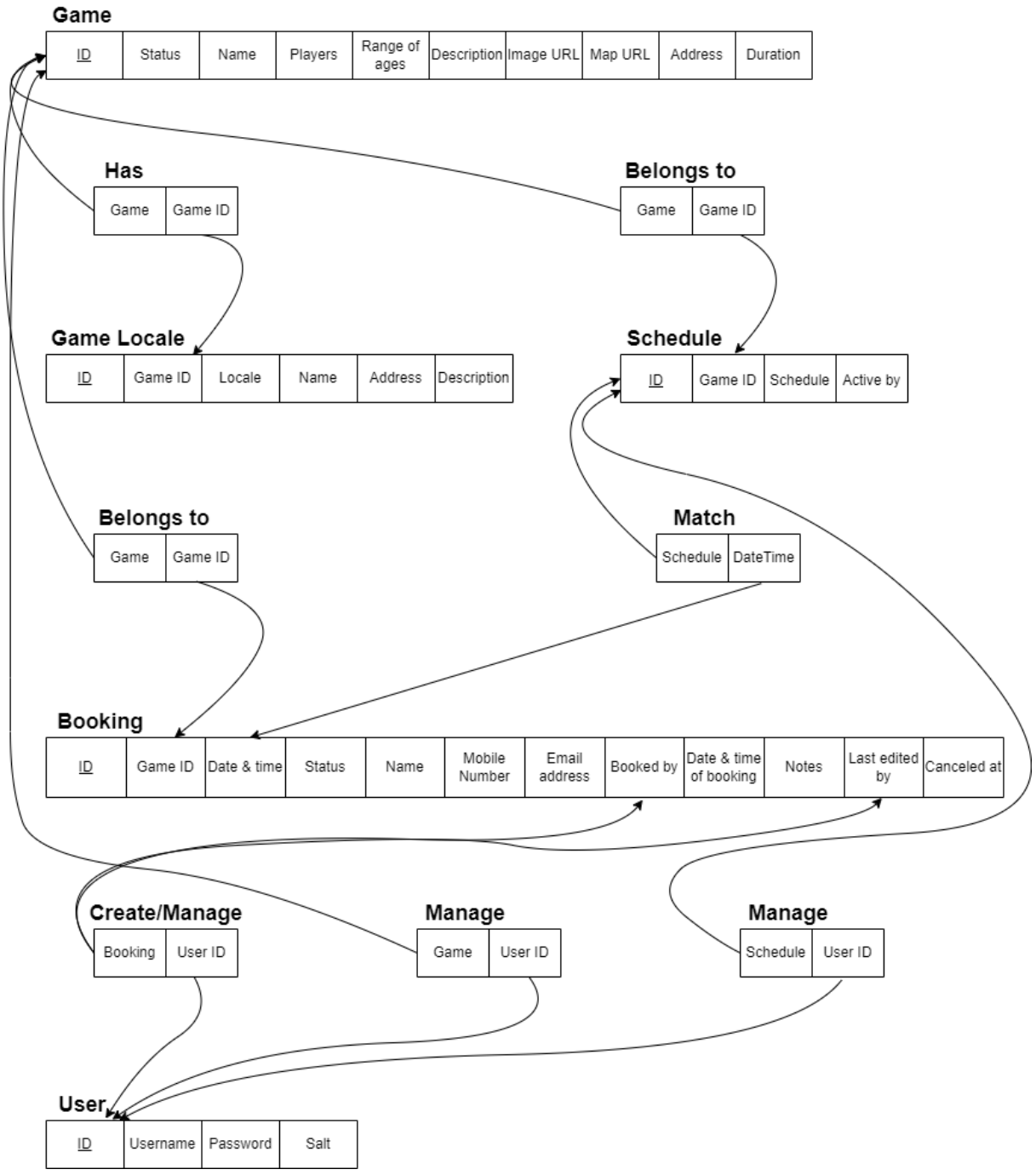
Όπως φαίνεται από την παραπάνω εικόνα όλες οι οντότητες βρίσκονται γύρω από την οντότητα θεμέλιο για τη βάση δεδομένων μας που είναι τα δωμάτια/παιχνίδια απόδρασης. Όλες οι υπόλοιπες οντότητες συσχετίζονται με αυτή με κάποιον τρόπο και έχουν σαν γνώρισμα τους, ώς ξένο κλειδί τον μοναδικό κωδικό του δωματίου. Οι οντότητες που διακρίνουμε είναι τα δωμάτια, η τοποθεσία δωματίου (πολύ-γλωσσικό), το πρόγραμμα, η κράτηση και ο χρήστης, και συνοδευονται από τα μοναδικά τους κλειδιά IDs. Η κάθε οντότητα πέραν από το πρωτεύον και ενδεχομένως ξένα κλειδιά έχει και τα υπόλοιπα γνωρίσματα που απαρτίζουν την οντότητα μας ώστε να συνθέσουν ένα ολοκληρωμένο αντικείμενο και εν συνεχεία πίνακα στη βάση του συστήματος κρατήσεων μας.

Κάθε δωμάτιο/παιχνίδι μπορεί να περιέχει ένα ή περισσότερες υποστηριζόμενες τοποθεσίες/γλώσσες (πολύ-γλωσσικό). Κάθε δωμάτιο επίσης θα μπορεί να έχει ένα ή περισσότερα προγράμματα λειτουργίας και αντίστοιχα μία ή περισσότερες κρατήσεις. Κάθε κράτηση πρέπει να ανήκει σε ένα δωμάτιο και σε ένα διαθέσιμο ωράριο του προγράμματος. Κάθε πρόγραμμα μπορεί να ανήκει σε ένα δωμάτιο και να αντιστοιχίζεται αν υπάρχει μόνο με μία κράτηση την ίδια ώρα και ημέρα. Όσον αφορά το πρόγραμμα θα πρέπει να επισημάνουμε εδώ ότι σχεδιάστηκε ως εβδομαδιαίο πρόγραμμα που θα επαναλαμβάνεται και θα διαχωρίζεται κάθε ραντεβού από την μοναδικότητα της ημερομηνίας και ώρας συνδυαστικά. Οι χρήστες του συστήματος ανάλογα με το ρόλο τους διαχωρίζονται από θετικούς και αρνητικούς μοναδικούς αριθμούς (θετικοί οι διαχειριστές συστήματος και αρνητικοί οι χρήστες ως πελάτες που πραγματοποιούν κρατήσεις για τον εαυτό τους). Οι χρήστες θα μπορούν να διαχειρίζονται τις υπόλοιπες οντότητες ώς διαχειριστές και να δημιουργούν κρατήσεις ως πελάτες.

5.1.3 Σχεσιακό Διάγραμμα

Από το παρακάτω σχεσιακό διάγραμμα προκύπτει ότι ο πίνακας πολύ-γλωσσικό παιχνίδι (Game locale) έχει σαν ξένο κλειδί το κύριο κλειδί του δωματίου/παιχνιδιού, αλλά και ότι το πρόγραμμα επίσης συσχετίζεται-ανήκει σε ένα δωμάτιο/παιχνίδι μέσω του ξένου κλειδιού game id. Ο πίνακας δωμάτιο/παιχνίδι δεν έχει κάποια εξάρτηση από τους υπόλοιπους πίνακες καθώς είναι η βάση πάνω στην οποία δομούνται όλοι οι υπόλοιποι, ο πολυ-γλωσσικός πίνακας βέβαια μπορεί να θεωρηθεί βοηθητικός καθώς μπορεί να περιέχει πολλές εγγραφές για ένα δωμάτιο/παιχνίδι. Όσον αφορά τη κράτηση μέσω των συνδέσεων παρατηρούμε ότι εναπόκειται στο πρόγραμμα ενός δωματίου/παιχνιδιού και φυσικά ότι συσχετίζεται μέσω του ξένου κλειδιού game id. Τέλος έχουμε το πίνακα χρήστη ο οποίος

σχετίζεται με την κράτηση όσον αφορά το ξένο κλειδί και ποιος πραγματοποίησε ή τροποποίησε τελευταίος μια κράτηση. Επιπλέον διαχειρίζεται τους πίνακες δωμάτιο/παιχνίδι και πρόγραμμα εφόσον είναι διαχειριστής συστήματος πράγμα που προκύπτει από το θετικό user id. Τα παραπάνω αποτυπώνονται στο σχεσιακό διάγραμμα της εφαρμογής μας με πίνακες και διασυνδέσεις.



Εικόνα 5.2 Σχεσιακό διάγραμμα πινάκων

5.2 Υλοποίηση Βάσης Δεδομένων

Η ανάπτυξη της βάσης δεδομένων μας όπως έχει αναφερθεί και παραπάνω έγινε με χρήση της PostgreSQL, καθώς αποτελεί μια από τις καλύτερες επιλογές για σχεσιακή βάση δεδομένων ώστε να υποστηρίξει τα υπάρχοντα και μελλοντικά χαρακτηριστικά του συστήματος κρατήσεων. Κατόπιν της δημιουργίας της βάσης δεδομένων, ξεκινήσαμε να δημιουργούμε τους πίνακες που σχεδιάστηκαν σε προηγούμενα βήματα ώστε να λάβει μορφή και υπόσταση. Ο πρώτος μας πίνακας είναι ο **gme** που θα αποθηκεύονται τα δωμάτια/παιχνίδια, Η δημιουργία του με SQL κώδικα φαίνεται στον κώδικα 5.1 του παραρτήματος και στην εικόνα 5.3 ένα στιγμιότυπο με τα περιεχόμενα του πίνακα. Ο πίνακας αποτελείται από το μοναδικό αριθμό **gme_id** που είναι και το πρωτεύον κλειδί, το **status** του δωματίου/παιχνιδιού, τα **img_url**, **map_url** όσον αφορά την εικόνα και την τοποθεσία του, τον αριθμό των παικτών **plrs**, το εύρος ηλικίας των παικτών **age_rng**, το όνομα του δωματίου/παιχνιδιού **nme**, μια περιγραφή descr, την διεύθυνση **addr** και τέλος την διάρκεια σε λεπτά **dur**.

Εικόνα 5.3 Στιγμιότυπο από μια εγγραφή στον πίνακα gme

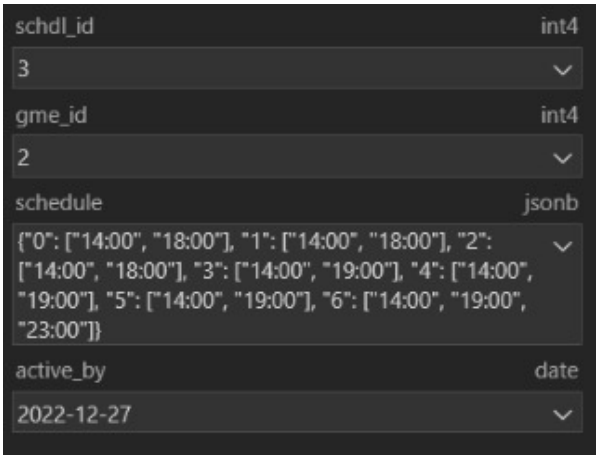
Ο επόμενος μας πίνακας είναι ο **gme_lcl** που θα αποθηκεύονται τα πολυγλωσσικά δεδομένα των δωματίων/παιχνιδιών, στον κώδικα 5.2 βλέπουμε την δημιουργία του με SQL κώδικα και στην εικόνα 5.4 ένα στιγμιότυπο με τα περιεχόμενα του πίνακα. Ο πίνακας αποτελείται από το μοναδικό αριθμό **gme_lcl_id** που είναι και το πρωτεύον κλειδί, το **gme_id** που είναι το ξένο κλειδί και ο συσχετισμός με τον παραπάνω πίνακα, η υποστηριζόμενη γλώσσα **lcl**, το όνομα του δωματίου/παιχνιδιού **nme**, μια περιγραφή descr και η διεύθυνση **addr**.

Εικόνα 5.4 Στιγμιότυπο από μια εγγραφή στον πίνακα gme_lcl

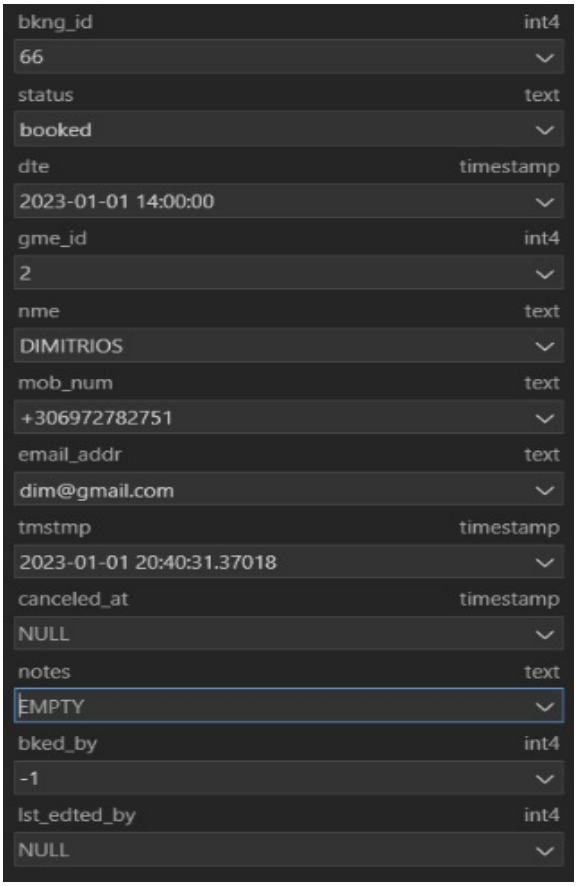
Ο επόμενος μας πίνακας είναι ο **schdl** που θα αποθηκεύονται τα προγράμματα λειτουργίας των δωματίων/παιχνιδιών, στον κώδικα 5.3 βλέπουμε την δημιουργία του με SQL κώδικα και στην εικόνα 5.5 ένα στιγμιότυπο με τα περιεχόμενα του πίνακα. Ο πίνακας αποτελείται από το μοναδικό αριθμό **schdl_id** που είναι και το πρωτεύον κλειδί, το **gme_id** που είναι το ξένο κλειδί και ο συσχετισμός με τον πίνακα **gme**, το πρόγραμμα λειτουργίας **schedule** και μια περιγραφή descr και η διεύθυνση **addr**.

Ο επόμενος μας πίνακας είναι ο **bkng** που θα αποθηκεύονται οι κρατήσεις των δωματίων/παιχνιδιών, στον κώδικα 5.4 βλέπουμε την δημιουργία του με SQL κώδικα και στην εικόνα 5.6 ένα στιγμιότυπο με τα περιεχόμενα του πίνακα. Ο πίνακας αποτελείται από το μοναδικό αριθμό **bkng_id** που είναι και το πρωτεύον κλειδί, το **gme_id** που είναι το ξένο κλειδί και ο συσχετισμός με τον πίνακα **gme**, το **status** της κράτησης, η ημερομηνία και ώρα της κράτησης dte, το όνομα του παίκτη που πραγματοποίησε τη κράτηση **nme,** το τηλεφωνικό αριθμό του παίκτη **mob_num,** το ηλεκτρονικό ταχυδρομείο του παίκτη **email_addr,** την ημέρα και ώρα που πραγματοποιήθηκε η κράτηση **tmstmp**, την ημέρα και ώρα που ενδεχομένως πραγματοποιήθηκε ακύρωση η κράτηση **canceled_at**, σημειώσεις για

την κράτηση **notes**, τον χρήστη από τον οποίο πραγματοποιήθηκε η κράτηση **bked_by** και τον χρήστη που έκανε τη τελευταία αλλαγή στη κράτηση **lst_edted_by**.

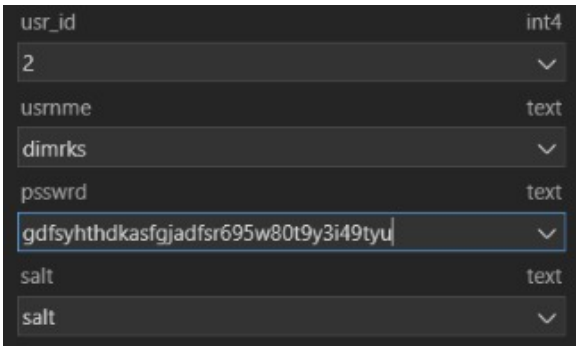


Εικόνα 5.5 Στιγμιότυπο από μια εγγραφή στον πίνακα schdl

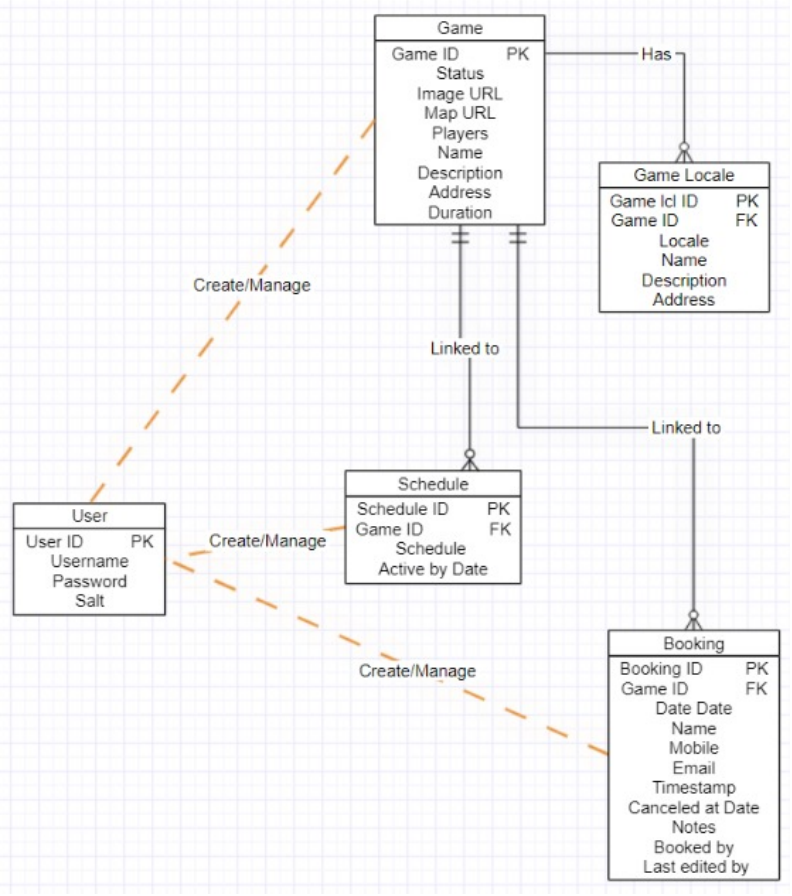


Εικόνα 5.6 Στιγμιότυπο από μια εγγραφή στον πίνακα bkng

Ο τελευταίος μας πίνακας είναι ο **usr** που θα αποθηκεύονται οι χρήστες της εφαρμογής μας (χρήστες συστήματος και χρήστες πελάτες), στον κώδικα 5.5 βλέπουμε την δημιουργία του με SQL κώδικα και στην εικόνα 5.7 ένα στιγμιότυπο με τα περιεχόμενα του πίνακα. Ο πίνακας αποτελείται από το μοναδικό αριθμό **usr_id** που είναι και το πρωτεύον κλειδί, το ονοματεπώνυμο του χρήστη **usrnme**, ο κωδικός εισόδου του χρήστη **psswrd** και το **salt** που βοηθά στην κρυπτογράφηση του κωδικού πρόσβασης.



Εικόνα 5.7 Στιγμιότυπο από μια εγγραφή στον πίνακα usr



Εικόνα 5.8 Διάγραμμα Συσχετίσεων Οντοτήτων της βάσης δεδομένων

Κεφάλαιο 6ο

Σχεδίαση και Υλοποίηση του Web service - Backend

Υπηρεσία Ιστού (Web Service) είναι ένα σύστημα λογισμικού που υποστηρίζει διαλειτουργική αλληλεπίδραση μηχανής με μηχανή μέσω ενός δικτύου. Έχει μια διεπαφή που περιγράφεται σε μορφή που μπορεί να επεξεργαστεί από μηχανή (συγκεκριμένα, Web Service Definition Language ή WSDL). Τα Web Service εκπληρώνουν μια συγκεκριμένη εργασία ή ένα σύνολο εργασιών. Ένα Web Service περιγράφεται χρησιμοποιώντας μια τυπική, επίσημη έννοια XML, που ονομάζεται περιγραφή της υπηρεσίας της, η οποία παρέχει όλες τις απαραίτητες λεπτομέρειες για την αλληλεπίδραση με την υπηρεσία, συμπεριλαμβανομένων των μορφών μηνυμάτων (που περιγράφουν λεπτομερώς τις λειτουργίες), των πρωτοκόλλων μεταφοράς και της τοποθεσίας. Η φύση της διεπαφής κρύβει τις λεπτομέρειες υλοποίησης της υπηρεσίας, ώστε να μπορεί να χρησιμοποιηθεί ανεξάρτητα από το υλικό ή την πλατφόρμα λογισμικού στην οποία υλοποιείται και ανεξάρτητα από τη γλώσσα προγραμματισμού στην οποία είναι γραμμένη. Αυτή η ανεξαρτησία επιτρέπει και ενθαρρύνει τις εφαρμογές που βασίζονται σε υπηρεσίες web να είναι χαλαρά συνδεδεμένες, προσανατολισμένες σε στοιχεία, υλοποιήσεις διασταυρούμενης τεχνολογίας. Οι υπηρεσίες Ιστού μπορούν να χρησιμοποιηθούν μόνες τους ή με άλλες υπηρεσίες Ιστού για την πραγματοποίηση μιας σύνθετης συγκέντρωσης ή μιας επιχειρηματικής συναλλαγής. Παρακάτω θα αναλύσουμε την σχεδίαση και την υλοποίηση του web service που χρειάζεται το σύστημα κρατήσεων ώστε να είναι λειτουργικό.

6.1 Σχεδίαση Web Service

6.1.1 Λειτουργίες Web Service

Από τις απαιτήσεις συστήματος προκύπτουν επτά βασικές υπηρεσίες προς αλληλεπίδραση με τους χρήστες. Οι υπηρεσίες αυτές είναι:

1. **Η δημιουργία, η ανάκτηση και επεξεργασία των χώρων ψυχαγωγίας** (Games).
2. **Η ανάκτηση και η επεξεργασία του ωραρίου για κάθε χώρο**.

3. **Η δημιουργία η ανάκτηση και η επεξεργασία των κρατήσεων καθώς και της διαθεσιμότητας για κάθε χώρο σε πραγματικό χρόνο**.
4. **Η δημιουργία, η ανάκτηση και επεξεργασία των χρηστών με ειδική πρόσβαση**.
5. **Η διαχείριση του προσωπικού λογαριασμού του ιδιοκτήτη**.
6. **Η εγκατάσταση της εφαρμογής για νέους ιδιοκτήτες**.
7. **Η απόκτηση κρυπτογραφημένου κλειδιού για πρόσβαση στην υπηρεσία**.

Επίσης, μια επιμέρους λειτουργία του λογισμικού που δεν φαίνεται παραπάνω είναι η αυτόματη αποστολή email στους πελάτες για την κατάσταση της κράτησής τους.

Οι παραπάνω υπηρεσίες μπορούν να χωριστούν και με βάση τους διαφορετικούς χρήστες της εφαρμογής.

- **Πελάτης**
    a. Ανάκτηση δωματίων.
    b. Ανάκτηση διαθεσιμότητας δωματίων σε πραγματικό χρόνο σε μορφή ημερολογίου.
    c. Δημιουργία κράτησης.
- **Χρήστης τρίτων υπηρεσιών**
    a. Ανάκτηση δωματίων.
    b. Ανάκτηση διαθεσιμότητας δωματίων σε πραγματικό χρόνο σε μορφή ημερολογίου.
    c. Δημιουργία, ανάκτηση και επεξεργασία κράτησης που έχει δικαιώματα.
    d. Απόκτηση κρυπτογραφημένου κλειδιού πρόσβασης.
- **Διαχειριστής**
    a. Η δημιουργία, η ανάκτηση και επεξεργασία των χώρων ψυχαγωγίας (Games)
    b. Η ανάκτηση και η επεξεργασία του ωραρίου για κάθε χώρο.
    c. Η δημιουργία, η ανάκτηση, η επεξεργασία των κρατήσεων καθώς και της διαθεσιμότητας για κάθε χώρο σε πραγματικό χρόνο σε μορφή ημερολογίου.
    d. Η δημιουργία, η ανάκτηση και επεξεργασία των χρηστών με ειδική πρόσβαση.

e. Η διαχείριση του λογαριασμού για κάθε ιδιοκτήτη.

f. Η απόκτηση κρυπτογραφημένου κλειδιού πρόσβασης.

- **Διαχείρισης συστήματος**

a. Η δημιουργία, η ανάκτηση και επεξεργασία των χώρων ψυχαγωγίας (Games).

b. Η ανάκτηση και η επεξεργασία του ωραρίου για κάθε χώρο.

c. Η δημιουργία, η ανάκτηση και η επεξεργασία των κρατήσεων καθώς και της διαθεσιμότητας για κάθε χώρο.

d. Η δημιουργία, η ανάκτηση και επεξεργασία των χρηστών με ειδική πρόσβαση.

e. Η διαχείριση του λογαριασμού για κάθε ιδιοκτήτη.

f. Η εγκατάσταση της εφαρμογής για νέους ιδιοκτήτες.

g. Η απόκτηση κρυπτογραφημένου κλειδιού πρόσβασης.

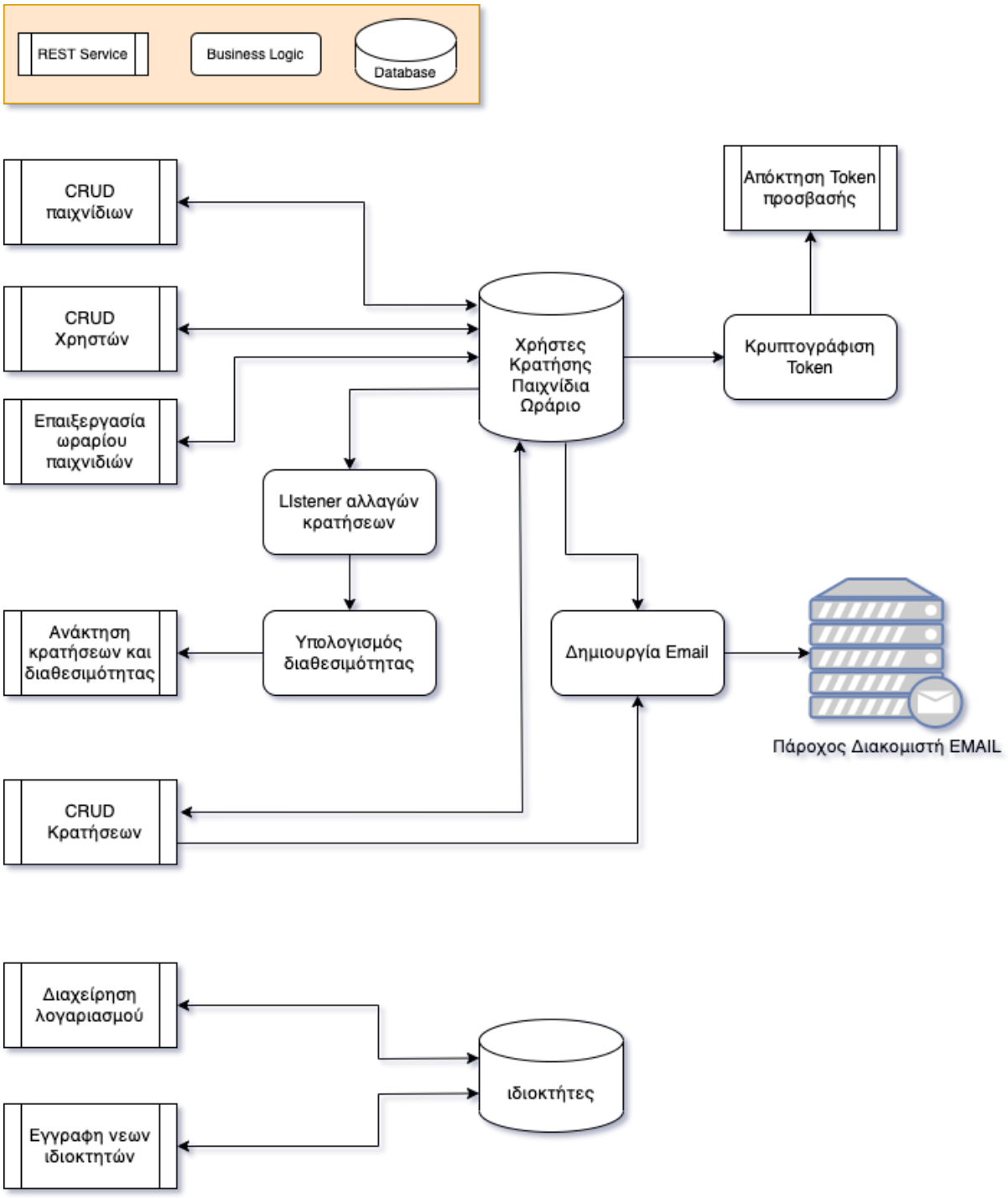
Η υπηρεσία καθώς είναι web service, θα είναι μια διαδικτυακή εφαρμογή όπου θα προσφέρει στους χρήστες της απομακρυσμένη πρόσβαση στις λειτουργίες της. Το κλασσικό μοντέλο client-server φαίνεται να εξυπηρετεί πλήρως το σύστημά μας, στο οποίο ο πελάτης θα καταναλώνει τις υπηρεσίες που προσφέρει ο διακομιστής μέσω της λογικής αίτησης/απάντησης (request/response).

6.1.2 Αρχιτεκτονική του λογισμικού της υπηρεσίας - Software Architecture

Όπως προκύπτει από τις λειτουργίες που αναλύσαμε παραπάνω ,μπορούμε να βγάλουμε κάποια συμπεράσματα για το πώς πρέπει να σχεδιαστεί το λογισμικό της υπηρεσίας. Οι βασικές συναλλαγές που θα γίνονται με αυτή είναι: η **δημιουργία**, η **ανάκτηση**, η **επεξεργασία** και η **διαγραφή** δεδομένων με σκοπό την διαχείριση των χώρων, της διαθεσιμότητας και των κρατήσεων. Αυτή η λειτουργικότητα ονομάζεται **CRUD** (create, read, update, and delete) και αποτελεί ένα μεγάλο κομμάτι της υπηρεσίας. Πέρα από την λειτουργικότητα CRUD υπάρχουν και κάποιες υπηρεσίες που χρειάζονται επιπλέον επεξεργασία των δεδομένων, όπως η διαδικασία ταυτοποίησης και η διαθεσιμότητα των κρατήσεων.

Από τα παραπάνω καταλαβαίνουμε ότι η υπηρεσία μας θα είναι ένα λογισμικό το οποίο θα λειτουργεί σαν διεπαφή για την βάση δεδομένων. Θα ανακτά δεδομένα και θα τα

επεξεργάζεται, έτσι ώστε να τα παρουσιάζει σε ειδική μορφή για κατανάλωση από μηχανές (άλλο λογισμικό). Μία αφηρημένη αναπαράσταση των λειτουργιών του λογισμικού φαίνεται στην εικόνα 6.1.



Εικόνα 6.1 Συνοπτική αναπαράσταση των λειτουργιών της υπηρεσίας

Από την εικόνα 6.1 μπορούμε να διακρίνουμε ότι το λογισμικό έχει τρία επίπεδα λειτουργικότητας:

1. Το επίπεδο **παρουσίασης**, το οποίο είναι το επίπεδο διεπαφής χρήστη. Το επίπεδο αυτό θα λάβει το αίτημα και θα αναλύσει οτιδήποτε απαιτείται από το αίτημα. Καλεί το επίπεδο λογικής, διασφαλίζει ότι η απόκριση είναι στην απαιτούμενη μορφή και την επιστρέφει σαν απάντηση στον χρήστη.
2. Το επίπεδο **επιχειρηματικής λογικής**, το οποίο είναι το επίπεδο που εκτελεί λεπτομερή επεξεργασία των δεδομένων. Αυτό το επίπεδο θα επικοινωνεί με το επίπεδο αποθήκευσης δεδομένων. Παίρνει ό,τι χρειάζεται από το επίπεδο παρουσίασης και στη συνέχεια καλεί το επίπεδο αποθήκευσης δεδομένων. Πριν και μετά την κλήση του επιπέδου αποθήκευσης δεδομένων, εφαρμόζει την επιχειρηματική λογική που απαιτείται.
3. Το επίπεδο **αποθήκευσης δεδομένων**, το οποίο επικοινωνεί με τον διακομιστή βάσης δεδομένων που αποθηκεύει πληροφορίες. Μέσω αιτημάτων στον διακομιστή της βάσης δεδομένων, εκτελεί την λειτουργία που απαιτείται.

Όταν δεν υπάρχει επιχειρηματική λογική (business logic), το επίπεδο παρουσίασης μπορεί να το προσπεράσει και να καλέσει απευθείας το επίπεδο αποθήκευσης δεδομένων. Όταν η λειτουργικότητα είναι τύπου CRUD, συνήθως δεν χρειάζεται επιχειρηματική λογική.

Η αρχιτεκτονική επιπέδων διαχωρίζει πλήρως την λειτουργικότητα του κώδικα όπως παρουσιάζεται στην εικόνα 6.2 και η επικοινωνία μεταξύ τους γίνεται μέσω διεπαφών. Έτσι ο κώδικας που προκύπτει είναι δομημένος, καθαρός και συντηρήσιμος. Επίσης, τα σενάρια δοκιμών (testing) θα είναι πιο αποτελεσματικά καθώς θα είναι στοχευμένα σε συγκεκριμένη λειτουργικότητα.

Εικόνα 6.2 Αρχιτεκτονική επιπέδων μιας διαδικτυακής υπηρεσίας

6.1.3 Διεπαφή Υπηρεσιών (Web service Interface)

Η διεπαφή χρήστη είναι ο τρόπος με τον οποίο οι χρήστες θα αλληλεπιδρούν με το λογισμικό. Οι τελικοί χρήστες δεν θα καταναλώνουν τις υπηρεσίες άμεσα αλλα μέσω μιας διαδικτυακή εφαρμογής με γραφικό περιβάλλον με εξαίρεση τους τρίτους παρόχους υπηρεσιών που θα καταναλώνουν τις υπηρεσίες μας μέσω των δικών τους συστημάτων. Οπότε η διεπαφή των υπηρεσιών προορίζεται για κατανάλωση από άλλο λογισμικό μέσω του Ιστού.

Η επικοινωνία με τον διακομιστή θα γίνεται μέσω αιτημάτων/απαντήσεων (request/response) με βάση το πρωτόκολλο HTTP. Τα αιτήματα θα ακολουθούν το αρχιτεκτονικό μοτίβο RESTful (Representational State Transfer) API (Application Programming Interface). Καθώς είναι το πιο διαδεδομενο αρχιτεκτονικό στυλ για Web services, ενδεικνύεται για συστήματα CRUD, υποστηρίζεται πλήρως από τους σύγχρονους περιηγητές και χρησιμοποιεί λιγοτερο bandwidth που το κάνει πιο αποδοτικό για τέτοιου είδους εφαρμογές.

Επίσης λόγω της ανάγκης για ενημέρωση της διαθεσιμότητας σε πραγματικό χρόνο, ο πελάτης πρέπει να λαμβάνει ενημερώσεις από τον διακομιστή όταν αλλάζει η κατάστασή της διαθεσιμότητας. Υπάρχουν διάφορες τεχνικές που μπορούν να βοηθήσουν στη επίλυση. Μια απλή προσέγγιση επίλυσης του προβλήματος είναι η κλασική χρήση του Rest, όπου ο πελάτης κάνει συνεχώς κλήσεις στον διακομιστή ώστε να λαμβάνει την πληροφορία που θέλει. Αυτή η μέθοδος όμως μπορεί να δημιουργήσει υπερβολικό φόρτο στο δίκτυο σε σχέση με τα χρήσιμα δεδομένα καθώς τις περισσότερες φορές δεν θα υπάρχει αλλαγή. Επίσης αφήνει μικρά κενά που ο client μπορεί να μην έχει ακριβή δεδομένα.

Μία καλύτερη προσέγγιση είναι η χρήση τεχνολογιών που επιτρέπουν την αποστολή δεδομένων από τον διακομιστή στον πελάτη χωρίς να χρειάζεται να προηγηθεί request για κάθε αποστολή. Τέτοιες τεχνολογίες είναι το SSE (server-sent events) και το WebSocket μεταξύ άλλων. Το WebSocket API είναι μια προηγμένη τεχνολογία που καθιστά δυνατό το άνοιγμα μιας αμφίδρομης διαδραστικής επικοινωνίας μεταξύ του προγράμματος περιήγησης του χρήστη και ενός διακομιστή. Το SSE είναι μια τεχνολογία που παρέχει ασύγχρονη επικοινωνία με ροή συμβάντων από τον διακομιστή στον πελάτη μέσω HTTP. Ο διακομιστής μπορεί να στείλει μη κατευθυνόμενα μηνύματα/συμβάντα στον πελάτη και μπορεί να ενημερώνει τον πελάτη ασύγχρονα. Το SSE σε αντιθεση με το WebSocket επιτρέπει στους διακομιστές να στέλνουν μηνύματα από τον διακομιστή στον πελάτη χωρίς αμφίδρομη επικοινωνια. [5] [6]

Έπειτα από μελέτη των παραπάνω τεχνολογιών και ανάλυσης των απαιτήσεων του συστήματος κρατήσεων μας, επιλέχθηκε η τεχνολογία του Server Sent Event. Το συμβάντα που αποστέλλονται από τον διακομιστή είναι η βέλτιστη λύση για εφαρμογές σε πραγματικό χρόνο καθώς είναι ελαφρύ, γρήγορο, εύκολο στην εφαρμογή και παρέχει χαμηλή καθυστέρηση.

Ένα RESTful API αναλύει μια συναλλαγή για να δημιουργήσει μια σειρά από μικρές μονάδες. Κάθε ενότητα απευθύνεται σε ένα υποκείμενο μέρος της συναλλαγής. Αυτό το modularity παρέχει στους προγραμματιστές μεγάλη ευελιξία.

Το REST χρησιμοποιεί εντολές για την απόκτηση πόρων. Η κατάσταση ενός πόρου σε οποιαδήποτε δεδομένη χρονική στιγμή ονομάζεται αναπαράσταση πόρου και για να προσδιοριστεί ενας πόρος χρησιμοποιείται ένα αναγνωριστικό πόρου. Ένα RESTful API χρησιμοποιεί υπάρχουσες μεθοδολογίες HTTP που ορίζονται από το πρωτόκολλο RFC 2616, όπως:

- **GET** για την ανάκτηση ένας πόρου.
- **PUT** για αλλαγή της κατάστασης ή ενημέρωση ενός πόρου, ο οποίος μπορεί να είναι αντικείμενο, αρχείο ή μπλοκ.
- **POST** για τη δημιουργία αυτού του πόρου.
- **DELETE** για να αφαιρέσει ή να καταργήσει έναν πόρο.

Μέσω του αναγνωριστικού του πόρου μπορούν να καλεστουν οι μεθοδοι HTTP. Το αναγνωριστικό θα έχει την μορφή URL σύμφωνα με τις παρακάτω πρακτικές του REST:

- Κάθε πόρος θα είναι στον πληθυντικό όταν αφορά ομάδες δεδομένων, όπως οι χρήστες (π.χ. /users).
- Όταν ο πόρος αφορά μονάδα κάποιας ομάδας, θα προσδιορίζεται με ένα μοναδικό αριθμό ID (π.χ. /users/1)
- Όταν ένας πόρος ανήκει σε κάποιον άλλον, όπως για παράδειγμα οι κρατήσεις υπάγονται κάτω από τους χώρους ψυχαγωγίας, θα συνεχίζει μετά το ID της μονάδας που ανήκει (π.χ. /games/1/bookings)

Εικόνα 6.3 Οι πόροι του REST με την αναπαράσταση τους και τα αναγνωριστικά τους

Η αναπαράσταση των πόρων μπορεί να έχει πολλές μορφές, όπως HTML αρχεία, JPEG εικόνες, δυαδικά δεδομένα ή και απλό κείμενο ή κείμενο με ειδική σημειογραφία (XML, JSON). Επιλέχθηκε το JSON για την αναπαράσταση των δεδομένων μας καθώς είναι γλωσσικά ανεξάρτητο, συμπαγές, έχει ευρεία υποστήριξη και έχει ευέλικτη μορφή.

Παρακάτω καταγράφουμε τους πόρους, τα αναγνωριστικά τους και τις αναπαραστάσεις τους όπως φαίνονται στην εικόνα 6.3 συνοπτικά και με τις μεταξύ τους σχέσεις.

games: Τα games είναι οι χώροι ψυχαγωγίας κάθε ιδιοκτήτη. Περιέχουν τις πληροφορίες του χώρου όπως κανόνες παιχνιδιού, όνομα, διεύθυνση κ.ά. Η υπηρεσία επιτρέπει την ανάκτηση των χώρων, την δημιουργία νέου χώρου καθώς και την επεξεργασία και την κατάργηση του.

Αναγνωριστικά:

Μέθοδος	Αναγνωριστικό	Περιγραφή
GET	/games	Ανάκτηση όλων των χώρων
POST	/games	Δημιουργία νέου χώρου
GET	/games/{gameID}	Ανάκτηση ενός χώρου μέσω του ID του
PUT	/games/{gameID}	Ενημέρωση ενός χώρου μέσω του ID του
DELETE	/games/{gameID}	Κατάργηση ενός χώρου μέσω του ID του

Πίνακας 6.1 Πόροι των χώρων (Games resource)

Αναπαράσταση:

Όνομα	Περιγραφή
id	Ο μοναδικός αριθμός του χώρου
status	Η κατάσταση του χώρου που προσδιορίζει αν είναι ενεργός ή ανενεργός
name	Το όνομα ή ο τίτλος του χώρου
description	Η περιγραφή του χώρου (τύπος παιχνιδιού, κανόνες παιχνιδιού

	κ.ά.)
address	Η διεύθυνση του χώρου
image URL	Το URL για την φωτογραφία του χώρου
map URL	Το URL της τοποθεσίας στο Google maps
number of players	Ο επιτρεπόμενος αριθμός παικτών
duration	Ο χρόνος που διαρκεί ένα παιχνίδι στον χώρο
Age restriction	Η επιτρεπόμενη ηλικία

Πίνακας 6.2 Αναπαράσταση των γνωρισμάτων των χώρων (Games representation)

bookings: Τα bookings είναι οι κρατήσεις για κάθε game. Περιέχουν τις πληροφορίες της κράτησης όπως ημερομηνία, στοιχεία του πελάτη και πιο game αφορά.Η υπηρεσία επιτρέπει την ανάκτηση των κρατήσεων, την δημιουργία νέας κράτησης καθώς, την επεξεργασία και την ακύρωση της. Η κράτηση αφορά πάντα ένα συγκεκριμένο game, οπότε το αναγνωριστικό του συνεχίζει από αυτό του game το οποίου ανήκει π.χ. /games/1/bookings

Αναγνωριστικά:

Μέθοδος	**Αναγνωριστικό**	**Περιγραφή**
GET	/games/{gameID}/bookings	Ανάκτηση όλων των κρατήσεων του συγκεκριμένου game.
POST	/games/{gameID}/bookings	Δημιουργία νέας κράτησης για το συγκεκριμένο game.

GET	/games/{gameID}/bookings/{bookingID}	Ανάκτηση μιας κράτησης μέσω του ID της για το συγκεκριμένο game.
PUT	/games/{gameID}/bookings/{bookingID}	Ενημέρωση μιας κράτησης μέσω του ID της για το συγκεκριμένο game.
DELETE	/games/{gameID}/bookings/{bookingID}	Ακύρωση μιας κράτησης μέσω του ID της για το συγκεκριμένο game.

Πίνακας 6.3 Πόροι των κρατήσεων (Bookings resource)

Αναπαράσταση:

Όνομα	Περιγραφή
id	Ο μοναδικός αριθμός της κράτησης
date	Η ώρα και η ημερομηνία της κράτησης
name	Το όνομα του πελάτη
mobile number	Ο αριθμός τηλεφώνου του πελάτη
email	Το email του πελάτη
notes	Χώρος ελεύθερου κειμένου για τυχόν σημείωμα του πελάτη

Πίνακας 6.4 Αναπαράσταση των γνωρισμάτων των κρατήσεων (Bookings representation)

schedule: Το schedule αφορά το εβδομαδιαίο ωράριο για κάθε game. Περιέχει τις προκαθορισμένες ώρες του game που είναι διαθέσιμες για κράτηση, για κάθε ημέρα της

εβδομάδας. Επίσης επειδή το ωράριο μπορεί να αλλάζει ανα περιόδους, μπορεί να υπάρχει και δεύτερο ωράριο με ισχύ από συγκεκριμένη ημερομηνία που αναφέρεται.

Η υπηρεσία επιτρέπει την ανάκτηση του ωραρίου, την επεξεργασία του τρέχων ωραρίου, καθώς και την επεξεργασία του ωραρίου για την επόμενη περίοδο. Θεωρείτε ότι κάθε game ξεκινάει με ένα κενό ωράριο.

Το schedule αφορά πάντα ένα συγκεκριμένο game, οπότε το αναγνωριστικό του συνεχίζει από αυτό του game το οποίου ανήκει π.χ. /games/1/schedule.

Αναγνωριστικά:

Μέθοδος	**Αναγνωριστικό**	**Περιγραφή**
GET	/games/{gameID}/schedule	Ανάκτηση του ωραρίου του συγκεκριμένου game.
PUT	/games/{gameID}/schedule	Ενημέρωση του τρέχων ωραρίου του συγκεκριμένου game.
PUT	/games/{gameID}/schedule/new	Ενημέρωση του ωραρίου της επόμενης περιόδου του συγκεκριμένου game.

Πίνακας 6.5 Πόροι του προγράμματος κρατήσεων (Schedule resource)

Αναπαράσταση:

Όνομα	**Περιγραφή**
Active schedule	Το τρέχον εβδομαδιαίο ωράριο. Το εβδομαδιαίο ωράριο περιέχει για κάθε μέρα της εβδομάδας τις ώρες που ξεκινάει ένα game. Π.χ. Δευτέρα: 14:00, 18:00, Τρίτη: 13:30, 17:00 κ.ο.κ.
Upcoming schedule	Το εβδομαδιαίο ωράριο της επόμενης περιόδου. Το εβδομαδιαίο ωράριο περιέχει για κάθε μέρα της εβδομάδας τις ώρες που

	ξεκινάει ένα game καθώς και την ημερομηνία που τίθεται σε ισχύ. Π.χ. Για ένα ωράριο που θα ισχύει από τις 14/03/2023 και μετά, θα είναι: Ημερομηνία: 14/03/2023, Δευτερα: 14:00, 18:00, Τρίτη: 13:30, 17:00 κ.ο.κ.

Πίνακας 6.6 Αναπαράσταση των γνωρισμάτων του προγράμματος κρατήσεων (Schedule representation)

calendar: Το calendar είναι ένα ημερολόγιο με τις κρατήσεις και την διαθεσιμότητα κάθε game για τον επιλεγμένο μήνα όπως προκύπτει από το schedule. Περιέχει τις επιτρεπόμενες ώρες για κράτηση κάθε μέρα του μήνα και αν αυτές είναι διαθέσιμες ή αν υπάρχει ήδη κράτηση. Επίσης, μπορεί να υπάρχουν κρατήσεις εκτός του προκαθορισμένου ωραρίου που ίσως προκύψουν λόγω αλλαγής του ωραρίου ενώ προϋπήρχε κράτηση. Η υπηρεσία επιτρέπει την ανάκτηση του ημερολογίου για τον επιλεγμένο μήνα. Επίσης λόγω της ανάγκης για ενημέρωση σε πραγματικό χρόνο υπάρχει η δυνατότητα ανάκτησης μέσω της τεχνολογίας SSE για να ανανεώνει την διαθεσιμότητα αυτόματα σε περίπτωση νέας κράτησης ή αλλαγής του προγράμματος. Το calendar αφορά πάντα ένα συγκεκριμένο game, οπότε το αναγνωριστικό του συνεχίζει από αυτό του game στο οποίου ανήκει π.χ. /games/1/calendar.

Αναγνωριστικά:

Μέθοδος	**Αναγνωριστικό**	**Παράμετροι**	**Περιγραφή**
GET	/games/{gameID}/calendar	Μήνας	Ανάκτηση του calendar του συγκεκριμένου game για τον επιλεγμένο μήνα.
GET	/games/{gameID}/schedule/sse	Μήνας	Ανάκτηση του calendar του συγκεκριμένου game για τον επιλεγμένο μήνα με αυτόματη ενημέρωση διαθεσιμότητας.

Πίνακας 6.7 Πόροι του ημερολογίου (Calendar resource)

Αναπαράσταση:

Όνομα	**Περιγραφή**
Event	Τα events είναι οι επιτρεπόμενες ώρες για κράτηση στο ημερολόγιο. Κάθε event περιέχει την μέρα του μήνα, την ώρα που αρχίζει, την διάρκειά του, το ID του booking αν υπάρχει, καθώς και μία σημείωση για την περίπτωση που η κράτηση είναι εκτός του προκαθορισμένου ωραρίου.

Πίνακας 6.8 Αναπαράσταση των γνωρισμάτων του ημερολογίου (Calendar representation)

users: Οι users είναι οι χρήστες του συστήματος με ειδική πρόσβαση. Αυτοί οι χρήστες μπορεί να είναι ο ιδιοκτήτης, ο διαχειριστής, ο χειριστής του παιχνιδιού καθώς και ο τρίτος πάροχος. Οι χρήστες πρέπει να έχουν ένα όνομα χρήστη και έναν κωδικό πρόσβασης που τους επιτρέπει να χρησιμοποιούν τις λειτουργίες περιορισμένης πρόσβασης καθώς και τον ρόλο που έχουν στο σύστημα για να προσδιορίζεται η επιτρεπόμενη πρόσβαση.

Η υπηρεσία επιτρέπει την ανάκτηση των χρηστών, την δημιουργία νέου χρήστη, την ενημέρωση των στοιχείων του και την κατάργηση του.

Αναγνωριστικά:

Μέθοδος	**Αναγνωριστικό**	**Περιγραφή**
GET	/users	Ανάκτηση όλων των χρηστών
POST	/users	Δημιουργία νέου χρήστη
GET	/users/{userID}	Ανάκτηση ενός χρήστη μέσω του ID του
PUT	/users/{userID}	Ενημέρωση ενός χρήστη μέσω του ID του
DELETE	/users/{userID}	Κατάργηση ενός χρήστη μέσω του ID του

Πίνακας 6.9 Πόροι των χρηστών (Users resource)

Αναπαράσταση:

Όνομα	Περιγραφή
id	Ο μοναδικός αριθμός του χρήστη
username	Το όνομα χρήστη που χρησιμοποιείται για την πρόσβαση στο σύστημα
password	Ο κωδικός που χρησιμοποιείται για την πρόσβαση στο σύστημα
role	Ο ρόλος του χρήστη στο σύστημα

Πίνακας 6.10 Αναπαράσταση των γνωρισμάτων των χρηστών (Users representation)

Account: Το Account αφορά τον λογαριασμό του ιδιοκτήτη στο σύστημα. Περιέχει το όνομα του ιδιοκτήτη, το όνομα της εταιρείας, τα στοιχεία της πληρωμής και της συνδρομής του.

Η υπηρεσία επιτρέπει την ανάκτηση των λογαριασμών, την δημιουργία νέου λογαριασμού, την ενημέρωσή του και την κατάργησή του.

Αναγνωριστικά:

Μέθοδος	Αναγνωριστικό	Περιγραφή
GET	/accounts	Ανάκτηση όλων των λογαριασμών
POST	/accounts	Δημιουργία νέου λογαριασμού
GET	/accounts/{accountID}	Ανάκτηση ενός λογαριασμού μέσω του ID του
PUT	/accounts/{accountID}	Ενημέρωση ενός λογαριασμού μέσω του ID του

DELETE	/accounts/{accountID}	Κατάργηση ενός λογαριασμού μέσω του ID του

Πίνακας 6.11 Πόροι του λογαριασμού (Accounts resource)

Αναπαράσταση:

Όνομα	**Περιγραφή**
id	Ο μοναδικός αριθμός του λογαριασμού
owner	Τα στοιχεία του ιδιοκτήτη
active subscription	Πληροφορίες για την συνδρομή του π.χ. μέχρι πότε είναι ενεργή
company name	Το όνομα της εταιρείας του ή της μάρκας του
billing info	Τα στοιχεία πληρωμής του για την συνδρομή

Πίνακας 6.12 Αναπαράσταση των γνωρισμάτων του λογαριασμού (Accounts representation)

Authentication: Το authentication αφορά την απόκτηση κρυπτογραφημένου κλειδιού (token) για χρήση των λειτουργιών περιορισμένης πρόσβασης στην υπηρεσία. Η υπηρεσία επιτρέπει την απόκτηση του token για χρήστες με σωστά στοιχεία πρόσβασης, δηλαδή όνομα και κωδικό χρήστη.

Αναγνωριστικά:

Μέθοδος	Αναγνωριστικό	Παράμετροι	Περιγραφή
POST	/authentication	username, password	Αποκτήση κρυπτογραφημένου κλειδιού για τον χρήστη με τα στοιχεία πρόσβασης

Πίνακας 6.13 Πόροι της αυθεντικοποίησης (Authentication resource)

Αναπαράσταση:

Όνομα	Περιγραφή
token	Το κρυπτογραφημένο κλειδί

Πίνακας 6.14 Αναπαράσταση των γνωρισμάτων της αυθεντικοποίησης (Authentication representation)

6.1.4 Επιχειρηματική λογική

Στο λογισμικό υπολογιστών, η επιχειρηματική λογική είναι το μέρος του προγράμματος που κωδικοποιεί τους επιχειρηματικούς κανόνες του πραγματικού κόσμου που καθορίζουν τον τρόπο δημιουργίας, αποθήκευσης και αλλαγής δεδομένων. Σε αντίθεση με το υπόλοιπο λογισμικό που μπορεί να αφορά λεπτομέρειες χαμηλότερου επιπέδου διαχείρισης μιας βάσης δεδομένων ή εμφάνισης της διεπαφής χρήστη, της υποδομής του συστήματος ή γενικά της σύνδεσης διαφόρων τμημάτων του προγράμματος.

Οι λειτουργίες επιχειρηματικής λογικής του συστήματος μας όπως φαίνονται στην εικόνα 6.1 είναι:

1. Η δημιουργία κρυπτογραφημένου κλειδιού για εξουσιοδοτημένη χρήση της υπηρεσίας.
2. Η δημιουργία και αποστολή δυναμικών email στους πελάτες για την κατάσταση της κράτησής τους.
3. Η δημιουργία ημερολογίου κατάστασης κρατήσεων και διαθεσιμότητας.

4. Ενημέρωση των χρηστών για αλλαγές στην διαθεσιμότητα και στις κρατήσεις σε πραγματικό χρόνο

Οι λειτουργίες που υλοποιούν το μοντέλο CRUD δεν χρειάζονται κάποια λογική καθώς ο σκοπός τους είναι η αλληλεπίδραση με την βάση δεδομένων. Η λογική για κάθε λειτουργία χρειάζεται διαφορετική λύση λόγω της διαφορετικής φύσεως των προβλημάτων. Παρακάτω θα σχεδιάσουμε τους μηχανισμούς για κάθε λειτουργία.

Δημιουργία κρυπτογραφημένου κλειδιού

Κάθε χρήστης θα πρέπει να αποκτά ένα τέτοιο κλειδι που θα μπορεί να εκδίδεται μόνο από την υπηρεσία για να έχει περαιτέρω πρόσβαση σε αυτή. Αυτό το κλειδί θα έχει περιορισμένο χρόνο ζωής για να τίθεται άχρηστο σε περίπτωση υποκλοπής. Επίσης θα πρέπει να αντιστοιχεί σε συγκεκριμένο αριθμό χρήστη λόγο διαφορετικής προσβάσης κάθε χρήστη καθώς και για δυνατότητα καταγραφής των ενεργειών που εκτελεί για λόγους ασφαλείας.

Σαν κρυπτογραφημένο κλειδί θα χρησιμοποιήσουμε το JWT. Το JWT καλύπτει όλες τις ανάγκες μας καθώς μπορεί να αποθηκεύσει πληροφορίες με συμπαγή και αυτόνομο τρόπο για την ασφαλή μεταφορά τους καθώς αυτές οι πληροφορίες μπορούν να επαληθευτούν και να εμπιστευτούν επειδή είναι ψηφιακά υπογεγραμμένες. Το JWT θα δημιουργείται όταν ένας χρήστης ζητάει εξουσιοδότηση μέσω της διεπαφής. Αφού επαληθευτούν τα στοιχεία πρόσβασης του χρήστη μέσω του επιπέδου αποθήκευσης δεδομένων θα δημιουργηθεί ένα JWT με ημερομηνία λήξης και τον μοναδικό αριθμό ID του χρήστη και θα υπογραφεί με ένα μυστικό αναγνωριστικό με τον αλγόριθμο HMAC. Κάθε φορά που ο χρήστη θα εκτελεί εντολές περιορισμένης πρόσβασης στην υπηρεσία, θα παρέχει το JWT. Πριν την εκτέλεση της εντολής, θα τρέχει ένας ενδιάμεσος αλγόριθμος που που θα πιστοποιεί ότι το JWT έχει δημιουργηθεί από εμάς μέσω του μυστικού αναγνωριστικού, και οτι δεν έχει λήξει. Έπειτα θα εξάγει το ID του χρήστη στο επίπεδο παρουσίασης.

Δημιουργία και αποστολή δυναμικών email

Κάθε φορά που δημιουργείται, τροποποιείται ή ακυρώνεται μια κράτηση, ο πελάτης που έκανε την κράτηση θα ενημερώνεται μέσω email. Τα emails θα δημιουργούνται μέσω προκαθορισμένων προτύπων ανα περίπτωση και θα συμπληρώνονται με τα δεδομένα της κράτησης. Η αποστολή τους θα γίνεται μέσω διακομιστή email τρίτου παρόχου. Ο μηχανισμός δημιουργίας email θα λαμβάνει το ID της κράτησης καθώς και τον κωδικό του

προτύπου. Θα ανακτά τα δεδομένα της κράτησης και το πρότυπο από το επίπεδο αποθήκευσης δεδομένων και θα παράγει ένα email έτοιμο προς αποστολή. Το email θα πρέπει να περιέχει αποστολέα, αποδεκτή, θέμα και σώμα μηνύματος για να μπορει να το δεχτεί ο διακομιστής email. Έπειτα θα στέλνεται στον διακομιστή για αποστολή. Επίσης κάθε προσπάθεια email θα πρέπει να καταγράφεται στην βάση δεδομένων μέσω του επιπέδου αποθήκευσης δεδομένων.

Δημιουργία ημερολογιου κατάστασης κρατήσεων και διαθεσιμότητας

Ο χρήστης θα μπορεί να ανακτά τις κρατήσεις και την διαθεσιμότητα ανα μήνα για κάθε χώρο. Αυτη η μηνιαία κατάσταση θα αποτυπώνεται σε συμβάντα ημερολογίου. Το ημερολόγιο θα περιέχει τις επιτρεπόμενες ώρες κράτησης για κάθε μέρα του μήνα όπως θα προκύπτουν από το εβδομαδιαίο πρόγραμμα του κάθε χώρου και την κατάσταση τους, αν είναι διαθέσιμα ή έχει γίνει ήδη κράτηση. Στην περίπτωση που την ανάκτηση του ημερολογίου την ζητά ο πελάτης ή ο τρίτος πάροχος, θα επιστρέφει μόνο τα διαθέσιμα για κράτηση ωράρια καθώς δεν πρέπει να έχουν πρόσβαση στις κρατήσεις του της επιχείρησης. Ο μηχανισμός θα λαμβάνει τα διαθέσιμα ωράρια και τις κρατησεις για την συγκεκριμένη περίοδο από την αποθήκη δεδομένων. Έπειτα θα πρέπει να υπολογίζει πιο ωράριο να εφαρμόσει για κάθε μέρα του μήνα. Στην περίπτωση που θέλουμε να δείξουμε μόνο διαθεσιμότητα, όλές οι μη διαθέσιμες ώρες θα δεν θα συμπεριλαμβάνονται, αλλιώς θα συμπεριλαμβάνει το ID της κράτησης αν υπάρχει. Επίσης λόγω αλλαγών στο ωράριο μπορεί να υπάρχουν κρατήσεις εκτός του προκαθορισμένου ωραρίου και να τις συμπεριλαμβάνει και αυτές μαρκαρισμένες σαν κρατήσεις εκτός ωραρίου.

Ενημέρωση ημερολογίου σε πραγματικό χρόνο

Το ημερολόγιο θα ανακτάται από τους χρήστες μέσω του REST. Την ενημέρωση του τελικού χρήστη σε πραγματικό χρόνο θα την αναλάβει το επίπεδο παρουσίασης μέσω της τεχνολογίας SSE. Δηλαδή ο χρήστης αφού αιτηθεί την ανάκτηση του ημερολογίου σε πραγματικό χρόνο, ο διακομιστής θα στέλνει ενημερώσεις όταν υπάρχουν. Ο μηχανισμός αυτός θα πρέπει να παρακολουθεί μέσω του επιπέδου αποθήκευσης δεδομένων για την δημιουργία, την αλλαγή, την ακύρωση κάποιας κράτησης ή πιθανές αλλαγές στο ωράριο. Η παρακολούθηση της βάσης δεδομένων θα αναλυθεί στο επίπεδο αποθήκευσης δεδομένων. Όταν υπάρξει κάποια αλλαγή, θα πρέπει να στείλει σήμα στο επίπεδο παρουσίασης για ενημέρωση του ημερολογίου σε όσους χρήστες έχουν εγγραφεί στα SSE.

6.1.5 Αποθήκευση δεδομένων

Η αλληλεπίδραση με την βάση δεδομένων θα γίνεται αποκλειστικά από αυτό το επίπεδο. Αυτή η απομόνωση μας επιτρέπει να ξεχωρίσουμε την βάση από το υπόλοιπο λογισμικό ώστε να μπορούμε να την αντικαταστήσουμε ή να την αλλάξουμε χωρίς να επηρεαστούν οι λειτουργίες του λογισμικού. Επίσης στο στάδιο των δοκιμών (testing) θα μας επιτρέψει να τρέξουμε σενάρια στοχεύοντας συγκεκριμένα μέρη του κώδικα χωρίς να τα επηρεάζονται από την βάση δεδομένων. Το επίπεδο αποθήκευσης δεδομένων θα συνδέεται στην βάση δεδομένων και θα προσφέρει προκαθορισμένες ενέργειες μέσω διεπαφής. Οι κύριες ενέργειες προς την βάση δεδομένων είναι η ανάκτηση (select), η ενημέρωση (update), εισαγωγή (insert) και διαγραφή (delete) δεδομένων. Για κάθε οντότητα του λογισμικού θα υπάρχουν οι συγκεκριμένες ενέργειες που θα απαιτεί για παράδειγμα μια κράτηση θα μπορεί να εισάγεται στην βάση, να ενημερώνεται και να ανακτάται. Οπότε η συνάρτηση που θα αναλαμβάνει αυτες τις ενέργειες θα λαμβάνει τα απαιτούμενα δεδομένα και θα δημιουργεί ένα ερώτημα (query) προς την βάση. Η μορφή του ερωτήματος και ο τρόπος εκτέλεσης της εξαρτάται από την τεχνολογια της βάσης δεδομένων.

Μια επιπλέον λειτουργία που πρέπει να αναλάβει η αποθήκευση δεδομένων είναι η παρακολούθησή ορισμένων πινάκων για αλλαγές. Αυτή η λειτουργία θα χρησιμοποιηθεί για την ενημέρωση της διαθεσιμότητας και των κρατήσεων σε πραγματικό χρόνο. Κάθε φορά που θα εισάγεται ή θα ενημερώνεται μια κράτηση ή θα ενημερώνεται το εβδομαδιαίο ωράριο ή βάση θα ενημερώνει το επίπεδο αποθήκευσης δεδομένων. Η βάση δεδομένων θα είναι η PostgreSQL, καθώς είναι μιά σύγχρονη σχεσιακή βάση δεδομένων που υποστηρίζει πλήρως τις λειτουργίες που χρειαζόμαστε.

6.2 Υλοποίηση Web Service

Για την υλοποίηση του web service επιλέξαμε να χρησιμοποιήσουμε την γλώσσα προγραμματισμού Golang(Go). Η Go έχει σχεδιαστεί για να επιτρέπει στους προγραμματιστές να αναπτύσσουν γρήγορα επεκτάσιμες και ασφαλείς εφαρμογές Ιστού. Η Go διατίθεται με έναν εύκολο στη χρήση, ασφαλή και αποτελεσματικό διακομιστή ιστού και περιλαμβάνει τη δική του βιβλιοθήκη προτύπων ιστού. Η Go έχει εξαιρετική υποστήριξη για όλες τις πιο πρόσφατες τεχνολογίες από το HTTP/2, μέχρι βάσεις δεδομένων όπως PostgreSQL, MongoDB και ElasticSearch, μέχρι τα πιο πρόσφατα πρότυπα κρυπτογράφησης, συμπεριλαμβανομένου του TLS 1.3. Οι εφαρμογές ιστού Go εκτελούνται εγγενώς σε οποιοδήποτε περιβάλλον, cloud ή λειτουργικό σύστημα χάρη στην εξαιρετική

φορητότητα του Go. Το λογισμικό θα είναι ένας διακομιστής HTTP που θα επιτρέπει την κατανάλωση του web service μέσω HTTP requests

6.2.1 Δομή του κώδικα

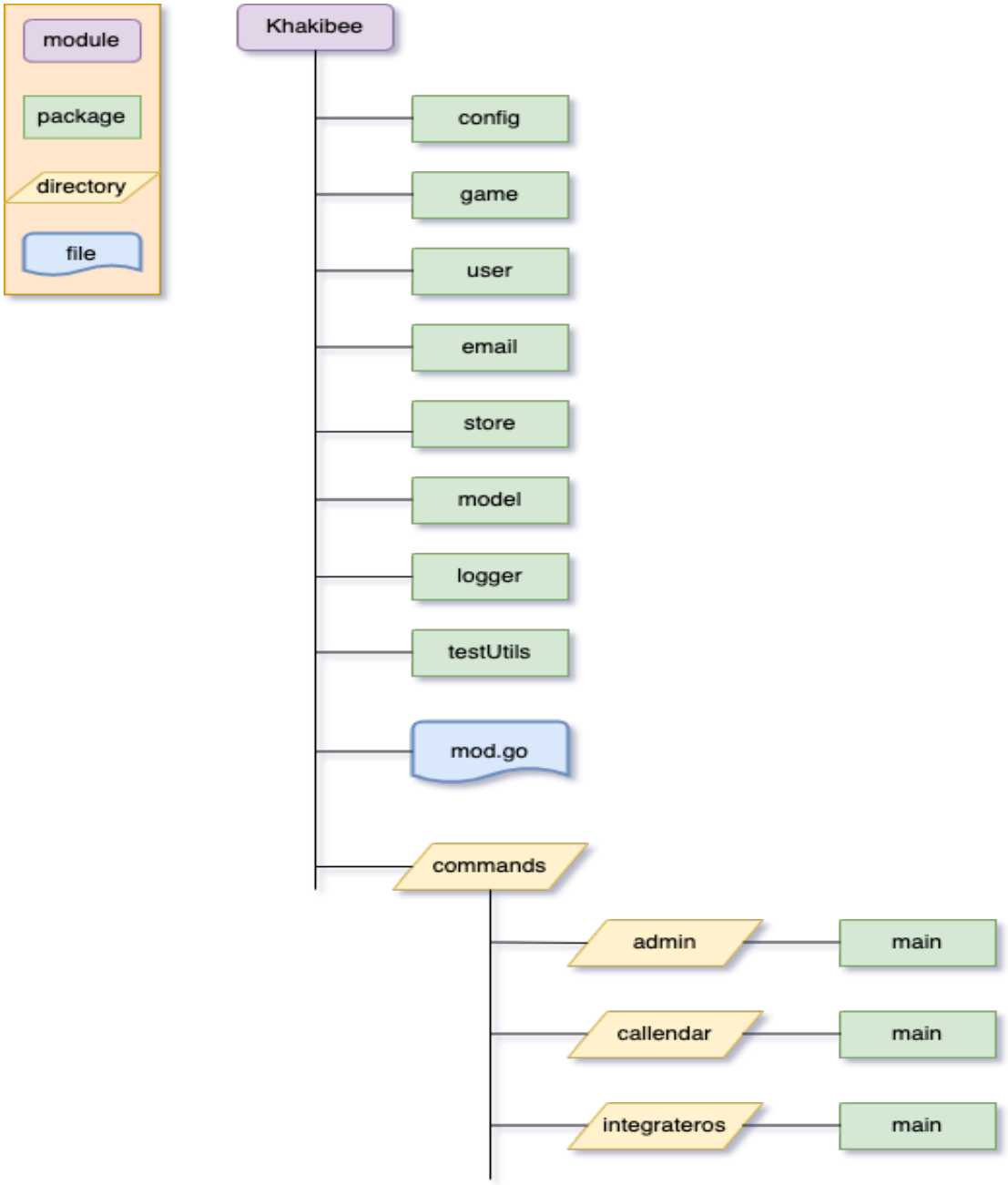
Τα προγράμματα Go είναι οργανωμένα σε packages. Ένα package είναι μια συλλογή αρχείων κώδικα στον ίδιο κατάλογο που γίνονται compile μαζί. Οι συναρτήσεις, οι τύποι, οι μεταβλητές και οι σταθερές που ορίζονται σε ένα αρχείο προέλευσης είναι ορατές σε όλα τα άλλα αρχεία προέλευσης στο ίδιο package. Ένα module είναι μια συλλογή σχετικών packages Go που κυκλοφορούν μαζί. Ένα αρχείο με το όνομα go.mod στη ρίζα(το πρώτο επίπεδο του καταλόγου) δηλώνει το module path: το import path prefix για όλα τα packages εντός του module. Το module περιέχει τα packages στον κατάλογο που περιέχει το αρχείο go.mod καθώς και υποκαταλόγους αυτού του καταλόγου, μέχρι τον επόμενο υποκατάλογο που περιέχει άλλο αρχείο go.mod (αν υπάρχει). Η πρώτη δήλωση σε ένα αρχείο κώδικα Go πρέπει να είναι όνομα του package. Οι εκτελέσιμες εντολές πρέπει πάντα να χρησιμοποιούν το package main.

Μπορούμε να χρησιμοποιήσουμε τα packages για να οργανώσουμε τα διαφορετικά επίπεδα και λειτουργίες του λογισμικού για να δημιουργήσουμε το module που αφορά το web service. Καθώς κάθε επίπεδο είναι ανεξάρτητο από το άλλο, μπορεί να είναι ένα package. Το package θα υλοποιεί όλη την λειτουργικότητα ανεξάρτητα από το υπόλοιπο module και θα επικοινωνει με τα υπόλοιπα μέσω ενός interface που θα εκθέτει τις επιτρεπόμενες λειτουργίες. Για παράδειγμα το package της αποθήκευσης δεδομένων θα μπορεί να συνδεθεί στην βάση δεδομένων και να εκτελεί queries σε αυτή. Οπότε για να γίνει μια εισαγωγη μιας κράτησης στην βάση δεδομένων θα πρέπει να γίνει import το package της αποθήκευσης δεδομένων και να κληθεί η αντίστοιχη συνάρτηση. Οπότε το λογισμικό θα είναι ένα module που θα περιέχει packages για κάθε ομάδα λειτουργιών που σχετίζονται ώστε να μπορούν να μοιράζονται κοινά κομμάτια κώδικα και να απομονώνονται από τα υπόλοιπα για καλύτερη οργάνωση. Από το package main θα συντίθεται το εκτελέσιμο πρόγραμμα χρησιμοποιώντας τις λειτουργίες από τα packages για την λειτουργία του διακομιστή. Η δομή του module φαίνεται στο διάγραμμα της εικόνας 6.4. Οπως φαινεται στην εικονα 6.4 υπάρχουν τρια main packages, με αυτον τον τρόπο μπορούμε να μεταγλωττίσουμε τρία διαφορετικά εκτελέσιμα αρχεία που θα χρησιμοποιούν μόνο τα packages που χρειάζονται. Η ανάγκη των διαφορετικών εκτελέσιμων θα αναλυθεί περισσότερο παρακάτω.

6.2.2 HTTP Handlers

Τα packages που θα λειτουργούν σαν HTTP handlers θα περιέχουν τις συναρτήσεις που θα χειρίζονται τα HTTP responses. Οι συναρτήσεις handler θα λαμβάνουν το HTTP request που αφορά έναν συγκεκριμένο πόρο. Θα επικυρώνουν τις παραμέτρους για τα αιτήματα GET και το body για τα αιτήματα POST ή PUT. Έπειτα θα εκτελούν την επιχειρηματική λογική ή θα χρησιμοποιούν την αποθήκευση δεδομένων. Και τέλος θα ετοιμάζουν ένα HTTP response. Για κάθε μέθοδο και αναγνωριστικό πόρου πρέπει να υπάρχει ένας http handler που θα χειρίζεται το request. Οι συναρτήσεις handler θα δέχονται παραμετρικά ένα στιγμιότυπο τύπου echo.Context που θα περιέχει το HTTP request. Το στιγμιότυπο είναι μέρος του package echo που θα χρησιμοποιηθεί για την δημιουργία του διακομιστή HTTP και θα αναλυθεί σε επόμενο υποκεφάλαιο.

Στον κώδικα 6.1 φαίνεται ο handler GetGame που είναι υπεύθυνος για την ανάκτηση κρατήσεων. Ο handler έχοντας πρόσβαση στην αποθήκη δεδομένων και στο HTTP request ανακτά τον πόρο και τον επιστρέφει. Σε περίπτωση σφάλματος επιστρέφει ένα HTTP response με το ανάλογο status code και ένα μήνυμα σφάλματος.



Εικόνα 6.4 Η δομή του modules με τα packages

Το package θα πρέπει να μπορεί να καλέσει την αποθήκευση δεδομένων και την επιχειρηματική λογική που χρειάζεται. Κάθε handler package θα έχει έναν τύπο struct και σαν πεδία θα έχει τα στιγμιότυπα για την αποθήκευση δεδομένων και την επιχειρηματική λογική που χρειάζεται. Στον κώδικα 6.2 βλέπουμε το struct GameHandler του package game. Οι συναρτήσεις handler θα είναι μέθοδοι του struct και μέσω ενός στιγμιότυπου του struct θα μπορουν να κληθούν.

Το package game θα χειρίζεται τα requests που αφορούν τους πόρους του game. Θα μπορεί να χρησιμοποιεί την αποθήκη δεδομένων για την πρόσβαση στα δεδομένα που αφορούν τα games. Επίσης θα μπορεί να στέλνει ενημερωτικά emails στους πελάτες σχετικά με την κράτησή τους χρησιμοποιώντας ένα στιγμιότυπο του package email που θα αναλυθεί παρακάτω. Και τέλος θα πρέπει να λαμβάνει σήμα για αλλαγή στις κρατήσεις για να μπορεί να στέλνει στον πελάτη SSE.

Το package game θα χειρίζεται τα requests που αφορούν τους πόρους του game. Θα μπορεί να χρησιμοποιεί την αποθήκη δεδομένων για την πρόσβαση στα δεδομένα που αφορούν τα games. Επίσης θα μπορεί να στέλνει ενημερωτικά emails στους πελάτες σχετικά με την κράτησή τους χρησιμοποιώντας ένα στιγμιότυπο του package email που θα αναλυθεί παρακάτω. Και τέλος θα πρέπει να λαμβάνει σήμα για αλλαγή στις κρατήσεις για να μπορεί να στέλνει στον πελάτη SSE.

Επικύρωση δεδομένων που εισάγονται από τον χρήστη

Ο χρήστης μπορεί να στείλει δεδομένα μέσω request είτε παραμετρικά για μέθοδο GET είτε στο body για τις μεθόδους POST και PUT. Αυτά τα δεδομένα πριν χρησιμοποιηθούν πρέπει επικυρωθούν οτι είναι σωστά ώστε να μην προκύψει κάποιο σφάλμα κατα την χρήση τους. Η επικύρωση των δεδομένων θα γίνεται με την χρήση του package validator της τυπικής βιβλιοθήκης Go. Για κάθε request θα υπάρχει ένα struct με τα πεδία των δεδομένων καθώς και ένας κανόνας επιτρεπόμενων τιμών για κάθε πεδίο. Το struct θα υλοποιεί την συνάρτηση bind που θα διαβάζει τα δεδομένα από το request, θα τα επικυρώνει και τέλος θα αναθέτει τις τιμές στο αντίστοιχο model struct για χρήση από τον handler. Στην περίπτωση που η επικύρωση δεν είναι επιτυχής, θα επιστρέφει ένα σφάλμα που θα περιέχει τον λόγο της αποτυχίας. Στον κώδικα 6.4 παραθέτουμε το struct για το request του BookGame και την μέθοδο bind για την επικύρωση των τιμών.

HTTP Response

Κάθε HTTP request θα έχει σαν αποτέλεσμα ένα HTTP response. Το HTTP response θα έχει πάντα ένα status code που θα δηλώνει το αποτέλεσμα του request. Στην περίπτωση που η μέθοδος είναι GET τότε θα περιέχει ένα body που θα είναι ένα JSON με το αποτέλεσμα της μεθόδου. Στην περίπτωση που προέκυψε κάποιο σφάλμα κατα την εκτέλεση, μπορεί να υπάρχει body που να περιέχει επιπλέον πληροφορίες για το σφάλμα. Παρακάτω

παραθέτουμε ένα παράδειγμα από τα πιθανά HTTP responses για το HTTP Request GET /games/1.

GET /games/1		
Περίπτωση	**Status**	**Body**
Οι παράμετροι του request δεν έχουν την σωστή μορφή	422 Unprocessable Entity	
Προέκυψε σφάλμα στον διακομιστή	500 Internal Server Error	
Ο πόρος δεν υπάρχει	404 Not Found	`["code=400, message=strconv.ParseInt: parsing \"δ\": invalid syntax, internal=strconv.ParseInt: parsing \"δ\": invalid syntax"]`
Ο πόρος υπάρχει	200 OK	`{ "game_id": 1, "status": "active", "name": "Mansion", "descr": "The first escape house in Europe !", "addr": "Charokopou 93, Kallithea, 2st Floor", "img_url": "https://domain.com/image/mansion.jpg", "map_url": "https://www.google.com/maps/", "players": "3-7", "duration": 180, "age_range": "15+" }`

Πίνακας 6.15 Παραδείγματα HTTP responses για τον πόρο “GET /games/1”.

6.2.3 Store

Το package store θα έχει συναρτήσεις για την αποθήκευση δεδομένων. Μέσω της σύνδεσης με την βάση δεδομένων θα μπορεί να εκτελεί SQL queries. Η επικοινωνία με την βάση δεδομένων γίνεται από ένα στιγμιότυπο τύπου db.sql του package database/sql της τυπικής βιβλιοθήκης της Go. Οι συναρτήσεις για την εκτέλεση queries στην βάση, έχοντας πρόσβαση στο στιγμιότυπο τύπου db.sql, δέχονται παραμετρικά την πληροφορία που πρόκειται να εισαχθεί στην βάση δεδομένων ή που χρειάζεται για την ανάκτηση δεδομένων όπως ένα id καποίας εγγραφής ή ένα φίλτρο. Στο σώμα της συνάρτησης δημιουργείται το query κατάλληλα και στέλνετε στην βάση δεδομένων. Έπειτα επιστρέφει το αποτέλεσμα της βάσης δεδομένων αν υπάρχει ή σφάλμα σε περίπτωση που αποτύχει. Η συνάρτηση δέχεται επίσης μια παράμετρο τύπου context.Context. Αυτή η παράμετρος χρησιμοποιείται για την ακύρωση

της εκτέλεσης της μεθόδου αν υπερβεί τον χρόνο που έχει οριστεί σε αυτή. Το contex το καταναλώνει το στιγμιότυπο που εκτελεί το query στην βάση και σε περίπτωση που το query υπερβεί τον απαιτούμενο χρόνο εκτέλεσης του, ακυρώνεται.

Παράδειγμα μιας συνάρτησης που εκτελεί ένα query στην βάση φαίνεται στον κώδικα 6.5. Η συνάρτηση InsertGame λαμβάνει τα δεδομένα για την δημιουργία ενός νέου game και εκτελεί το query για την εισαγωγή ενός νέο game δίνοντας παραμετρικά τα δεδομένα του game ώστε ο driver να συνθέσει ένα query με ασφαλή τρόπο. Σε περίπτωση που το context λήξει ή το query περιέχει κάποιο σφάλμα η συνάρτηση θα επιστρέψει ένα σφάλμα.

Το package store θα έχει ένα store struct για κάθε οντότητα της βάσης δεδομένων. Το struct θα περιέχει σαν μεθόδους τις συναρτήσεις για την ανάκτηση, την εισαγωγή και την ενημέρωση των δεδομένων για αυτή την οντότητα καθώς και ένα πεδίο για το στιγμιότυπο της σύνδεσης με την βάση δεδομένων. Τα store structs θα πρέπει να υλοποιούν ένα store interface. Ένα Go interface περιέχει τις δηλώσεις μιας ομάδας μεθόδων και όταν ένας τύπος μεταβλητής υλοποιεί όλες τις μεθόδους του interface μπορεί να χρησιμοποιηθεί σε μεταβλητές τύπου του interface που υλοποιεί.

Στον κώδικα 6.6 βλέπουμε δήλωση του interface GameStore και στον κώδικα 6.7 το struct PSQLGameStore με τις μεθόδους του. Το interface περιέχει την δήλωση των μεθόδων της οντότητας game και το struct περιέχει το στιγμιότυπο της σύνδεσης με την βάση δεδομένων PostgreSQL. Έχοντας ένα στιγμιότυπο PSQLGameStore μπορούμε να εκτελέσουμε τις συναρτήσεις του interface με την υλοποίηση για την βάση PostgreSQL. Αντίστοιχα θα μπορούσαμε να έχουμε ένα struct που υλοποιεί τις μεθόδους για διαφορετική βάση δεδομένων ή και μεθόδους που θα έχουν μια εικονική υλοποίηση για χρήσει στις δοκιμές (testing), απομονώνοντας την λειτουργία που καλεί το store. Στο κεφάλαιο 9 που αφορά τις δοκιμές του κώδικα θα αναλυθεί περισσότερο η ανάγκη των interface.

6.2.4 Email Handler

Για την αποστολή δυναμικών emails δημιουργήσαμε το package email. Στο συγκεκριμένο package υλοποιείται ένα interface EmailHandler και ένα struct SMTPEmailHandler που υλοποιεί το interface για την αποστολή emails μέσω τρίτου διακομιστή email. Το struct περιέχει σαν πεδία τα στοιχεία σύνδεσης του διακομιστή email, το store για την αποθήκευση δεδομένων σχετικά με τα email και μία μεταβλητή που δηλώνει αν χρειάζεται

αυθεντικοποίηση με τον διακομιστή. Στον κώδικα 6.8 και 6.9 φαίνεται η υλοποίηση του interface και του struct.

Μέσω της συνάρτησης NewSMTPEmailHandler δημιουργούμε ένα στιγμιότυπο του SMTPEmailHandler και στέλνουμε και ένα email στον διακομιστή για να βεβαιωθούμε ότι τα στοιχεία σύνδεσης είναι σωστά. Η υλοποίηση του υπάρχει στον κώδικα 6.10

Το struct SMTPEmailHandler υλοποιεί την μεθοδο SendEmail του interface EmailHandler για την αποστολή ενός προκαθορισμένου email για μία συγκεκριμένη κράτηση που παρέχονται παραμετρικά. Η SendEmail ανακτά από το store τις πληροφορίες της κράτησης, το πρότυπο email που πρέπει να χρησιμοποιηθεί, παράγει το email προς αποστολή μέσω της βοηθητικής συνάρτησης generateEmail, αποστέλει το email στο διακομιστή email και τέλος, καταγράφει στην βάση δεδομένων την προσπάθεια αποστολής του συγκεκριμένου email και το σφάλμα στην περίπτωση αποτυχίας του. Στον κώδικα 6.11 φαίνεται η υλοποίηση της μεθόδου SMTPEmailHandler.

6.2.5 Ενημέρωση ημερολογίου σε πραγματικό χρόνο

Για την ενημέρωση του πελάτη για αλλαγές στο calendar σε πραγματικό χρόνο το package game υλοποιεί έναν HTTP handler με Server-Sent Events ώστε να μπορεί να στέλνει δεδομένα στον πελάτη χωρίς να τα αιτηθεί. Μέσω του package store μπορούμε να παρακολουθούμε τον πίνακα κρατήσεων στην βάση δεδομένων για αλλαγές ώστε να ενεργοποιούμε τον handler να ενημερώνει τον πελάτη με το νέο calendar. Παρακάτω παρουσιάζεται η υλοποίηση των λειτουργιών στο package game και store.

Server-Sent Events

Το SSE είναι μια τεχνική που επιτρέπει σε ένα πρόγραμμα περιήγησης (πελάτη) να λαμβάνει αυτόματες ενημερώσεις από έναν διακομιστή μέσω HTTP. Ο handler που υλοποιεί την τεχνική SSE είναι ο GetCalendarSSE. Ο συγκεκριμένος handler πέρα από την συνηθισμένη λειτουργία που κάνει ένας handler ενημερώνει τον πελάτη μέσω του response header οτι θα στέλνει συνεχώς δεδομένα σαν συμβάντα και να κρατήσει την σύνδεση ανοιχτή. Έπειτα δημιουργεί το calendar μέσω βοηθητικής συνάρτησης και το στέλνει στον πελάτη γράφοντας το στο stream. Έπειτα μέσω ενός Go channel από το struct GameHandler περιμένει να λάβει σήμα ότι υπήρξε αλλαγή ώστε να ξανά δημιουργήσει και να στείλει το calendar μέχρι να

κλείσει η σύνδεση από τον πελάτη που την αιτήθηκε. Στον κώδικα 6.12 φαίνεται η υλοποίηση του SSE handler.

Για να ειδοποιηθεί ο handler οτι υπήρξε αλλαγή το struct GameHandler έχει την μέθοδο DispatchBookingUpdate που μπορεί να στέλνει το σήμα και μπορεί κάποιος να την καλέσει μέσω ενός στιγμιότυπου του GameHandler. Αυτή η συνάρτηση χρησιμοποιείτε από το notifier του store όταν υπάρξει κάποια αλλαγή στις κρατήσεις.

PSQL Notifier

Ο notifier είναι μία επιπλέον λειτουργία του store που τρέχει έναν listener στην βάση PostgreSQL. Συγκεκριμένα ο notifier συνδέεται σε ενα PostgreSQL channel και όταν λαμβάνει κάποιο συμβάν από αυτό θα τρέχει μία συνάρτηση που έχει λάβει παραμετρικά. Η υλοποίηση του notifier αποτελείται από μία συνάρτηση που λειτουργεί σαν constructor για το struct listener που περιέχει το struct pq.Listener από το driver της PostgreSQL για την Go και ένα Go channel για να λαμβάνει σήμα σφάλματος από την PostgreSQL. Η υλοποίηση τους φαίνεται στον κώδικα 6.13.

Μέσω ενός στιγμιότυπου του struct listener μπορούμε να καλέσουμε την μέθοδο ListenAndNotify που δέχεται ως παραμέτρους το όνομα του PostgreSQL channel και μια συνάρτηση callback. Η ListenAndNotify ξεκινάει να περιμένει συμβάντα από το συγκεκριμένο PostgreSQL channel και εκτελεί την συνάρτηση callback όταν λάβει ένα συμβάν με επιτυχία, αλλιώς στην περίπτωση σφάλματος καταγράφει το σφάλμα. Στον κώδικα 6.14 φαίνεται η υλοποίηση της μαζί με την βοηθητική συνάρτηση listen.

Στην βάση δεδομένων έχει δημιουργηθεί μια συνάρτηση που εκτελείται όταν υπάρξουν αλλαγές στον πίνακα κρατήσεων. Σε περίπτωση που δημιουργηθεί μια νέα κράτηση, ή αλλάξει η ημερομηνία μιας υπάρχουσας κράτησης ή ακυρωθεί μια κράτηση, στέλνετε μία ειδοποίηση στο PostgreSQL channel με όνομα bkng_changes_chan. Η συνάρτηση φαίνεται στον κώδικα 6.15.

6.2.6 Δημιουργία ημερολογιου κατάστασης κρατήσεων και διαθεσιμότητας

Αυτός ο μηχανισμός πρέπει να δημιουργεί την κατάσταση των κρατήσεων του game για έναν μήνα. Συγκεκριμένα θα λαμβάνει παραμετρικά τα ενεργά schedules, τις κρατήσεις του μήνα,

τον μήνα που χρειάζεται να υπολογίσει σαν αριθμό μηνών μακριά από των τρέχων μήνα, ζώνη ώρας και μια μεταβλητή που δηλώνει αν πρέπει να φαίνεται μόνο η διαθεσιμότητα ή και οι κρατήσεις. Η συγκεκριμένη συνάρτηση χρησιμοποιείται για να δημιουργηθεί ο πόρος calendar οπότε βρίσκεται στο package game για χρήση από τον gameHanlder.

6.2.7 Αυθεντικοποίηση χρήστη

Ο χρήστης μέσω του πόρου “/authentication” μπορεί να πιστοποιηθεί και να λάβει το JWT που θα χρησιμοποιεί τους πόρους περιορισμένης πρόσβασης. Για την δημιουργία του JWT θα χρησιμοποιήσουμε το package golang-jwt/jwt της τυπικής βιβλιοθήκης της Go. Στο package user υπάρχει ο HTTP handler AuthUser που πιστοποιεί τον χρήστη στην βάση δεδομένων και δημιουργεί ένα JWT με ημερομηνία λήξης και το user ID υπογεγραμμένο με το μυστικό το οποίο επιστρέφει μέσω του HTTP response. Η συνάρτηση που δημιουργεί το JWT φαίνεται στον κώδικα 6.16.

Όταν ένας χρήστης ζήτα έναν πόρο περιορισμενης πρόσβασης, παρέχει στο HTTP request το JWT τοποθετώντας το στο request header στο πεδίο Authorization που χρησιμοποιείται για αυτον τον σκοπό. Ο διακομιστής HTTP πριν καλέσει τον handler του πόρου τρέχει μια ενδιάμεση συνάρτηση middleware του package echo που επιβεβαιώνει ότι το JWT έχει εκδοθεί από εμάς μέσω του μυστικού, και οτι δεν έχει λήξει, επίσεις καλεί μία συνάρτηση του package user που εξάγει το user ID από το JWT και το καταχωρεί στο echo.context του αιτήματος για χρήση από τον handler. Στην περίπτωση που το JWT δεν είναι έγκυρο, επιστρέφει στον πελάτη ένα HTTP response με status 401 Unauthorized. Στον κώδικα 6.17 φαίνεται ή προσθήκη του JWT middleware στο στιγμιότυπο του διακομιστή και η συνάρτηση που εξάγει το user ID από ένα έγκυρο JWT.

6.2.8 Καταγραφή συμβάντων και σφαλμάτων

Η καταγραφή συμβάντων και σφαλμάτων του συστήματος είναι μία σημαντική λειτουργία που μας επιτρέπει να παρακολουθούμε και να επιβλέπουμε οτι το λογισμικό λειτουργεί σωστά. Για την καταγραφή υλοποιήσαμε το package logger που χρησιμοποιείται για την τυποποιημένη καταγραφή των συμβάντων και των σφαλμάτων ώστε να είναι εύκολη η προσπέλαση και η αναζήτηση τους, καθώς και να μπορεί να συνδεθεί με συστήματα αυτόματης παρακολούθησης για να βλέπουμε στατιστικά των επιδόσεων του λογισμικού αλλα και για άμεση ενημέρωση στην περίπτωση κάποιου κρίσιμου σφάλματος που κάνει το λογισμικό μη

λειτουργικό. Το package logger μέσω του εξωτερικού package sirupsen/logrus ορίζει τον τρόπο καταγραφής. Η καταγραφή γίνεται στο συστημα που τρέχει το λογισμικό μέσω της βασικής εξόδου του προγράμματος system out. Η καταγραφή μπορεί να αφορά πληροφορία για κάποια ενέργεια του συστήματος ή κάποιο σφάλμα. Η μορφή θα ξεκινάει με τον τύπο του συμβάντος , δηλαδη αν είναι πληροφορία ή σφάλμα, την ακριβή ώρα που έγινε και το μήνυμα που περιγράφει το συμβάν.

Παράδειγμα καταγραφή συμβάντος:

```
INFO[2023-01-30T23:22:27+02:00]  ::1 [2023-01-30T23:22:27+02:00]
localhost:8080 POST    /api/games/1/bookings 500 186 36  124.425584ms
PostmanRuntime/7.30.0
```

Παράδειγμα καταγραφή σφάλματος:

```
ERROR[2023-01-30T23:22:27+02:00] [sql: Scan error on column index 0, name
"exists": sql/driver: couldn't convert <nil> (<nil>) into type bool]
caller="gitlab.com/khakibee/khakibee/api/game.(*GameHandler).BookGame"
line=140
```

6.2.9 Μεταβλητές περιβάλλοντος διακομιστή

Το λογισμικό μπορεί να τρέχει σε οποιοδήποτε σύστημα ανεξάρτητα από τα υπόλοιπα στοιχεία του συστήματος κρατήσεων. Ορισμένες πληροφορίες για την λειτουργία του δεν μπορούν να υπάρχουν μέσα στον κώδικα αλλα θα πρέπει να τροφοδοτούνται στο λογισμικό κατά την εκτέλεση του. Τέτοιες πληροφορίες αφορούν ρυθμίσεις του λογισμικού ή και τα στοιχεία σύνδεσης στην βάση δεδομένων. Δηλαδή πληροφορίες που εξαρτώνται αποκλειστικά από το σύστημα που θα εκτελείται το πρόγραμμα και χρειάζονται μόνο κατα την εκτέλεση του. Αυτές οι πληροφορίες θα διατηρούνται στο σύστημα που εκτελείται το πρόγραμμα σαν μεταβλητές περιβάλλοντος ή σε κάποιο αρχείο ρυθμίσεων που το εκτελέσιμο πρόγραμμα γνωρίζει την τοποθεσία του.

Για να φορτώσει το λογισμικό τις μεταβλητές περιβάλλοντος υλοποιήσαμε το package config. Το package config είναι υπεύθυνο να ενημερώσει τις μεταβλητές περιβάλλοντος με το αρχείο ρυθμίσεων, αν υπάρχει, και να διαβάσει από το σύστημα τις τιμές τους μέσω του ονόματος τους. Το package config υλοποιεί το struct Config που έχει σαν πεδία τις μεταβλητές περιβάλλοντος που χρειάζονται και την μέθοδο LoadEnv που διαβάζει τις μεταβλητές από το σύστημα. Στον κώδικα 6.18 φαίνεται η υλοποίηση του package config.

6.2.10 Δημιουργία Διακομιστή HTTP

Όπως περιγράψαμε παραπάνω το λογισμικό θα είναι ένας διακομιστής HTTP που θα λειτουργεί ως ενδιάμεσος για την βάση δεδομένων. Δηλαδή θα περιμένει να λάβει αιτήματα HTTP από τον πελάτη, θα εκτελεί τις απαιτούμενες ενέργειες και θα επιστρέφει μια απάντηση HTTP. Επίσης ο διακομιστής θα μπορεί να συντάσει και να στέλνει email μέσω διακομιστή email τρίτου παρόχου.

Το package main είναι η εκτελέσιμη εντολή του λογισμικού. Εδώ θα φορτώνονται οι μεταβλητές συστήματος, θα δημιουργούνται τα στιγμιότυπα των λειτουργιών του διακομιστή, όπως οι HTTP handlers και τα stores, θα ξεκινάει ο διακομιστής HTTP και θα γίνεται η ανάθεση του http handlers στον διακομιστή HTTP. Από αυτό το package θα παράγεται το εκτελέσιμο πρόγραμμα.

Καθώς το web service αφορά τρεις διαφορετικούς χρήστες με διαφορετική χρήση των υπηρεσιών, θα παράξουμε τρία διαφορετικά εκτελέσιμα προγράμματα. Με αυτόν τον τρόπο το κάθε εκτελέσιμο θα επιτρέπει την χρήση των υπηρεσιών μόνο για τον χρήστη που προορίζεται για επιπλέον ασφάλεια και καθώς οι απαιτήσεις των χρηστών είναι διαφορετικές και τα εκτελέσιμα μπορούν να τρέχουν σε διαφορετικά συστήματα. Για παράδειγμα το εκτελέσιμο που προορίζεται για τον πελάτη θα δέχεται περισσότερα αιτήματα από το εκτελέσιμο που προορίζεται για τον διαχειριστή καθώς οι πελάτες θα είναι περισσότεροι από τους διαχειριστές, οπότε το εκτελέσιμο του πελάτη μπορεί να χρειάζεται περισσότερους πόρους συστήματος για καλύτερη επίδοση. Κάθε εκτελέσιμο θα έχει το δικό του package main για να συνθέτει τον διακομιστή.

Μεταβλητές περιβάλλοντος

Όταν εκτελεστεί το πρόγραμμα, αρχικά μέσω του package config πρέπει να φορτώσει τις μεταβλητές περιβάλλοντος καθώς θα χρειαστούν παρακάτω για την ρύθμιση του διακομιστή.

Σύνδεση στην βάση δεδομένων

Έπειτα θα ανοίγει μία σύνδεση με την βάση δεδομένων χρησιμοποιώντας το package database/sql της τυπικής βιβλιοθήκης Go και θα επιβεβαιώνει την σωστή επικοινωνία με την βάση. Σε περίπτωση που η σύνδεση δεν είναι επιτυχής, το πρόγραμμα θα σταματάει την

λειτουργία του καθώς ο σκοπός του λογισμικού είναι να λειτουργεί σαν ενδιάμεσος μεταξύ άλλων λογισμικών και της βάσης δεδομένων.

Προσθήκη λειτουργιών διακομιστή

Ο διακομιστής ανάλογα με τον χρήστη που θα προορίζεται θα έχει διαφορετικές λειτουργίες.

Λειτουργίες πελάτη: Ο πελάτης θα μπορεί να ανακτά τα games, το calendar με την διαθεσιμότητα του χώρου και να δημιουργεί κρατήσεις. Επίσης οι υπηρεσίες που προορίζονται για τον πελάτη δεν θα χρειάζονται πιστοποίηση. Για αυτές τις λειτουργίες χρειάζεται τα στιγμιότυπα για το PSQLGameStore, το PSQLBookingStore και το PSQLScheduleStore και ένα στιγμιότυπο για το GameHandler.

Λειτουργίες χρήστη τρίτων υπηρεσιών: Ο χρήστης τρίτων υπηρεσιών θα μπορεί να ανακτά τα games, το calendar με την διαθεσιμότητα του χώρου και τις κρατήσεις που έχει δημιουργήσει ο ίδιος και να δημιουργεί κρατήσεις. Επίσης θα μπορεί να πιστοποιηθεί για την απόκτηση JWT ώστε να έχει πρόσβαση στις υπηρεσίες. Για αυτές τις λειτουργίες χρειάζεται τα στιγμιότυπα για το PSQLGameStore, το PSQLBookingStore, το PSQLScheduleStore και το PSQLUserStore και τα στιγμιότυπα για το GameHandler και UserHandler.

Λειτουργίες διαχειριστή: Ο χρήστης διαχειριστής θα έχει πλήρη πρόσβαση στις λειτουργίες του διακομιστή. Επίσης θα μπορεί να πιστοποιηθεί για την απόκτηση JWT ώστε να έχει πρόσβαση στις υπηρεσίες. Οπότε χρειάζεται τα στιγμιότυπα για όλα τα stores και για όλους τους HTTP handlers.

Πέρα από τις λειτουργίες των χρηστών, οι διακομιστές χρειάζονται το Email Handler για την αποστολή email καθώς και την ενεργοποιηση του store.Notifier για την λειτουργία του SSE

Ρύθμιση διακομιστή HTTP

Η τυπική βιβλιοθήκη της Go παρέχει packages για την δημιουργία ενός διακομιστή HTTP αλλα καθώς το package της τυπικής βιβλιοθήκης είναι χαμηλού επιπέδου, επιλέξαμε να χρησιμοποιήσουμε το Echo της labstack. Το Echo είναι ένα Go web framework που παρέχει μεθόδους για την δημιουργία του διακομιστή HTTP καθώς και για διαχείριση των response/request HTTP. Η χρήση του Echo μας βοηθά να χτίσουμε γρηγορότερα ένα REST service μέσω των μεθόδων του και παρέχει περισσότερο ασφάλεια στην συγγραφή κώδικα

καθώς έχει ήδη λύσει τα σημαντικότερα προβλήματα της δημιουργίας μιας διεπαφής χρήστη. Η λειτουργία του Echo θα είναι να λαμβάνει όλα τα αιτήματα HTTP. Να αναγνωρίζει το URI και να καλεί την αντίστοιχη συνάρτηση HTTP handler που μπορεί να διαχειριστεί το συγκεκριμένο αίτημα παρέχοντας του όλη την πληροφορία του αιτήματος. Και τέλος να συντάσει την απάντηση HTTP και να την στέλνει στον πελάτη. Όταν το echo λαμβάνει ένα HTTP request δημιουργεί ένα στιγμιότυπο echo.Context. Το echo.Context περιέχει το HTTP request, μεθόδους για την διαχείριση του HTTP request καθώς και για την αποστολή του HTTP response στον πελάτη, τον μέγιστο χρόνο που πρέπει να ολοκληρωθεί το αίτημα καθώς και πληροφορίες που μπορούμε να εισάγουμε εμείς μέσω ενδιάμεσων συναρτήσεων middlewares. Η δημιουργία του διακομιστή HTTP ξεκινάει δημιουργώντας ένα στιγμιότυπο echo.Echo. Στο στιγμιότυπο echo αναθέτουμε ένα στιγμιότυπο του logger να καταγράφει συμβάντα και σφάλματα του συστήματος χρησιμοποιώντας το δικό μας package. Επίσης του αναθέτουμε ένα στιγμιότυπο του package validator της τυπικής βιβλιοθήκης της Go για να επικυρώνει τα δεδομένα που στέλνει ο πελάτης με το HTTP request. Επίσης σε αυτό το σημείο μπορούν να ανατεθούν συναρτήσεις middleware. Middleware είναι συναρτήσεις που παίρνουν παραμετρικά το echo.Context και εκτελούνται πριν τον HTTP handler για κάθε HTTP request. Οι συναρτήσεις Handler πρέπει να αναθέτονται στο echo μαζί με την μέθοδο και το αναγνωριστικό του πόρου. Επίσεις το echo επιτρέπει την ομαδοποίηση πόρων με βάση το αναγνωριστικό τους. Στον κώδικα 6.25 βλέπουμε ένα παράδειγμα ανάθεσης των πόρων του game στο group api. Το API είναι ένα στιγμιότυπο του echo τύπου *echo.Echo που αναθέτουμε τους handlers. Στο group /games αναθέτουμε τους handlers στην κατάλληλη μέθοδο για κάθε αναγνωριστικό πόρου. Όταν θέλουμε το αναγνωριστικό να ταιριάζει με περισσότερους από έναν πόρο, χρησιμοποιούμε το σύμβολο “:” έτσι το “GET /games/:gid/” μπορεί να ταιριάξει με τα αναγνωριστικά GET /games/1 ή GET /games/2 και ο handler θα εξάγει το ID ώστε να επιστρέψει τον σωστό πόρο. Όταν το URI του HTTP response ταιριάζει με το αναγνωριστικό κάποιο πόρου, καλείτε η συνάρτηση handler και λαμβάνει παραμετρικά το echo.Context.

Έπειτα μπορούμε να ξεκινήσουμε τον διακομιστή HTTP στην επιλεγμένη πόρτα του συστήματος.

Κεφάλαιο 7ο

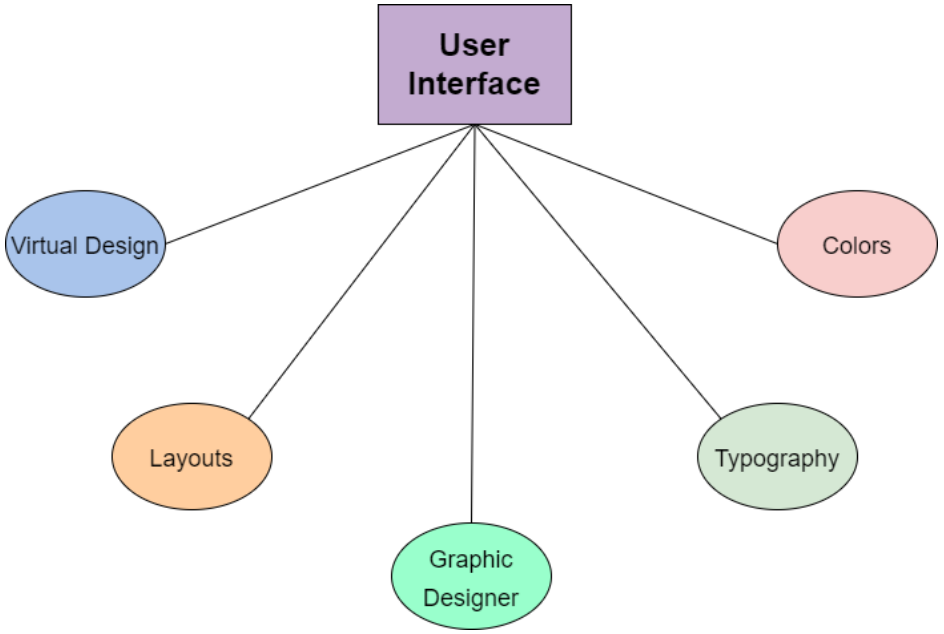
Σχεδίαση και Υλοποίηση του User Interface

Η διεπαφή χρήστη (UI) αποτελεί στον σύγχρονο κόσμο των εφαρμογών ένα αναπόσπαστο κομμάτι για την ανάπτυξη μιας εφαρμογής, καθώς έχουν αυξηθεί τόσο οι ανάγκες των χρηστών αλλά και οι απαιτήσεις τους στο οπτικό αποτέλεσμα μιας εφαρμογής. Η πλευρά που αλληλεπιδρά ο χρήστης είναι από τα πιο σημαντικά κομμάτια για μία εφαρμογή και γι αυτό έχουν δημιουργηθεί τομείς όπως είναι η διεπαφή χρήστη (UI) και η εμπειρία χρήστη (UX). Στα πλαίσια της διπλωματικής θα σχεδιαστεί και υλοποιηθεί το κομμάτι της διεπαφή χρήστη για να δώσουμε στους χρήστες τη μέγιστη εμπειρία.

7.1 Σχεδίαση Διεπαφής Χρήστη (UI)

Η διεπαφή χρήστη (UI) είναι ο χώρος όπου πραγματοποιούνται οι αλληλεπιδράσεις μεταξύ των ανθρώπων (χρηστών) και των μηχανών. Πιο συγκεκριμένα θα μπορούσαμε να πούμε ότι είναι ο τρόπος με τον οποίο ο χρήστης αλληλεπιδρά και ελέγχει μια συσκευή ή μια εφαρμογή. Η διεπαφή χρήστη θα πρέπει να διατηρεί κάποια χαρακτηριστικά όπως η διαισθητικότητα και η ευχρηστία, επιτρέποντας στον χρήστη να έχει εύκολη πρόσβαση και να χρησιμοποιεί τις λειτουργίες της συσκευής ή της εφαρμογής. Τα στοιχεία που αποτελούν μια εφαρμογή όπως είναι τα κουμπία, τα εικονίδια, το μενού κλπ, αποτελούν κομμάτια της διεπαφής της εφαρμογής και ο καλός σχεδιασμός τους θα μας δώσει ένα εύχρηστο αποτέλεσμα και την ικανοποίηση των χρηστών μας. Για το λόγο αυτό οι χρωματισμοί τις εφαρμογής μας και το στυλ της επιλέχθηκαν με κριτήρια απλότητας και διαδικτυακής συνήθειας των χρηστών που έχουν υιοθετήσει από μεγάλες εφαρμογές μέχρι τώρα. Στα πλαίσια του δικού μας συστήματος κρατήσεων σχεδιάσαμε ο χρήστης να μπορεί να εισάγει τις επιθυμητές ημερομηνίες για την κράτηση του και να μπορεί να περιηγηθεί ανάμεσα στα διαθέσιμα δωμάτια απόδρασης της κάθε επιχείρησης και να επιλεξει το επιθυμητό για αυτόν. Οι οδηγίες για την πλοήγηση, την επιλογή δωματίου και τελικά την ολοκλήρωση μιας κράτησης είναι σαφείς και ακολουθούν μία λογική ροή. Η εφαρμογή μας προκειμένου να ανταποκρίνεται στις προκλήσεις της τεχνολογίας σχεδιάστηκε ώστε να λειτουργεί ομαλά σε διάφορες συσκευές, όπως φορητούς υπολογιστές, τάμπλετ και κινητά τηλέφωνα. Η ευρεία χρήση των κινητών συσκευών από την

πλειοψηφία των χρηστών, μας ώθησε στο να σχεδιάσουμε την εφαρμογή μας με βάση την ανταπόκριση και σε τέτοιου είδους συσκευές (responsive).



Εικόνα 7.1 Διακριτά στοιχεία της διεπαφής χρήστη

7.2 Προσθήκη Bootstrap

Προκειμένου η εφαρμογή μας να αναπτυχθεί εύκολα και με μεγάλη ταχύτητα επιλέχθηκε η χρήση της Bootstrap. Η Bootstrap είναι ένα styling framework με έτοιμες κλάσεις, στοιχεία και πολλά άλλα που έχουν ήδη σχεδιαστεί για εμάς. Αυτό μας δίνει τη δυνατότητα να χρησιμοποιήσουμε τους γενικούς κανόνες του για την ανάπτυξη και παραμετροποίηση όσων στοιχείων του UI υιοθετήσουμε. Με τη βοήθεια της Bootstrap αναπτύξαμε τη γενικότερη δομή (layout) του συστήματος κρατήσεων μας καθώς και πιο απλά σημεία, όπως φόρμες, κουμπιά, κάρτες, κλπ. Στα σημεία που χρειάστηκε μεγάλη παραμετροποίηση αναπτύχθηκε εξ ολοκλήρου CSS κανόνων για να αποδώσει το επιθυμητό αποτέλεσμα στον χρήστη.

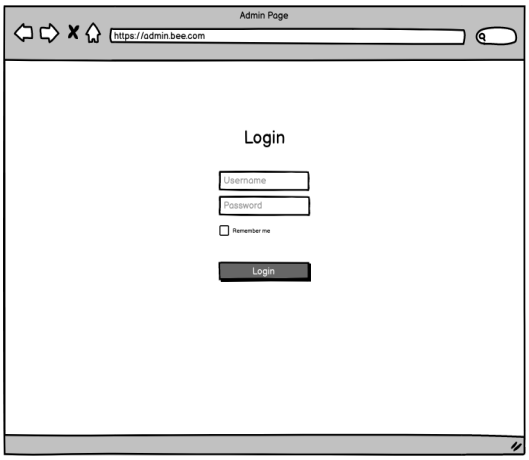
7.3 Δημιουργία εφαρμογής διαχείρισης κρατήσεων

Προκειμένου να αναπτυχθεί η εφαρμογή μας στο front-end κομμάτι χρησιμοποιήθηκαν οι HTML, CSS με την βοήθεια του framework Bootstrap για να δομηθεί και στοιχηθεί η διαδικτυακή εφαρμογή διαχείρισης κρατήσεων. Για να προστεθεί λειτουργικότητα και διαδραστικότητα στην εφαρμογή μας έγινε χρήση της React (JavaScript Library). Μέσω ενός

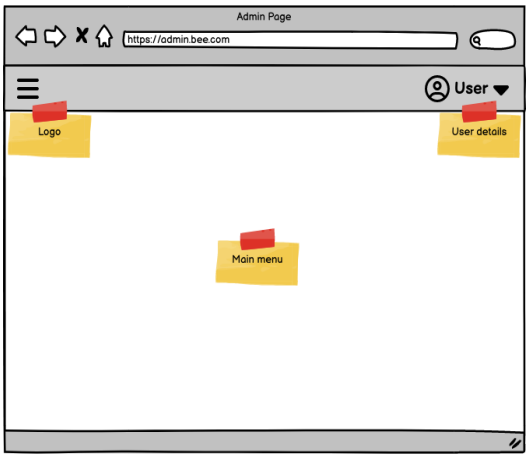
package manager έγινε η προσθήκη ενός NodeJS server που θα μας βοηθήσει να τρέχουμε την εφαρμογή μας σε περιβάλλον ανάπτυξης. Αφού εγκαταστήσαμε τα απαραίτητα πακέτα για την React ξεκινήσαμε με μια εντολή που μας δίνεται έτοιμη 'npm start' μία αρχικοποιημένη React εφαρμογή. Η React κάνει αρχικοποιήσει κάποιον αρχείων και μας παρουσιάζει μια αρχική εφαρμογή React, από κει και πέρα αναλαμβάνουμε εμείς να χτίσουμε τα κατάλληλα μέρη (components) της εφαρμογής μας, σύμφωνα με την σχεδίαση που έγινε παραπάνω.

7.3.1 Διαμόρφωση layout και διάταξη στοιχείων

Σε αυτό το σημείο καλούμαστε να διαμορφώσουμε τη γενική δομή (layout) της εφαρμογής και την διάταξη της σύμφωνα με τις ανάγκες που έχουν αναλυθεί. Το πρώτο κομμάτι που πρέπει να κατασκευάσουμε είναι η σελίδα εισόδου της εφαρμογής (Login page), δηλαδή μία σελίδα που θα είναι υπεύθυνη να λάβει τα κατάλληλα δεδομένα και να δώσει το δικαίωμα εισόδου στους χρήστες, στο μενού της εφαρμογής. Κατόπιν επιτυχούς εισόδου στην εφαρμογή, θα κατασκευάσουμε το βασικό πλαίσιο μέσα στο οποίο θα παρουσιάζεται το περιεχόμενο της εφαρμογής μας. Πιο συγκεκριμένα θα πρέπει να κατασκευάσουμε μία μπάρα στο πάνω μέρος της εφαρμογής όπου στα δεξιά θα υπάρχουν πληροφορίες σχετικά με τον χρήστη που έχει συνδεθεί και διάφορες επιλογές σχετικά με αυτόν. Η άλλη πλευρά της μπάρας θα φιλοξενήσει το λογότυπο της εκάστοτε επιχείρησης και το όνομα της ώστε να προσδίδεται η προσωποποίηση στους πελάτες της εφαρμογής. Στο κάτω μέρος θα βρίσκεται το βασικό περιεχόμενο της εφαρμογής και θα εναλλάσσεται από τις επιλογές του χρήστη. Το βασικό περιεχόμενο θα εναπόκειται στην επιλογή δωματίου καθώς όλες οι λειτουργίες έχουν ώς σημείο έναρξης ένα δωμάτιο ψυχαγωγίας.



Εικόνα 7.2 Είσοδος στην εφαρμογή



Εικόνα 7.3 Στοίχιση βασικής σελίδας

7.3.2 Δρομολόγηση ιδιωτικών και δημόσιων μονοπατιών

Ένα βασικό στοιχείο στην εφαρμογή μας είναι να μπορούν να έχουν πρόσβαση σε συγκεκριμένες σελίδες της διαχειριστικής σελίδας μόνο όσοι χρήστες είναι διαπιστευμένοι με τα στοιχεία εισόδου τους. Για τον λόγο αυτό χρειάζεται ένας μηχανισμός με τον οποίο θα μπορούν να προστατεύονται το κομμάτια εκείνα της εφαρμογής τα οποία δεν πρέπει να έχουν πρόσβαση κάποιος που δεν είναι διαπιστευμένος χρήστης. Το βασικό και πρωταρχικό παράδειγμα στην εφαρμογή μας είναι το βασικό μενού (main menu), πριν κάποιον μπορέσει να δει την αρχική σελίδα θα πρέπει να έχει εισάγει το συνθηματικό και τον κωδικό του και αφού έχει πάρει το κατάλληλο διαπιστευτήριο (token), θα μπορεί να πλοηγηθεί στις υπόλοιπες σελίδες. Στην εφαρμογή μας έγινε χρήση ενός εργαλείου που συνδυάζεται πολλές φορές με εφαρμογές React και είναι το react-router-dom. Το εργαλείο αυτό μας δίνει την δυνατότητα για καλύτερη και ευκολότερη διαχείριση των μονοπατιών της διαχειριστικής μας σελίδας, Στον κώδικα 7.1 θα δούμε ένα παράδειγμα της χρήσης των μονοπατιών.

Όπως βλέπουμε ο πρώτος κανόνας των μονοπατιών είναι ελεύθερος για όλους τους χρήστες καθώς δεν υπάρχει κάποιος περιορισμός. Στην συνέχεια βλέπουμε στο επόμενο μεγάλο μονοπάτι ένα ιδιωτικό κανόνα που περικλείει όλα τα άλλα μονοπάτια τα οποία θα πρέπει να προστατεύονται από τους μη διαπιστευμένους χρήστες. Όπως για παράδειγμα το μονοπάτι που οδηγεί στα παιχνίδια (**/games**) είναι ένα προστατευμένο μονοπάτι που έχουν πρόσβαση μόνο όσοι έχουν συνδεθεί και λάβει το αντίστοιχο αναγνωριστικό (token). Σε περίπτωση που κάποιος επιχειρήσει να το προσπελάσει μέσω ενός περιηγητή ή ακόμη και απευθείας μέσω των APIs η εφαρμογή μας θα τον επιστρέψει στην σελίδα εισόδου ή θα λάβει αρνητική απάντηση.

7.3.3 Διαχωρισμός του UI σε μέρη (components)

Η React όπως είναι γνωστό είναι μια βιβλιοθήκη της JavaScript που στηρίζεται κατα κύριο λόγο στην δημιουργία στοιχείων του κώδικα (components) που μαζί συνθέτουν μία μεγαλύτερη οντότητα. Τα components αποτελούν θεμελιώδες στοιχείο επομένως και του συστήματος κρατήσεων μας, πράγμα που μας οδήγησε στο καταμερισμό μεγαλύτερων στοιχείων της εφαρμογής σε μικρότερα. Ο καταμερισμός αυτός προσθέτει πληθώρα πλεονεκτημάτων όπως είναι ο ευκολότερος τρόπος διαβάσματος του κώδικα, η μελλοντική αναβάθμιση του, ο έλεγχος του από άλλον άνθρωπο (manual testing) ή από σενάρια unit και integration tests. Έχοντας το σύστημα κρατήσεων μας ώς αναφορά θα αναλύσουμε ένα σημαντικό στοιχείο της εφαρμογής μας που είναι τα δωμάτια ψυχαγωγίας ή αλλιώς παιχνίδια (games). Δημιουργούμε ένα αρχείο εν ονόματι Games.js μέσα στο οποίο γίνεται μία HTTP

κλήση ώστε να λάβουμε την απαραίτητη πληροφορία από τον διακομιστή για τα διαθέσιμα δωμάτια/παιχνίδια που υπάρχουν στη βάση μας για τη συγκεκριμένη επιχείρηση. Προκειμένου να χρησιμοποιήσουμε το πλεονέκτημα του κατακερματισμού σε μικρότερα components ώστε κάθε component να πραγματοποιεί κατά κύριο λόγο μόνο μία ενέργεια, με τα δεδομένα που θα λάβουμε από την HTTP κλήση θα τροφοδοτησουμε ένα αλλο component το οποίο θα είναι υπεύθυνο για την παρουσίαση της αναγκαίας πληροφορίας του εκάστοτε δωματίου. Επειδή τα δωμάτια όπως μπορεί να είναι περισσότερα του ενός, για αυτό το λόγο το κάνουμε μία επαναληπτική διαδικασία και τροφοδοτούμενο το component μας που του δίνουμε και το όνομα gameList. Τα ονόματα είναι σημαντικά στην ανάπτυξη του κώδικα και γι αυτό θα πρέπει να δηλώνουν ακριβώς το τι κάνει το κάθε στοιχείο - component. Τα παραπάνω αποτυπώνονται και στην εικόνα από τον κώδικα, με ποίον τρόπο δομήθηκε και χωρίστηκε για να κερδίσουμε όλα τα πλεονεκτήματα που μας προσφέρει η React.

Κάθε νέα μεγάλη οντότητα του συστήματος κρατήσεων θα δομηθεί με αυτό το μοτίβο, για τον μέγιστο καταμερισμό των αρμοδιοτήτων ανά στοιχείο - component. Το πατρικό component όλης της εφαρμογής αξίζει να σημειωθεί ότι είναι το App.js και πάνω σε αυτό εν συνεχεία δομούνται όλα τα υπόλοιπα στοιχεία.

7.3.4 Δημιουργία βοηθητικών συναρτήσεων

Οι βοηθητικές συναρτήσεις στο σύστημα κρατήσεων μας και πιο συγκεκριμένα στην διαχειριστική σελίδα παίζουν μεγάλο ρόλο στο κομμάτι της επαναχρησιμοποίησης. Καθ όλη την διάρκεια της ανάπτυξης της εφαρμογής χρειάστηκε πολλές φορές η χρήση λειτουργιών για να βοηθούν στην επίλυση προβλημάτων ή να δίνουν ένα συγκεκριμένο αποτέλεσμα. Βασικό παράδειγμα για το σύστημα κρατήσεων μας, σε διάφορα σημεία χρειάζεται η μετατροπή μιας ημερομηνίας σε ένα συγκεκριμένο φορμάτ, προκειμένου να μην χρειάζεται συνεχώς να ξαναγράφεται το ίδιο ή παρόμοιο κομμάτι κώδικα, απομονώνουμε αυτές τις γραμμές του κώδικα να κάνουν μόνο μία λειτουργία. Αυτήν την συνάρτηση εν συνεχεία την καλούμε όπου είναι απαραίτητη και έτσι κάνουμε χρήση βοηθητικών συναρτήσεων στην εφαρμογής μας πράγμα που αποτελεί και βέλτιστη πρακτική για την ανάπτυξη κώδικα.

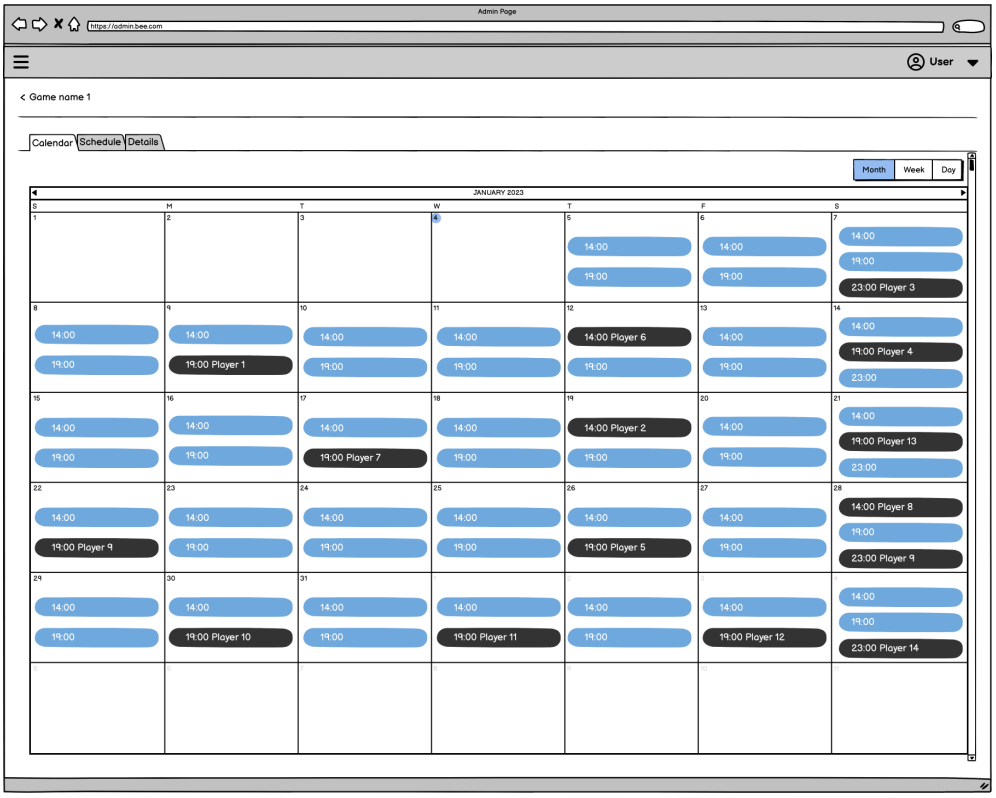
Εικόνα 7.4 Δομή Parent-child component

7.4 Δημιουργία Ημερολογίου διαχείρισης κρατήσεων

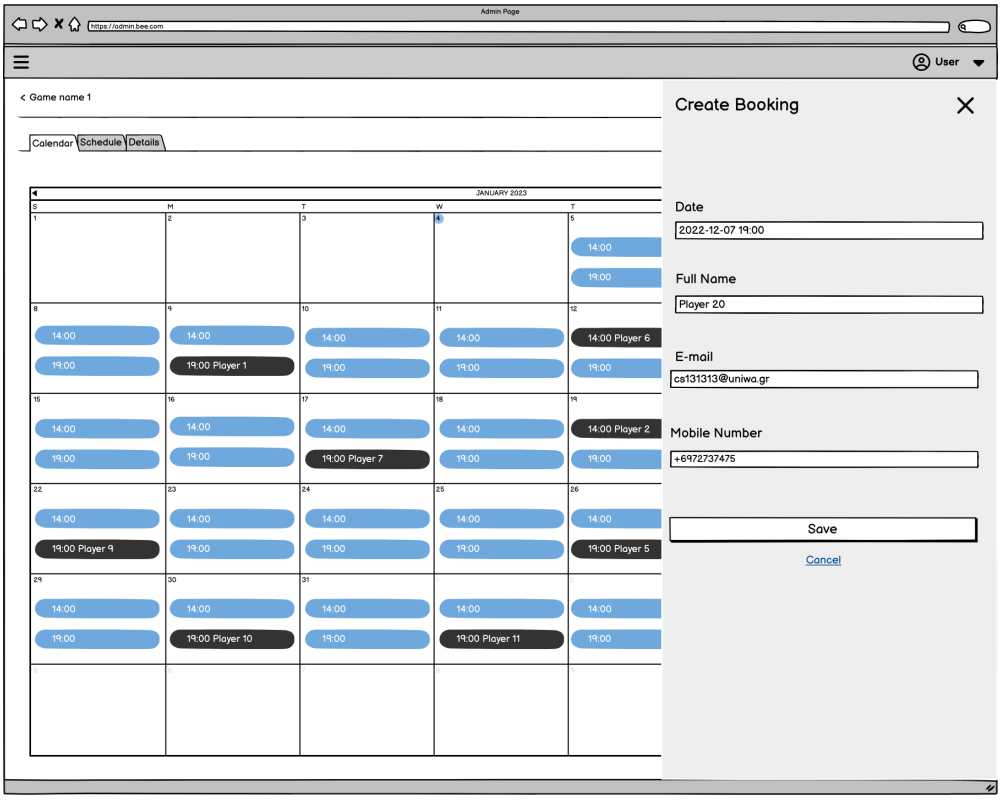
Σύμφωνα με τις λειτουργικές ανάγκες των χρηστών της διαχειριστικής σελίδας, διαπιστώθηκε ότι είναι απαραίτητη η δημιουργία ενός ημερολογίου παραμετροποιημένο σύμφωνα με τις ανάγκες των χρηστών μας. Το ημερολόγιο αυτό θα πρέπει να παρουσιάζει στον διαχειριστή του δωματίου εύκολα και γρήγορα την συνολική εικόνα του δωματίου σε μηνιαία, εβδομαδιαία και ημερήσια βάση αλλά και να μπορεί να δίνει λεπτομερείς πληροφορίες ανά κράτηση. Ξεκινήσαμε σχεδιάζοντας το ημερολόγιο ώς μια ξεχωριστή οντότητα (component)

επαναχρησιμοποιήσιμη που θα μπορέσουμε να την εντάξουμε στο frontend. Σύμφωνα και με την εικόνα 7.5 βλέπουμε όλα τα διαθέσιμα και κλεισμένα ραντεβού (με σκούρο χρώμα), καθώς και επιλογές για να επιλέξουμε πως θέλουμε να βλέπουμε ένα συγκεκριμένο εύρος ημερομηνιών, για παράδειγμα θέλουμε να επιλέξουμε εβδομαδιαίο και την τρίτη εβδομάδα του τρέχοντος μήνα. Αυτές οι επιλογές σχεδιάστηκαν για να προσφέρουν απλότητα και ευχρηστία στον διαχειριστή του συστήματος. Επιπλέον θα μπορεί πατώντας πάνω σε ένα κλεισμένο ραντεβού να ανοίξει μία καρτέλα στα δεξιά του και δει όλες τις πληροφορίες για την συγκεκριμένη κράτηση, είτε να πραγματοποιησει κάποια τροποποίηση σε αυτήν. Σε οποιοδήποτε άλλο ελεύθερο ραντεβού πατήσει του εμφανίζεται μία καρτέλα στα δεξιά με κενά πεδία που θα μπορεί να συμπληρώσει ο διαχειριστής για την πραγματοποιήση μιας νέας κράτησης.

Για την δημιουργία και την κατασκευή του ημερολογίου μας χρησιμοποιήσαμε σαν οδηγό μια open source βιβλιοθήκη που ονομάζεται Big Calendar που στηρίζεται στο FullCalendar.io και ασχολείται με την ανάπτυξη ημερολογίων παντός τύπου. Διαθέτει πληθώρα παραδείγματα για την δημιουργία ενός ημερολογίου και διαφόρων στοιχείων που μπορούν να προστεθούν στο δικό μας σύστημα. Προκειμένου να φτάσουμε στην τελική μορφή του ημερολογίου αναπτύξαμε κώδικα που να ανταποκρίνεται στις ανάγκες των δωματίων ψυχαγωγίας καθώς και επηρεάσαμε στυλιστικά το ημερολόγιο ώστε να το κάνουμε πιο εύχρηστο για τον χρήστη. Το ημερολόγιο δομήθηκε σε ένα δικό του component ώστε να μπορεί να επαναχρησιμοποιηθεί σε μελλοντικές επεκτάσεις, έγινε ουσιαστικά wrapped από τον υπάρχον ημερολόγιο του Big Calendar με τα δικά μας χαρακτηριστικά (λειτουργικά και στυλιστικά). Στο component του ημερολογίου περνάμε σαν μεταβλητές όλα τα απαραίτητα δεδομένα που χρειάζεται για να μας παρουσιάζει κάθε στιγμή πιθανές αλλαγές στις κρατήσεις του ημερολογίου. Ένα άλλο σημείο που μπορούμε να τονίσουμε είναι οι μεγάλες δυνατότητες παραμετροποίησης που μπορούμε να έχουμε στο ημερολόγιο της εφαρμογής μας καθώς μπορούμε να ορίσουμε το εύρος ζώνης ώρας, το φορμάτ των ημερομηνιών και πολλά άλλα που δίνουν την ευελιξία στο σύστημα και κατ’ επέκταση στο διαχειριστή να το προσαρμόσει στις ανάγκες της επιχείρησης του. Μια βιβλιοθήκη που μας βοήθησε σημαντικά σε αυτό το κομμάτι είναι το Moment.js.



Εικόνα 7.5 Σχεδιασμός ημερολογίου κρατήσεων διαχειριστικής σελίδας



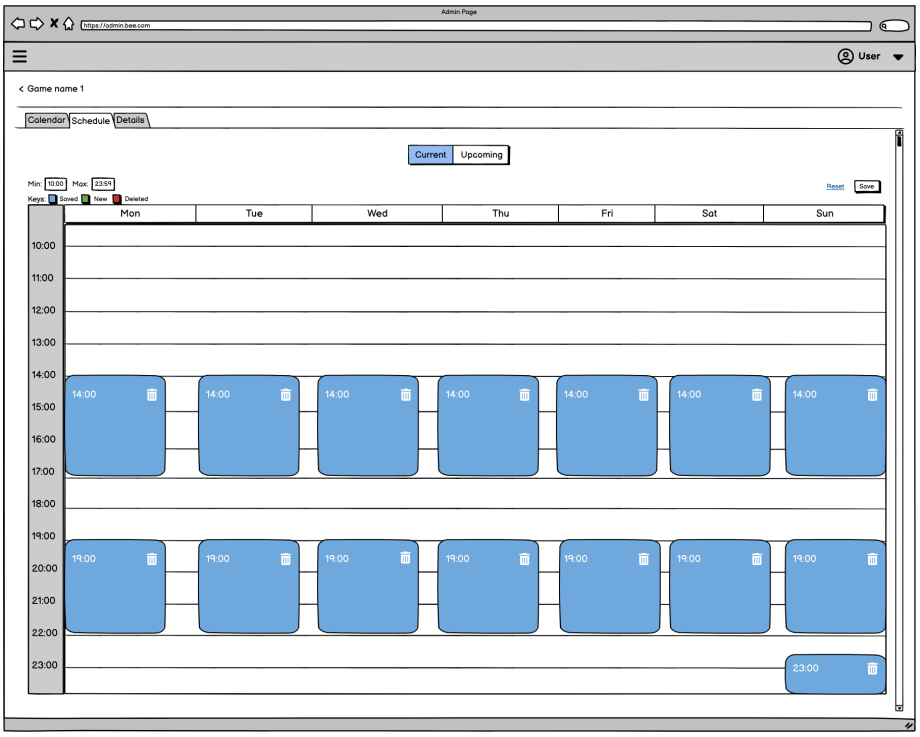
Εικόνα 7.6 Δημιουργία κράτησης μέσω της διαχειριστικής σελίδας

7.5 Δημιουργία Ημερολογίου διαχείρισης ωραρίου λειτουργίας

Σύμφωνα με τις λειτουργικές ανάγκες των χρηστών της διαχειριστικής σελίδας, μια εξίσου απαραίτητη λειτουργία είναι η δημιουργία ενός ημερολογιακού καμβά για την διαχείριση του ωραρίου λειτουργίας της επιχείρησης. Οι διαθέσιμες και μη κρατήσεις ενός δωματίου στο σύστημα κρατήσεων μας στηρίζονται εξ ολοκλήρου στο τρέχων πρόγραμμα λειτουργίας της επιχείρησης, πράγμα που το καθιστά πολύ σημαντικό για την λειτουργία ολόκληρης της επιχείρησης. Προκειμένου να δώσουμε στον διαχειριστή την δυνατότητα να παραμετροποιεί τις ανάγκες του εκάστοτε δωματίου δημιουργήσαμε ένα wrapped component με βάση το Big Calendar (υλοποίηση σε React JS) που στηρίζεται στη βιβλιοθήκη του FullCalendar.io. Μέσω των δυνατοτήτων που μας παρέχει η βιβλιοθήκη εκμεταλευτήκαμε λειτουργίες όπως το Drag&Drop και συναρτήσεις που μας βοηθούν στην δημιουργία γεγονότων πάνω στον ημερολογιακό καμβά και δημιουργήσαμε ένα εβδομαδιαίο ημερολογιακό καμβά που θα ορίζεται το πρόγραμμα του δωματίου βάση της διάρκειας του ραντεβού κάθε δωματίου. Η διάρκεια της εμπειρίας του εκάστοτε δωματίου ψυχαγωγίας ορίζεται από τον διαχειριστή κατά την δημιουργία του, έτσι το πρόγραμμα και οι κρατήσεις θα εναπόκειται στην διάρκεια αυτή. Ο διαχειριστής θα έχει την δυνατότητα να δίνει στο πρόγραμμα την ελάχιστη ώρα έναρξης και την λήξης ώστε να περιορίσει το εύρος στον καμβά. Επίσης έχει προβλεφθεί μέσω συναρτήσεων που μας παρέχονται από την βιβλιοθήκη του Big Calendar να μην μπορεί κάποιο ραντεβού να επικαλύπτει μερικώς ή ολικώς κάποιο άλλο, αυτό εξορισμού του είναι λάθος και μέσω ελέγχων και συναρτήσεων που ελέγχουν την κίνηση των δημιουργημένων κρατήσεων στον καμβά αποφεύγεται.

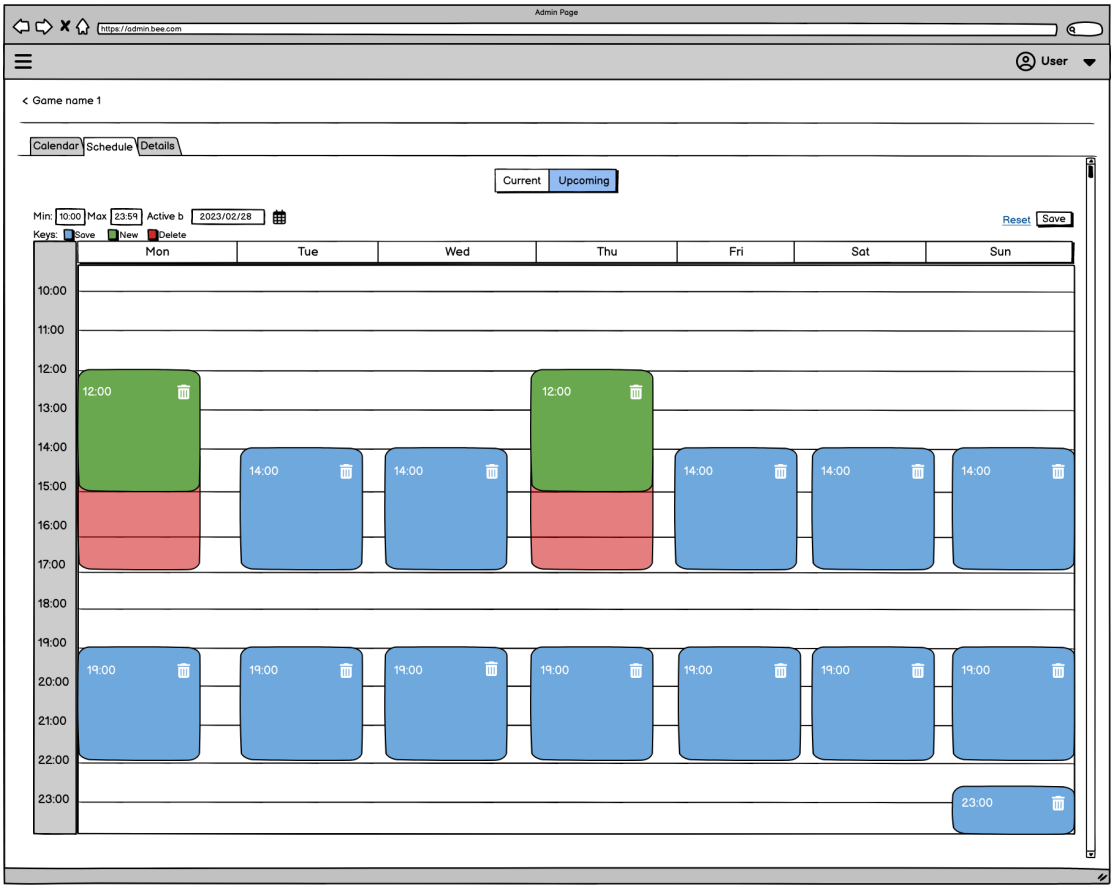
Στην παραπάνω εικόνα 7.7 που δημιουργήθηκε κατά τον αρχικό σχεδιασμό του ημερολογίου αποτυπώνεται η λειτουργικότητα και η οπτική του απεικόνιση στον διαχειριστή. Όπως μπορούμε επίσης να διακρίνουμε οι ρυθμίσεις, το πρόγραμμα και το ημερολόγιο κρατήσεων χωρίζονται με ένα διακριτικό σελιδοδείκτη στο πάνω μέρος της σελίδας κοντά στο όνομα του εκάστοτε δωματίου. Στην συνέχεια αναπτύξαμε και ένα επιπλέον πρόγραμμα λειτουργίας του κάθε δωματίου που ουσιαστικά θα ορίζεται το επερχόμενο πρόγραμμα από μία ορισμένη ημερομηνία και μετά. Αυτή η λειτουργία αναπτύχθηκε για να δώσει στους διαχειριστές την δυνατότητα να λύνουν αυτόνομα τα προβλήματα που προκύπτουν από μία ενδεχόμενη αλλαγή του προγράμματος λειτουργίας. Μέσω της ανάπτυξης κατάλληλων συναρτήσεων θα μπορεί ένας διαχειριστής να επεξεργάζεται το επερχόμενο πρόγραμμα λειτουργίας, να διαγράφει ή να δημιουργεί νέες διαθέσιμες ώρες προς κράτηση. Μία λειτουργία που αναπτύχθηκε για να κάνει πιο εύχρηστο το πρόγραμμα λειτουργίας είναι το Drag&Drop, είναι

μία λειτουργία που έχει εκπαιδευτεί ο μέσος χρήστης να χρησιμοποιεί, δίνει μεγαλύτερη ευκολία, απλοποίηση διαδικασιών και αποφυγή λαθών και δυσφορίας των χρηστών. Ακόμη μέσω της επιλογής κατάλληλων χρωματισμών τα οποία δίνονται και σαν κλειδιά στην αρχή του ημερολογιακού καμβά δίνεται καλύτερη εμπειρία στον χειριστή της εφαρμογής καθώς μπορεί να ξεχωρίζει απλά και εύκολα τι αντιπροσωπεύει κάθε χρωματισμός.



Εικόνα 7.7 Δημιουργία ημερολογίου του προγράμματος λειτουργίας ενός δωματίου

Τέλος ορίζεται από τον χρήστη και η ημερομηνία έναρξης ισχύος του επερχόμενου προγράμματος που θα αντικαταστήσει το τρέχων. Μέσω αυτής της ημερομηνίας βοηθάμε τα APIs μας να παρουσιάζουν στο frontend τα κατάλληλα ραντεβού που είναι διαθέσιμα προς το κοινό σύμφωνα με το ισχύον πρόγραμμα λειτουργίας. Επίσης έχει προβλεφθεί και ο διαχωρισμός των κρατήσεων που συμπίπτουν σε περίπτωση αλλαγής του προγράμματος λειτουργίας εντός μικρού χρονικού διαστήματος. Οι παλαίες κρατήσεις θα παίρνουν συγκεκριμένο χρώμα και θα εμφανίζεται κατάλληλη ειδοποίηση στον διαχειριστή για την επίλυση των επικαλύψεων μεταξύ των κρατήσεων παλαιού και νέου προγράμματος λειτουργίας.



Εικόνα 7.8 Επεξεργασία επερχόμενου προγράμματος λειτουργίας ενός δωματίου

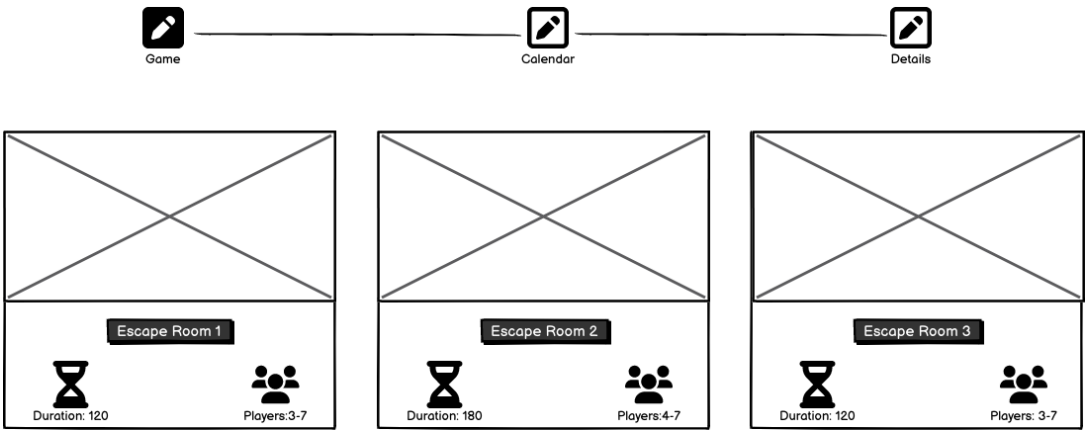
7.6 Δημιουργία Εφαρμογής Κρατήσεων

Η εφαρμογή που οι πελάτες της επιχείρησης θα μπορούν να πλοηγούνται και να πραγματοποιούν κρατήσεις θα είναι ένα widget το οποίο θα εγκατασταθεί στην ιστοσελίδα κάθε ενδιαφερόμενης επιχείρησης. Επιλέξαμε την ανάπτυξη ενός τέτοιου είδους εφαρμογής ώστε να μπορεί να προσαρμόζεται και εγκαθίσταται εύκολα και γρήγορα σε όλων των ειδών τις ιστοσελίδες των πελατών του συστήματος κρατήσεων μας, φυσικά δεν διαφέρει σε τίποτα με τα παραπάνω ώς πρός την υλοποίηση και τις τεχνολογίες.

7.6.1 Διαμόρφωση δομής και διάταξης στοιχείων

Κατα τον σχεδιασμό της εφαρμογής αποφασίσαμε να δημιουργήσουμε όσο το δυνατόν πιο απλό και εύχρηστο σύστημα για τους πελάτες ώστε να μην ακολουθούν πολύπλοκες διαδικασίες. Στην παρακάτω εικόνα 7.9 φαίνεται η σχεδίαση της εφαρμογής κρατήσεων για

πελάτες, η γενική δομή (layout) και τα στοιχεία που την αποτελούν. Πιο συγκεκριμένα η εφαρμογή μας θα χωρίζεται σε δύο βασικά μέρη, το πάνω κομμάτι θα μας δείχνει την τρέχουσα κατάσταση της κράτησης μας με στοιχεία για αυτή και το δεύτερο θα περιλαμβάνει το βασικό περιεχόμενο του κάθε βήματος, όπως είναι τα διαθέσιμα δωμάτια, το ημερολόγιο με τις διαθέσιμες ημέρες και ώρες, φόρμα για την συμπλήρωση των απαραίτητων στοιχείων του πελάτη και οι πληροφορίες της κράτησης.



Εικόνα 7.9 Σχεδιασμός εφαρμογής κρατήσεων για πελάτες

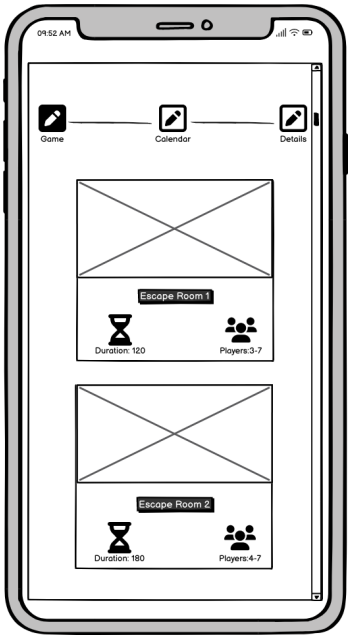
7.6.2 Διαχωρισμός του UI σε μέρη (components)

Η δημιουργία μιας κράτησης θα χωρίζεται σε τρία βασικά βήματα, τα οποία θα αποτυπώνονται στο πάνω μέρος της εφαρμογής μας και στο κάτω μέρος θα βρίσκεται το βασικό περιεχόμενο. Τα βήματα αυτά θα είναι:

- Επιλογή δωματίου
- Επιλογή ημερομηνίας και ώρας βάση διαθεσιμότητας
- Συμπλήρωση απαραίτητων πληροφοριών

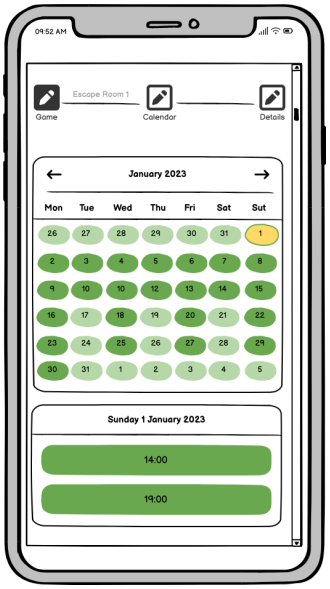
Στο τέλος θα ακολουθεί και η επιβεβαίωση της κράτησης από το σύστημα κρατήσεων.

Πριν αναπτύξουμε τα παραπάνω βήματα ως προς τα μέρη που δομήθηκε θα θέλαμε σε αυτό το σημείο να επισημάνουμε οτι οι πλειοψηφία των μέσων χρηστών κάνει χρήση του διαδικτύου και των εφαρμογών από την κινητή τους συσκευή. Πράγμα που μας ωθεί στην σχεδίαση και ανάπτυξη της εφαρμογής μας ώστε να ανταποκρίνεται στις κινητές συσκευές (responsive) για την βέλτιστη εμπειρία των πελατών.



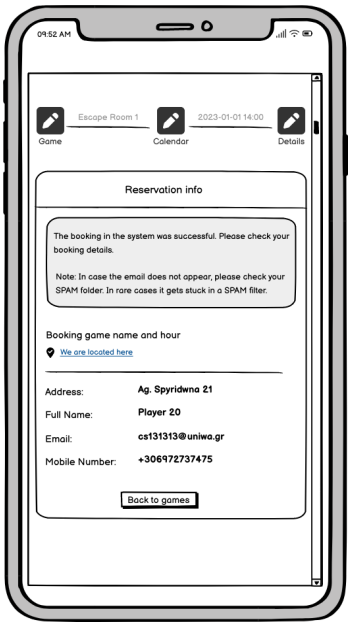
Εικόνα 7.10 Σχεδιασμός εφαρμογής κρατήσεων για κινητές συσκευές (responsive)

Στο πρώτο βήμα παρουσιάζονται τα διαθέσιμα δωμάτια κατόπιν επικοινωνίας με τον διακομιστή μέσω HTTP call για να λάβει το frontend τις αναγκαίες πληροφορίες. Εδώ μπορούμε να διακρίνουμε δύο βασικά components, το πρώτο είναι ο stepper ο οποίος διαχωρίζει τα βήματα της κράτησης και ενημερώνει αναλόγως το περιεχόμενο και το δεύτερο τμήμα αποτελείται από μία λίστα με τα διαθέσιμα παιχνίδια. Στο δεύτερο βήμα ενημερώνεται ο stepper με το δωμάτιο που επιλέγει και κατόπιν επικοινωνίας με το backend της εφαρμογής έρχονται σε μορφή JSON οι διαθέσιμες ημέρες και ώρες για το συγκεκριμένο δωμάτιο. Εδώ αναλαμβάνουμε μέσω κατάλληλων συναρτήσεων να οργανώσουμε και εν συνεχεία να παρουσιάσουμε στον χρήστη αυτή τη πληροφορία. Σε αυτό το σημείο αναπτύχθηκε ένα ημερολόγιο εξ΄ολοκλήρου από εμάς για να καλύπτει πλήρως τις ανάγκες για τις κινητές συσκευές. Μέσω των συναρτήσεων που μας παρέχονται από την javascript αναπτύξαμε κατάλληλες συναρτήσεις που θα υπολογίζουν τις ημερομηνίες, τις ώρες καθώς και την διάρκεια κάθε δωματίου ώστε να αποστέλλουμε πίσω την σωστή πληροφορία.



Εικόνα 7.11 Υλοποίηση του μηνιαίου ημερολογίου διαθεσιμότητας

Εικόνα 7.12 Συμπλήρωση φόρμας



Εικόνα 7.13 Επιβεβαίωση κράτησης

Στο τελευταίο βήμα η εφαρμογή μας ζητάει από τον πελάτη να συμπληρώσει κάποια προσωπικά του στοιχεία για να επικυρωθεί η κράτηση και να αποσταλούν τα απαραίτητα ενημερωτικά μηνύματα. Σε αυτό το σημείο χρησιμοποιήθηκε μία φόρμα για τη συλλογή και εν συνεχεία αποστολή των δεδομένων μέσω HTTP στο κατάλληλο API για την διεκπεραίωση του αιτήματος. Τέλος παρουσιάζεται ένα μήνυμα επιβεβαίωσης στην οθόνη και επαναλαμβάνονται τα στοιχεία της κράτησης που πραγματοποιήθηκε, καθώς και κάποιες

πληροφορίες σχετικά με την λήψη του επιβεβαιωτικού email που θα λάβει ο πελάτης και κάποια στοιχεία επικοινωνίας και τοποθεσίας της επιχείρησης.

7.6.3 Αναπτυξη πολυγλωσσικής λειτουργίας

Η εφαρμογή που αλληλεπιδρά με τους πελάτες των επιχειρήσεων πολλές φορές έρχεται αντιμέτωπη με χρήστες διαφορετικών εθνικοτήτων και γλωσσών πράγμα που σημαίνει ότι το σύστημα κρατήσεων οφείλει να προσαρμόζεται σε αυτήν την ιδιαιτερότητα ώστε να μπορεί να καλύψει και να εξυπηρετήσει και ομιλητές διαφορετικών γλωσσών. Φυσικά δεν μπορούν να καλυφθούν όλες οι γλώσσες αλλά οι βασικές όπως τα αγγλικά που είναι παγκόσμια γλώσσα είναι μια επιβεβλημένη ανάγκη. Για τον λόγο αυτό αναπτύξαμε ένα μηχανισμό στη React μέσω του οποίου μπορούμε να φορτώνουμε κάθε φορά την γλώσσα που επιλέγει ο χρήστης. Με την είσοδο του χρήστη στην εφαρμογή φορτώνεται η προκαθορισμένη γλώσσα (default) και μία λίστα με τις διαθέσιμες γλώσσες που υποστηρίζει το σύστημα μας. Αν ο χρήστης επιλέξει για παράδειγμα την αγγλική τότε μέσω μιας ανώτερης συνάρτηση (High Order Component) φορτώνεται το περιεχόμενο από το αγγλικό αρχείο παίρνοντας κάθε component που παρουσιάζεται εκείνη τη στιγμή στην οθόνη μέσα από αυτήν την συνάρτηση. Αυτή η λειτουργία είναι πολύ σημαντική ειδικά σε κλάδους που το πελατολόγιο τους δεν προέρχεται μόνο από την ίδια χώρα.

7.7 Σχεδίαση και Υλοποίηση HTTP Service

Ένα ζητούμενο στον προγραμματισμό είναι η δημιουργία των συνθηκών για επαναχρησιμοποίηση μηχανισμών και κώδικα. Η δημιουργία τέτοιων μηχανισμών μειώνει σε μεγάλο βαθμό τα πιθανά λάθη στον κώδικα, γλιτώνει αρκετό χρόνο και κόστος από την ομάδα που το αναπτύσει. Ένα σημείο που χρησιμοποιείται ξανά και ξανά στην ανάπτυξη του συστήματος μας είναι η επικοινωνία του front-end μέσω HTTP κλήσεων με το back-end. Αυτή η επικοινωνία είναι διαρκείς καθώς λαμβάνονται και αποστέλλονται πληροφορίες σχετικά με τους χρήστες, τις κρατήσεις, το πρόγραμμα λειτουργίας κλπ. Η επιλογή για επικοινωνία μέσω HTTP έιναι η μέθοδος fetch, μία απλή μέθοδος με εύρος επιλογών και δυνατότητα χειρισμού λαθών. Αναπτύξαμε επομένως ένα HTTP service το οποίο θα αναλαμβάνει όλη την διαδικασία της επικοινωνίας με τα APIs χωρίς να χρειάζεται από πλευράς front-end σε κάθε HTTP κλήση να ξαναγράφεται το ίδιο κομμάτι επικοινωνίας και να χειρίζονται ενδεχόμενα λάθη ή τα δεδομένα που λαμβάνονται.

Εικόνα 7.14 Χρήση του Http Service για την κλήση των διαθέσιμων δωματίων

Οι συναρτήσεις τύπου Hook εισήχθησαν στην React 16.8 ως εναλλακτικός τρόπος ανάπτυξης component αντί των κλασσικών components για να προσφέρουν σημαντικά πλεονεκτήματα ταχύτητας και επαναχρησιμοποίησης. Μερικά πλεονεκτήματα τους είναι η χρήση τους επαναλαμβανόμενες φορές με πολλαπλές τιμές κατάστασης ή αποτελέσματα, επίσης υπάρχουν ήδη έτοιμες πολλές συναρτήσεις Hook για εμάς όπως το useState(), useEffect(), useCallback() κλπ που κάνουν σημαντικές λειτουργίες για τον χειρισμό της

εσωτερικής κατάστασης κάθε component, αλλά μπορούμε να αναπτύξουμε και δικά μας custom Hooks για την κάλυψη πιο ειδικών απαιτήσεων. Το δικός μας Http Service αποτελεί ένα custom Hook, ώστε να επωφεληθούμε προγραμματιστικά από τα προτερήματα του καθώς μπορεί να χρησιμοποιηθεί πολλαπλές φορές καθώς κάθε κλήση της συνάρτησης αυτής αποτελεί αντιγραφή της αρχικής, κρατώντας φυσικά τη δική του κατάσταση τιμών και αποτελεσμάτων. Όπως βλέπουμε παραπάνω στο Http service που αναπτύξαμε έχουμε ορίσει όλες τις πιθανές Http μεθόδους και τα APIs που θα καλέσει το σύστημα μας, με αποτέλεσμα η κλήση του Http να αποτελεί μία πολύ απλή διαδικασία κλήσης του service με τις κατάλληλες συναρτήσεις. Το Http service μας ενημερώνει συνεχώς για την κατάσταση του αιτήματος μας και γι αυτό τα αποτελέσματα που μας επιστρέφει είναι τεσσάρων ειδών μεταβλητές, συν την συνάρτηση κλήσης (callback). Πιο συγκεκριμένα τα αποτελέσματα που μας επιστρέφονται είναι:

- **Ok**: Σήμα επιτυχούς κλήσης του Http
- **Data**: Τα αποτελέσματα από την κλήση εφόσον υπάρχουν
- **Error**: Πιθανά λάθη που μπορεί να προκύψουν
- **Loading**: Σήμα φόρτωσης καθ’ όλη τη διάρκεια της κλήσης
- **Request**: Κλήση της συνάρτησης Fetch

Παράδειγμα κλήσης του Http Service:

```
const { ok, data, error, loading, request } = service.games.useAll();
```

Όπως βλέπουμε καλούμε το Http Service με επέκταση το games που μας αφορά στη προκειμένη περίπτωση και από αυτό το custom Hook για να μας επιστρέψει όλα τα διαθέσιμα games, στο μπροστά κομμάτι αποθηκεύουμε τις επιστρεφόμενες τιμές και η κλήση γίνεται μέσω της callback συνάρτησης request σε κατάλληλο σημείο στη React, όπως για παράδειγμα στη συνάρτηση `useEffect()` που χειρίζεται θέματα δεδομένων και άλλων ενεργειών. Η ανάπτυξη της συνάρτησης έγινε με γνώμονα να καλυφθούν και οι ανάγκες για Server-Sent Event κλήσεις, ώστε να καλυφθεί και η ανάγκη για άμεση ενημέρωση του ημερολογίου σε περίπτωση κρατήσεων από κάποιον άλλον χρήστη του συστήματος (διαχειριστική σελίδα ή widget). Όπως μπορούμε να διακρίνουμε στην εικόνα 7.14 στο κάτω δεξιά μέρος που έχουμε τους τύπους των requests, τελευταίο βρίσκεται και το getSSE, που υλοποιεί τον agent για την κλήση του Server-Sent Event. Στον κώδικα 7.2 φαίνεται η υλοποίηση του agent SSE

Κεφάλαιο 8

Υποδομή της εφαρμογής

Η εφαρμογή θα διανέμεται σαν υπηρεσία στους ιδιοκτήτες των χώρων ψυχαγωγίας και η πρόσβαση σε αυτή θα γίνεται μέσω του Ιστού χωρίς ο ιδιοκτήτης να χρειάζεται να εγκαταστήσει το λογισμικό στα δικά του συστήματα. Ο ιδιοκτήτης θα αποκτά πρόσβαση στην εφαρμογή αφού εγγραφεί στην υπηρεσία. Οπότε η εγκατάσταση και η ρύθμιση της εφαρμογής για κάθε νέο ιδιοκτήτη εξαρτάται εξ ολοκλήρου από εμάς.

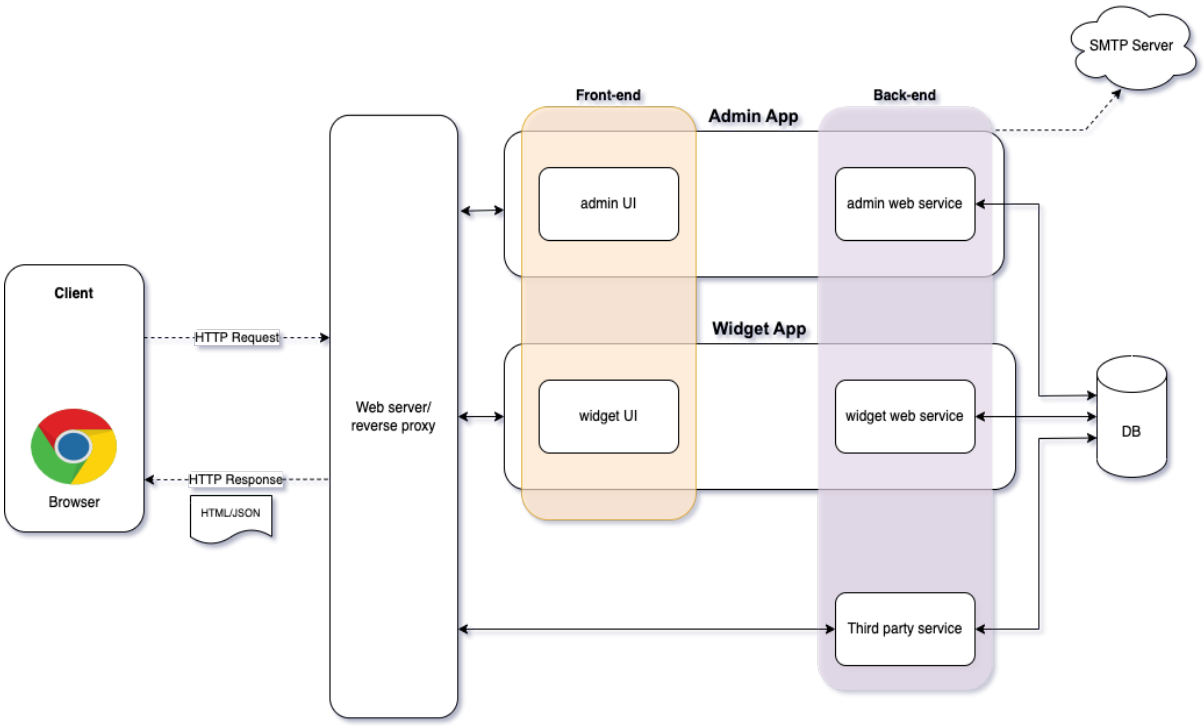
Για κάθε ιδιοκτήτη θα έχουμε ένα διαφορετικό στιγμιότυπο της εφαρμογής που θα παραμετροποιείτε ανάλογα με τις ανάγκες της επιχείρησης.

Η εφαρμογή μας είναι ένα σύστημα που αποτελείται από δύο web applications, τρία web services και μία βάση δεδομένων καθώς και έναν εξωτερικό SMTP server για την αποστολή emails. Παρακάτω αναλύουμε τα υποσυστήματα της εφαρμογής:

- **Admin App**: Η εφαρμογή αυτή αποτελείται από το ui-admin (front-end) και το api-admin (back-end). Το ui-admin είναι ένα ReactJS application το οποίο αποτελείται από στατικά JS αρχεία τα οποία θα σερβίρονται στον client μέσω ενός web server. Το api-admin είναι ένας web server, ο οποίος αποτελεί τον συνδετικό κρίκο μεταξύ του front-end και της βάσης δεδομένων. Το ui-admin εκτελείται στον περιηγητή και επικοινωνεί με το api-admin μέσω HTTP request.
- **Widget App**: Η εφαρμογή αυτή αποτελείται από το ui-widget (front-end) και το api-widget (back-end). Το ui-widget είναι ένα ReactJS application το οποίο αποτελείται από στατικά JS αρχεία τα οποία θα σερβίρονται στον client μέσω ενός web server. Το api-widget είναι ένας web server, ο οποίος αποτελεί τον συνδετικό κρίκο μεταξύ του front-end και της βάσης δεδομένων. Το ui-widget εκτελείται στον περιηγητή και επικοινωνεί με το api-widget μέσω HTTP request.
- **Public APIs**: Αποτελείται από το api-public το οποίο εκθέτει το web service για χρήση από τρίτους παρόχους.

Οι χρήστες θα αποκτούν πρόσβαση στην εφαρμογή μέσω ενός web server που θα λειτουργεί σαν reverse proxy, δηλαδή θα λαμβάνει όλα τα HTTP requests και θα τα αναθέτει στο

κατάλληλο διακομιστή με βάση ορισμένους κανόνες που θα έχουμε ορίσει. Η υποδομή της εφαρμογής φαίνεται συνοπτικά στην εικόνα 8.1.



Εικόνα 8.1 Υποδομή της εφαρμογής

8.1 Reverse proxy server

Ο proxy server είναι ένας ενδιάμεσος διακομιστής που προωθεί αιτήματα για περιεχόμενο από πολλούς πελάτες σε διαφορετικούς διακομιστές στο Διαδίκτυο. Ο reverse proxy server είναι ένας τύπος proxy server που συνήθως βρίσκεται πίσω από το τείχος προστασίας σε ένα ιδιωτικό δίκτυο και κατευθύνει αιτήματα πελατών στον κατάλληλο backend server. Ένας reverse proxy server παρέχει ένα πρόσθετο επίπεδο αφαίρεσης και ελέγχου για τη διασφάλιση της ομαλής ροής της κυκλοφορίας δικτύου μεταξύ πελατών και διακομιστών.

Οι συνήθεις χρήσεις για έναν reverse proxy server περιλαμβάνουν:

- **Load balancing**: Ένας reverse proxy server μπορεί να λειτουργήσει ως “τροχονόμος”, που κάθεται μπροστά από τους backend servers και διανέμει αιτήματα πελατών σε μια ομάδα διακομιστών με τρόπο που μεγιστοποιεί την ταχύτητα και τη

χρήση χωρητικότητας, διασφαλίζοντας ταυτόχρονα ότι κανένας διακομιστής δεν είναι υπερφορτωμένος ώστε να μην υποβαθμίσει την απόδοση. Εάν ένας διακομιστής διακοπεί, το πρόγραμμα load balancer ανακατευθύνει την κυκλοφορία στους υπόλοιπους web servers.

- **Web acceleration:** Οι Reverse proxies μπορούν να συμπιέζουν τα εισερχόμενα και εξερχόμενα δεδομένα, καθώς και την προσωρινή αποθήκευση περιεχομένου (caching) που συνήθως ζητείται, τα οποία επιταχύνουν τη ροή της κυκλοφορίας μεταξύ πελατών και διακομιστών. Μπορούν επίσης να εκτελούν πρόσθετες εργασίες, όπως κρυπτογράφηση SSL για να αφαιρέσουν το φορτίο των διακομιστών ιστού, ενισχύοντας έτσι την απόδοσή τους.
- **Security and anonymity**: Αναχαιτίζοντας αιτήματα που κατευθύνονται προς τους backend servers, ένας reverse proxy server προστατεύει τις ταυτότητες τους και λειτουργεί ως πρόσθετη άμυνα έναντι επιθέσεων ασφαλείας. Εξασφαλίζει επίσης ότι είναι δυνατή η πρόσβαση σε πολλούς διακομιστές από έναν ενιαίο record locator ή URL, ανεξάρτητα από τη δομή του τοπικού σας δικτύου.

Σαν reverse proxy server θα χρησιμοποιήσουμε το λογισμικό Nginx. Ο Nginx είναι ένας δωρεάν, ανοιχτού κώδικα, υψηλής απόδοσης HTTP server επίσης μπορεί να χρησιμοποιηθεί και ως reverse proxy, load balancer, mail proxy και HTTP cache. Ο Nginx είναι γνωστός για την υψηλή απόδοση, τη σταθερότητα, το πλούσιο σύνολο χαρακτηριστικών, την απλή διαμόρφωση και τη χαμηλή κατανάλωση πόρων. [37]

Ο Nginx σαν reverse proxy server θα λαμβάνει τα HTTP requests και θα τα χειρίζεται μέσω των κανόνων που έχουν οριστεί στο configuration file. Ο Nginx διαιρεί λογικά τα configurations που προορίζονται για την εξυπηρέτηση διαφορετικού περιεχομένου σε blocks, τα οποία ζουν σε μια ιεραρχική δομή. Κάθε φορά που υποβάλλεται ένα HTTP request, ο Nginx ξεκινά μια διαδικασία καθορισμού των configuration blocks που πρέπει να χρησιμοποιηθούν για τον χειρισμό του HTTP request.

Τα κύρια blocks που θα αναλύσουμε είναι το server block και το location. Ένα server block είναι ένα υποσύνολο του configuration του Nginx που ορίζει έναν εικονικό διακομιστή που χρησιμοποιείται για τη διαχείριση αιτημάτων ενός καθορισμένου τύπου. Οι διαχειριστές συχνά διαμορφώνουν πολλαπλά μπλοκ διακομιστών και αποφασίζουν πιο block θα χειρίζεται

ποια σύνδεση με βάση το ζητούμενο server name, port και διεύθυνση IP. Στο παρακάτω παράδειγμα ένα request στο www.example.org:80 θα το αναλάβει το πρώτο block, ενώ ένα request στο www.example.net:80 θα το αναλάβει το δεύτερο.

```
server {
    listen      80;
    server_name example.org www.example.org;
    ...
}

server {
    listen      80;
    server_name example.net www.example.net;
    ...
}
```

Ένα location block ζει μέσα σε ένα server block και χρησιμοποιείται για να ορίσει πώς ο Nginx θα πρέπει να χειρίζεται αιτήματα για διαφορετικούς πόρους και URIs. Το URI μπορεί να υποδιαιρεθεί με όποιον τρόπο επιθυμεί ο διαχειριστής χρησιμοποιώντας αυτά τα blocks.

Η εφαρμογή αποτελείται από τρεις διαφορετικές εφαρμογές ανάλογα με τον χρήστη. Η εφαρμογή διαχείρισης κρατήσεων, η εφαρμογή δημιουργίας κρατήσεων από τους πελάτες και η υπηρεσία διαχείρισης κρατήσεων για τρίτους παρόχους και κάθε εφαρμογή αποτελείται από το frontend που είναι ένα web app με γραφικό περιβάλλον που εκτελείται στον φυλλομετρητή και ένα web service που χρησιμοποιείται για την επικοινωνία του Web app με την βάση δεδομένων.

Στον configuration file του Nginx ορίζουμε τρεις servers, έναν για κάθε εφαρμογή ή υπηρεσία και θα διαχωρίζονται με βάση το domain name. Οι servers θα είναι ο admin.${HOST} που θα αφορά το web app του διαχειριστή, ο publi.${HOST} που θα αφορά το web service για integration με τρίτες υπηρεσίες και ο ${HOST} που θα αφορά το web app για την δημιουργία κρατήσεων από τους πελάτες. Το ${HOST} θα είναι παραμετρικό και θα χρησιμοποιείτε το domain name κάθε επιχείρησης.

```
server {
    listen       80;
```

```
    listen  [::]:80;
    server_name  admin.${HOST};

    ...
}

server {
    listen       80;
    listen  [::]:80;
    server_name  public.${HOST};

    ...
}

server {
    listen       80;
    listen  [::]:80;
    server_name ${HOST};

    ...
}
```

Μέσα στο σώμα του server block μπορούμε να ορίσουμε κρυπτογράφηση SSL, να προσθέσουμε ή να διαβάσουμε το header request και τέλος να ορίσουμε τα NGINX locations.

Στον κώδικα 8.1 βλέπουμε ένα παράδειγμα του server block για το web app του διαχειριστή. Σε αυτό έχουμε ορίσει τρία location blocks που λειτουργούν σαν reverse proxy server για τα αιτήματα προς αυτό. Το πρώτο location κατευθύνει όλα τα requests που ταιριάζουν στον server που έχει το ReactJS application. Το δεύτερο location κατευθύνει όλα τα requests που το URI ξεκινάνε με “/api” στον server του web service. Και το τρίτο location κατευθύνει όλα τα requests που το URI ξεκινάνε με “/api” και τελειώνουν σε “/sse” στον server του web service αλλά θέτει δύο επιπλέον headers για την υποστήριξη του SSE .

8.2 Deployment

Deployment είναι είναι η διαδικασία προώθηση αλλαγών ή ενημερώσεων από το ένα περιβάλλον αναπτυξης στο άλλο. Κατά τη δημιουργία μιας εφαρμογής Ιστού θα υπάρχει πάντα το live website, το οποίο ονομάζεται production environment. Για να υπάρχει η

δυνατότητα να γίνονται αλλαγές χωρίς αυτές να επηρεάζουν το production environment, μπορουν να υπάρχουν επιπλέον environments, όπως το dev environment ή το test environment.

Το deployment της εφαρμογής μας σε ένα περιβάλλον απαιτεί την εγκατάσταση και την ρύθμιση όλων τον υποσυστημάτων της εφαρμογής. Η εγκατάσταση αφορά τα εξής σημεία:

- Εγκατάσταση της βάσης δεδομένων
- Εγκατάσταση του api-admin και η ρύθμισή των παραμέτρων: Το web service που αφορά το admin web app
- Εγκατάσταση του api-widget και η ρύθμισή των παραμέτρων: Το web service που αφορά το web app των πελατών.
- Εγκατάσταση του api-public και η ρύθμισή των παραμέτρων: Το web service που αφορά τα συστήματα των τρίτων παρόχων.
- Εγκατάσταση ενός web server που θα σερβίρει τα στατικά αρχεία του React app ui-admin: Το admin web app
- Εγκατάσταση ενός web server που θα σερβίρει τα στατικά αρχεία του React app ui-widget: Το widget web app που μπορούν οι πελάτες να κανουν κρατήσεις.
- Εγκατάσταση του Nginx και ρύθμισή του configuration file για να κατευθυνει τα αιτήματα στα κατάλληλα συστήματα.

Τα υποσυστήματα μπορούν να βρίσκονται όλα στο ίδιο φυσικό server ή να μοιραστούν σε διαφορετικούς ανάλογα με της απαίτησης του κάθε υποσυστήματος.

8.3 Infrastructure as Code (IaC)

Το Infrastructure as Code (IaC) χρησιμοποιεί μια περιγραφική γλώσσα κωδικοποίησης υψηλού επιπέδου για την αυτοματοποίηση της υποδομής ενός πληροφοριακό συστήματος. Αυτός ο αυτοματισμός εξαλείφει την ανάγκη για τους προγραμματιστές να παρέχουν και να διαχειρίζονται μη αυτόματα διακομιστές, λειτουργικά συστήματα, συνδέσεις βάσεων δεδομένων, αποθήκευση και άλλα στοιχεία υποδομής κάθε φορά που θέλουν να αναπτύξουν, να δοκιμάσουν ή να αναπτύξουν μια εφαρμογή λογισμικού.

Σε μια εποχή που δεν είναι ασυνήθιστο για μια επιχείρηση να αναπτύσσει εκατοντάδες εφαρμογές στην παραγωγή κάθε μέρα - και όταν η υποδομή συνεχώς περιστρέφεται, καταρρέει και κλιμακώνεται πάνω-κάτω ως απάντηση στις απαιτήσεις προγραμματιστών και χρηστών - είναι απαραίτητο για έναν οργανισμό να αυτοματοποιήσει την υποδομή

προκειμένου να ελέγξει το κόστος, να μειώσει τους κινδύνους και να ανταποκριθεί με ταχύτητα σε νέες επιχειρηματικές ευκαιρίες και ανταγωνιστικές απειλές. Το IaC καθιστά δυνατό αυτόν τον αυτοματισμό.[38]

8.3.1 Docker Interface

Ένας τρόπος να οργανώσουμε την υποδομή του συστήματος με IaC είναι τα Docker containers. Το Docker είναι μια πλατφόρμα ανοιχτού κώδικα που επιτρέπει στους προγραμματιστές να δημιουργούν, να αναπτύσσουν, να εκτελούν, να ενημερώνουν και να διαχειρίζονται τυποποιημένα container, εκτελέσιμα στοιχεία που συνδυάζουν τον πηγαίο κώδικα εφαρμογής με τις βιβλιοθήκες του λειτουργικού συστήματος (OS) και τις εξαρτήσεις που απαιτούνται για την εκτέλεση αυτού του κώδικα σε οποιοδήποτε περιβάλλον. Παρακάτω παρουσιάζουμε τα βασικά στοιχεία του Docker που θα χρησιμοποιήσουμε.

DockerFile

Κάθε container Docker ξεκινά με ένα απλό αρχείο κειμένου που περιέχει οδηγίες για τον τρόπο δημιουργίας ενός Docker container image. Το DockerFile αυτοματοποιεί τη διαδικασία δημιουργίας του Docker image. Είναι ουσιαστικά μια λίστα από οδηγίες command-line interface (CLI) που θα εκτελέσει το Docker Engine για να συναρμολογήσει το image. Η λίστα των Docker commands είναι τεράστια, αλλά τυποποιημένη: Οι λειτουργίες Docker λειτουργούν το ίδιο ανεξάρτητα από το περιεχόμενο, την υποδομή ή άλλες μεταβλητές περιβάλλοντος.

Docker images

Τα Docker images περιέχουν εκτελέσιμο πηγαίο κώδικα εφαρμογής καθώς και όλα τα εργαλεία, τις βιβλιοθήκες και τις εξαρτήσεις που χρειάζεται ο κώδικας εφαρμογής για να εκτελεστεί ως container. Όταν εκτελείται το Docker image, δημιουργείται ένα container που είναι ένα instance του image.

Docker containers

Τα Docker containers είναι οι ζωντανές, εκτελούμενες εμφανίσεις των Docker images. Ενώ τα Docker images είναι αρχεία μόνο για ανάγνωση, τα containers είναι ζωντανά, εφήμερα, εκτελέσιμα περιεχόμενα. Οι χρήστες μπορούν να αλληλεπιδράσουν μαζί τους και οι

διαχειριστές μπορούν να προσαρμόσουν τις ρυθμίσεις και τις συνθήκες τους χρησιμοποιώντας Docker commands.

Docker registry

Το Docker registry είναι ένα επεκτάσιμο σύστημα αποθήκευσης και διανομής ανοιχτού κώδικα για Docker images. Το registry δίνει τη δυνατότητα να παρακολουθούνται οι εκδόσεις των images στα repositories, χρησιμοποιώντας την προσθήκη ετικετών για αναγνώριση. Αυτό επιτυγχάνεται χρησιμοποιώντας το git, ένα εργαλείο version control.

Docker Compose

Οι προγραμματιστές μπορούν να χρησιμοποιήσουν το Docker Compose για τη διαχείριση εφαρμογών πολλαπλών containers, όπου όλα τα containers εκτελούνται στον ίδιο κεντρικό υπολογιστή Docker. Το Docker Compose δημιουργεί ένα αρχείο YAML (.yml) που καθορίζει ποίες υπηρεσίες περιλαμβάνονται στην εφαρμογή και μπορεί να αναπτύξει και να εκτελέσει containers με μία μόνο εντολή. Επειδή η σύνταξη YAML είναι αγνωστική ως προς τη γλώσσα, τα αρχεία YAML μπορούν να χρησιμοποιηθούν σε προγράμματα γραμμένα σε Java, Python, Ruby και πολλές άλλες γλώσσες.

8.3.2 Deploy with docker

Χρησιμοποιώντας dockerfiles για κάθε σύστημα μπορούμε να κάνουμε build τον κώδικα και να δημιουργήσουμε ένα docker image που περιέχει μόνο το εκτελέσιμο κώδικα για κάθε σύστημα. Στον κώδικα 8.2 βλέπουμε το Dockerfile που δημιουργεί το image για το ui-admin. Το ui-admin είναι ένα web app γραμμένο σε ReactJS οπότε χρησιμοποιώντας το επίσημο image για node JS προσθέτουμε τον πηγαίο κώδικα της εφαρμογής και το κάνουμε build. Αφού γίνει build ο κώδικας στο node container, δημιουργούμε ένα image από το επίσημο nginx image στην έκδοση alpine. Το Alpine Linux είναι μια διανομή Linux που έχει σχεδιαστεί για να είναι μικρή, απλή και ασφαλής. Το τελικό image θα είναι ένα Alpine Linux με Nginx που θα περιέχει τον εκτελέσιμο κώδικα της react και ένα Nginx configuration που θα λειτουργεί σαν web server για να σερβίρει τα στατικά αρχεία του React app. Αντίστοιχα θα υπάρχει ένα Dockerfile για κάθε υποσύστημα.

Τα τελικά images θα είναι τα παρακάτω:

- registry.gitlab.com/khakibee/khakibee/nginx

- registry.gitlab.com/khakibee/khakibee/ui-admin
- registry.gitlab.com/khakibee/khakibee/api-admin
- registry.gitlab.com/khakibee/khakibee/ui-widget
- registry.gitlab.com/khakibee/khakibee/api-widget
- registry.gitlab.com/khakibee/khakibee/api-public
- registry.gitlab.com/khakibee/khakibee/psql

Όπου registry.gitlab.com/khakibee/khakibee είναι το git repository. Για το version control της εφαρμογής κάθε φορά που δημιουργείται νέα έκδοσή κατάλληλη για deployment θα δημιουργούμε τα Docker images με tag το version number ώστε να διατηρούμε τις διαφορετικές εκδοσεις. Για παράδειγμα στο image registry μπορούν να υπάρχουν τα images:

- registry.gitlab.com/khakibee/khakibee/nginx:1.0.0
- registry.gitlab.com/khakibee/khakibee/nginx:1.1.0
- registry.gitlab.com/khakibee/khakibee/nginx:1.2.3.

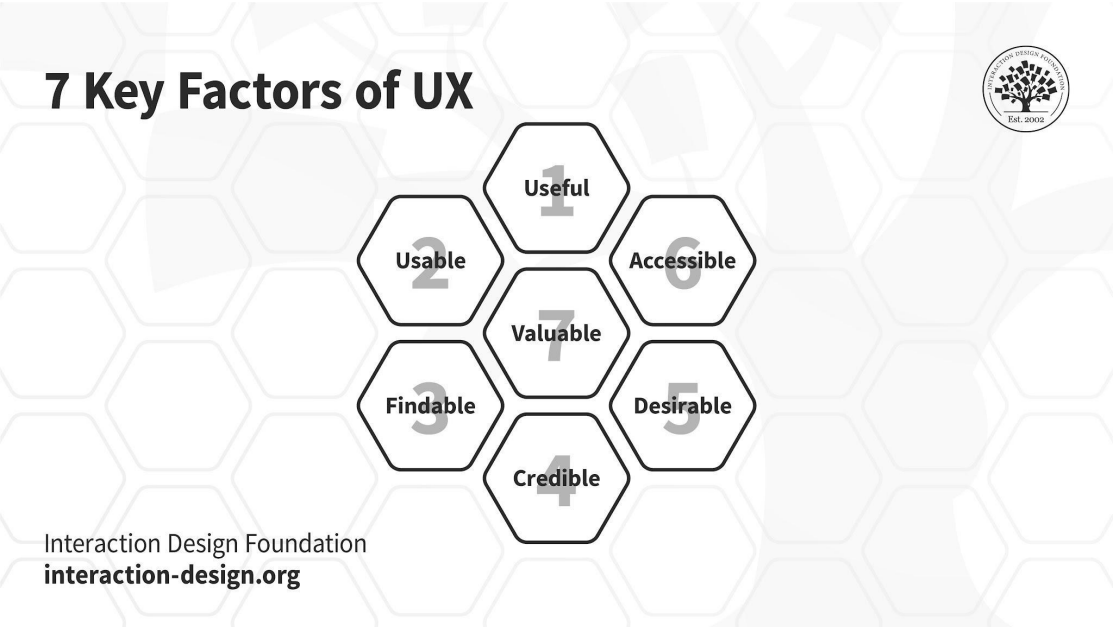
Χρησιμοποιώντας το Image registry του gitlab μπορούμε να δημιουργούμε containers έτοιμα για deployment. Με αυτόν τον τρόπο μπορούμε να έχουμε γρήγορα όποια έκδοσή θέλουμε καθώς και να ορίζουμε τους πόρους για κάθε container. Υπάρχουν διάφορες τεχνολογίες για να δημιουργηθεί η υποδομή μιας εφαρμογής μέσω images. Η πιο διαδεδομένη είναι το Kubernetes. Το Kubernetes είναι μια πλατφόρμα ενορχήστρωσης container για τον προγραμματισμό και την αυτοματοποίηση της ανάπτυξης, διαχείρισης και κλιμάκωσης εφαρμογών με container. Εμείς χρησιμοποιείσαμε το Docker compose ένας πιο απλός τρόπος για να κανουμε deploy την επιθυμητή έκδοσή της εφαρμογής γρήγορα. Στον κώδικα 8.4 φαίνεται ένα παράδειγμα του αρχείου docker-compose και στον κώδικα 8.3 ένα παράδειγμα του .env αρχείου που χρησιμοποιεί. Εκτελώντας το docker-compose μέσω των Docker commands το docker κατεβάζει τα images με το κατάλληλο tag και δημιουργεί και τρέχει όλα τα containers της εφαρμογής. Με αυτον τον τρόπο μπορούμε να τρέξουμε την εφαρμογή μας σε κάθε συστημα που υπάρχει το docker με μια μόνο εντολή. Επίσης μέσω των Docker commands μπορούμε να δούμε την κατάσταση του κάθε container, τα log files ή και να σταματήσουμε ή να επανεκκινήσουμε κάποιο container.

Κεφάλαιο 9ο

Α) Αξιολόγηση ευχρηστίας με χρήση Γνωστικού Περιδιαβάσματος

9.1α Βασικές αρχές της ευχρηστίας

Η αξιολόγηση ευχρηστίας με χρήση Γνωστικού Περιδιαβάσματος (Cognitive Walkthrough) είναι μια μέθοδος αξιολόγησης της ευχρηστίας ενός συστήματος ή μιας διεπαφής χρήστη. Κατά τη διαδικασία αυτής της αξιολόγησης, ο αξιολογητής προσπαθεί να εκτιμήσει το πώς ένας τυπικός χρήστης θα χρησιμοποιήσει το σύστημα ή τη διεπαφή, μέσω μιας σειράς ερωτήσεων και αναλύσεων. Η ιδέα είναι ότι αν τους δοθεί η δυνατότητα επιλογής, οι περισσότεροι χρήστες προτιμούν να κάνουν πράγματα για να μάθουν ένα προϊόν παρά να διαβάσουν ένα εγχειρίδιο ή να ακολουθήσουν ένα σύνολο οδηγιών.



Εικόνα 9.1 Παράγοντες της ευχρηστίας (UX)

(https://public-images.interaction-design.org/literature/articles/materials/3Aehi4beGA0JJ78rzO6WdqG8oDu7ZyGVGF8tInxh.jpg)

7 βασικοί παράγοντες που συμβάλλουν σε καλύτερο UX:

- Useful
- Usable
- Findable
- Credible
- Desirable
- Accessible
- Valuable

9.2α Αξιολόγηση συστήματος μέσω Γνωστικού Περιδιαβάσματος

Με την βοήθεια της μελέτης και χρήσης γνωστικού περιδιαβάσματος θα μπορέσουμε να ικανοποιήσουμε τους περισσότερους, αν όχι όλους αυτούς τους ποιοτικούς δείκτες καλής εμπειρίας χρήστη.

Οι βασικές αρχές της μεθόδου αυτής είναι οι εξής:

1. Ο αξιολογητής περιγράφει το σενάριο χρήσης που θα ακολουθήσει ο χρήστης για να πετύχει ένα συγκεκριμένο στόχο.
2. Στη συνέχεια, ο αξιολογητής εξετάζει τις διάφορες επιλογές που έχει ο χρήστης σε κάθε στάδιο του σεναρίου και προσπαθεί να προβλέψει τις σκέψεις και τις αντιδράσεις του χρήστη κατά τη χρήση του συστήματος.
3. Ο αξιολογητής εξετάζει τη δυνατότητα του συστήματος να ανταποκριθεί στις αντιδράσεις του χρήστη και να τον καθοδηγήσει στο επόμενο βήμα του σεναρίου.

Στο δικό μας σύστημα η γνωστική μελέτη περιδιαβάσματος ξεκινά με τον καθορισμό μιας εργασίας που αναμένεται να εκτελέσει ο χρήστης της διαχειριστικής σελίδας του συστήματος κρατήσεων. Ως εργασία επιλέξαμε την επεξεργασία του προγράμματος λειτουργίας για ένα δωμάτιο καθώς είναι μια βασική λειτουργία για τους χώρους ψυχαγωγίας και ένα από τα προβλήματα που λύνει το σύστημα μας.

Δεδομένου ότι η εμπειρία είναι υποκειμενική για κάθε χρήστη, δομήσαμε τον τρόπο με τον οποίο τεκμηριώσουμε τις απαντήσεις των χρηστών μας, έτσι ώστε όλοι οι έλεγχοι να

χρησιμοποιούν τα ίδια κριτήρια. Συμβουλευόμενη την θεωρία θέσαμε τέσσερις ερωτήσεις κατά τη διάρκεια της παραπάνω εργασίας.

1. Θα καταλάβει ο χρήστης πώς ξεκινάει η εργασία για να επιτύχει το σωστό αποτέλεσμα;
2. Θα παρατηρήσει ο χρήστης ότι η σωστή ενέργεια είναι στη διάθεση του;
3. Θα συνδέσει ο χρήστης τη σωστή ενέργεια με το αποτέλεσμα που περιμένει να επιτύχει;
4. Εάν εκτελεστεί η σωστή ενέργεια, θα δει ο χρήστης ότι σημειώνεται πρόοδος προς το επιδιωκόμενο αποτέλεσμα; [41]

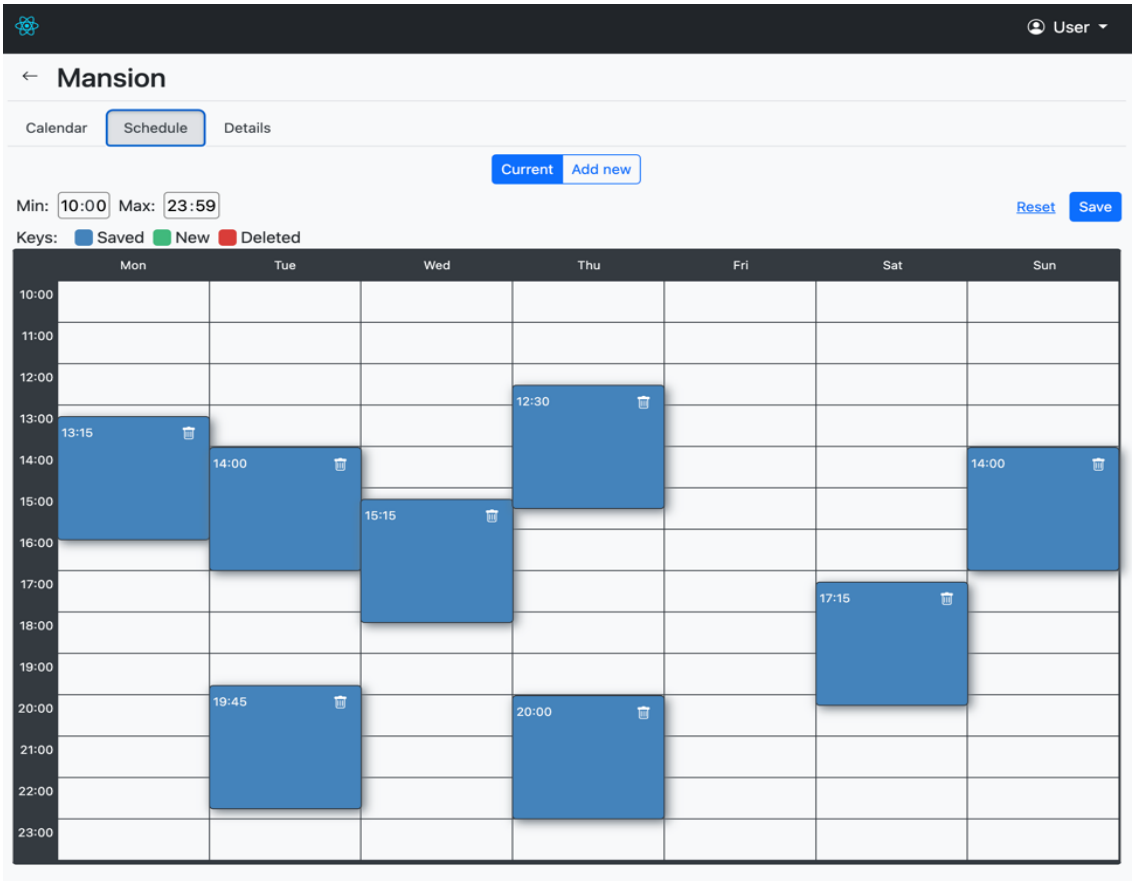
Μια καλή πρακτική είναι τα άτομα που θα απαντήσουν στις ερωτήσεις να μην είναι εξοικειωμένα με την ορολογία, τη γλώσσα και το σύστημα μας ώστε να μην χαθούν συμπεριφορές και πιθανή χρήση από κάποιον μη εξοικειωμένο όπου θα είναι και η πλειοψηφία των χρηστών. Παρακάτω παραθέτουμε τα αποτελέσματα της αναφοράς μας κατόπιν αξιολόγησης του σεναρίου για την επεξεργασία του προγράμματος λειτουργίας ενός δωματίου απόδρασης.

	Θα καταλάβει ο χρήστης πώς ξεκινάει η εργασία για να επιτύχει το σωστό αποτέλεσμα;	Θα παρατηρήσει ο χρήστης ότι η σωστή ενέργεια είναι στη διάθεση του;	Θα συνδέσει ο χρήστης τη σωστή ενέργεια με το αποτέλεσμα που περιμένει να επιτύχει;	Εάν εκτελεστεί η σωστή ενέργεια, θα δει ο χρήστης ότι σημειώνεται πρόοδος προς το επιδιωκόμενο αποτέλεσμα;
Επεξεργασία προγράμματος λειτουργίας				
Κλίκ στο ενδιαφερόμενο δωμάτιο	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Κλίκ στη καρτέλα “προγραμμα”	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Μετακίνηση ενός ή περισσοτέρων ραντεβού πρός όλες τις κατευθύνσεις (Drag and Drop)	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ
Επαναφορά ή	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ

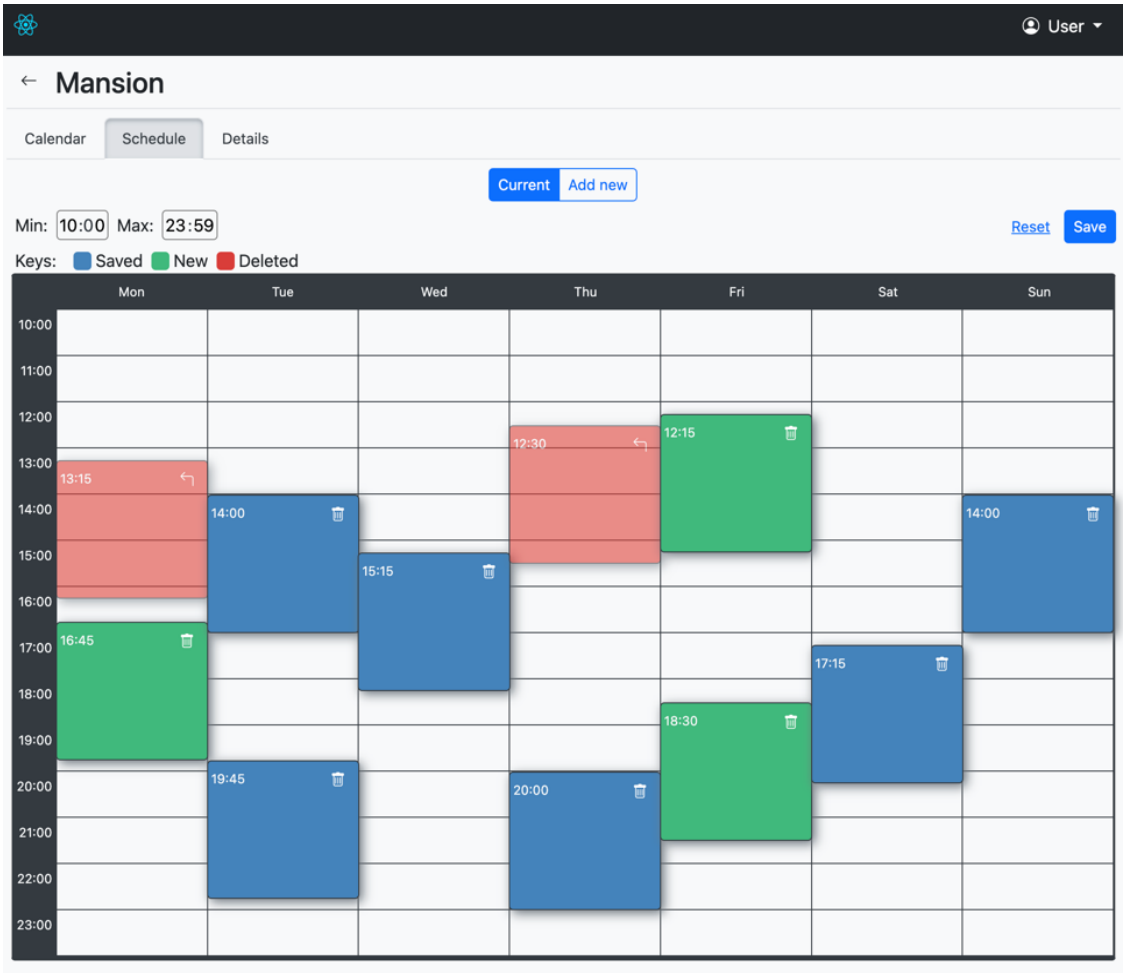
διαγραφή ραντεβού				
Αποθήκευση του επεξεργασμένου προγράμματος	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ

Πίνακας 9.1 Αναφορά σεναρίου επεξεργασίας προγράμματος λειτουργίας

Οι γνωστικοί έλεγχοι είναι φθηνοί και εύκολοι στη διεξαγωγή. Επιτρέπουν την εξέταση του συστήματος ως προς τη χρηστικότητα από την οπτική γωνία του χρήστη και παρέχουν γρήγορη ανατροφοδότηση για να καταστεί δυνατή η λήψη αποφάσεων κατά τη διαδικασία σχεδιασμού.



Εικόνα 9.2 Ημερολογιακός καμβάς προγράμματος λειτουργίας



Εικόνα 9.3 Επεξεργασμένος ημερολογιακός καμβάς προγράμματος λειτουργίας

Αξιολογώντας τα αποτελέσματα της αναφοράς μας συμπεραίνουμε ότι ένα από τα πιθανά σημεία που μπορεί να διορθωθεί ώστε το σύστημα μας να πληρεί τους βασικούς παράγοντες ενός εύχρηστου UX είναι το σημείο της “Μετακίνησης των υφιστάμενων ραντεβού πρός όλες τις κατευθύνσεις” μέσω της λειτουργίας Drag and Drop. Προκειμένου να βελτιώσουμε αυτό το σημείο θα μπορούσαμε να δώσουμε στον κέρσορα το κατάλληλο εικονίδιο ώστε να παραπέμπει στην λειτουργία του “πιάνω” ένα αντικείμενο και το “μετακινώ”.

- Όταν ο κέρσορας αιωρείται πάνω από ένα ωράριο:

```
div { cursor: move; }
```

- Όταν γίνει κρατημένο κλίκ πάνω στο επιθυμητό ωράριο:

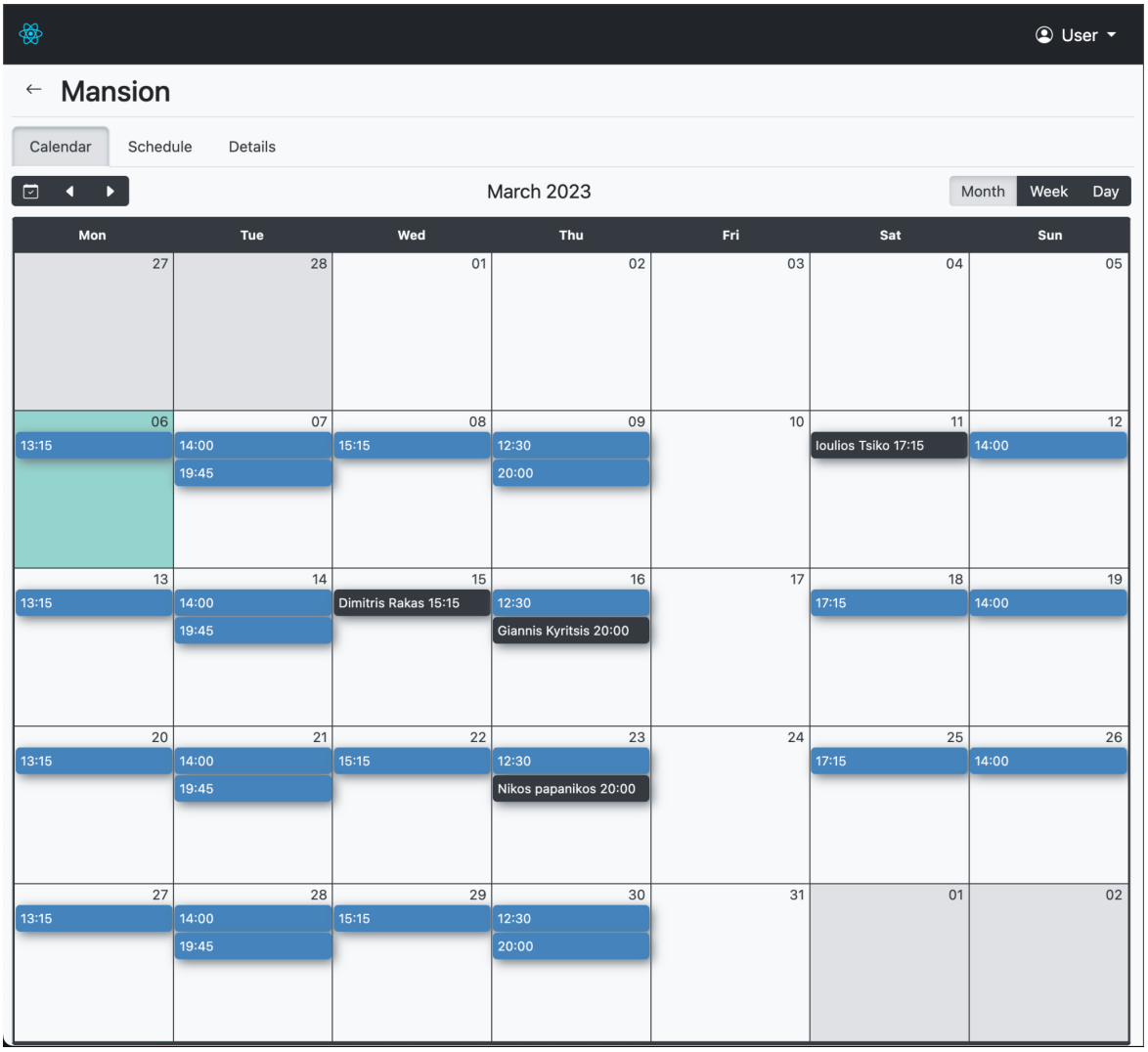
```
div { cursor: grapping; }
```

Σε συνέχεια της μελέτης του γνωστικού περιδιαβάσματος ένα άλλο σενάριο που θα αξιολογήσουμε είναι να εκτελέσει την πραγματοποίηση μιας νέας κράτησης ο χρήστης της διαχειριστικής σελίδας του συστήματος κρατήσεων.

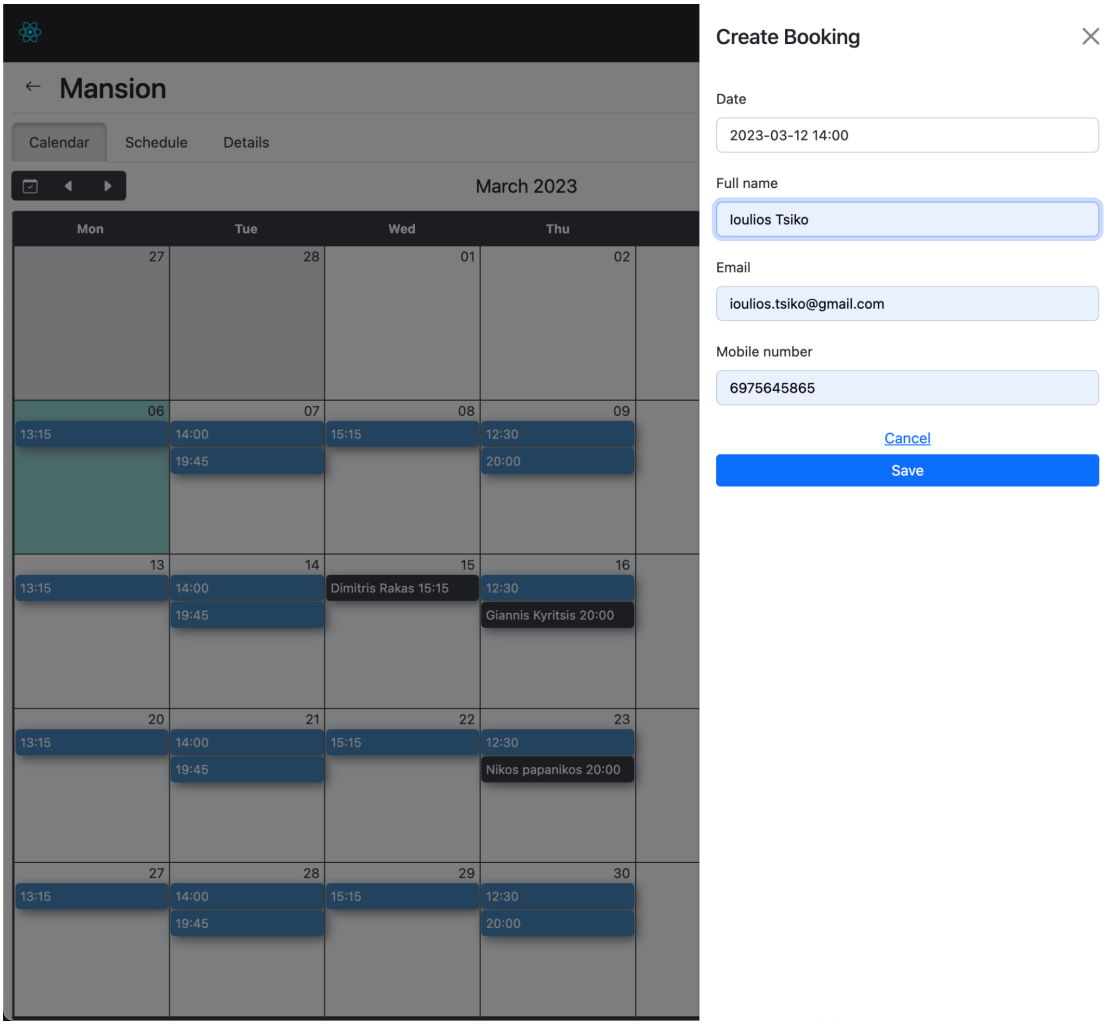
	Θα καταλάβει ο χρήστης πώς ξεκινάει η εργασία για να επιτύχει το σωστό αποτέλεσμα;	Θα παρατηρήσει ο χρήστης ότι η σωστή ενέργεια είναι στη διάθεση του;	Θα συνδέσει ο χρήστης τη σωστή ενέργεια με το αποτέλεσμα που περιμένει να επιτύχει;	Εάν εκτελεστεί η σωστή ενέργεια, θα δει ο χρήστης ότι σημειώνεται πρόοδος προς το επιδιωκόμενο αποτέλεσμα;
Πραγματοποίηση νέας κράτησης από την διαχειριστική σελίδα				
Κλίκ στο ενδιαφερόμενο δωμάτιο	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Κλίκ στη καρτέλα “Ημερολόγιο”	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Πλοήγηση στο ημερολόγιο (μήνας, εβδομάδα, ημέρα)	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Κλίκ στο ανοιχτό ενδιαφερόμενο διαθέσιμο ραντεβού (μπλέ χρώμα)	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Προσυμπληρωμένη ημερομηνία κράτησης βάση	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ

επιλογής				
Συμπλήρωση της φόρμας κράτησης με τα απαραίτητα στοιχεία	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ
Αποθήκευση της νέας κράτησης	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ

Πίνακας 9.2 Πραγματοποίηση νέας κράτησης από την διαχειριστική σελίδα



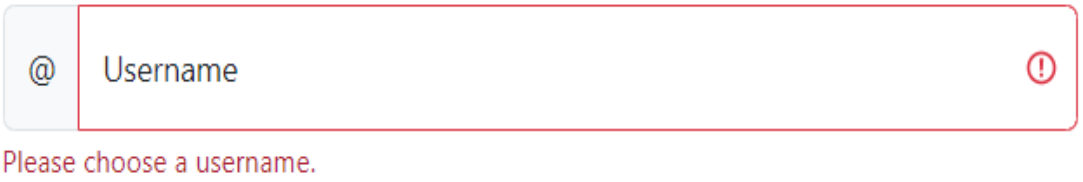
Εικόνα 9.4 Ημερολογιακός καμβάς προγράμματος λειτουργίας



Εικόνα 9.5 Ημερολογιακός καμβάς προγράμματος λειτουργίας

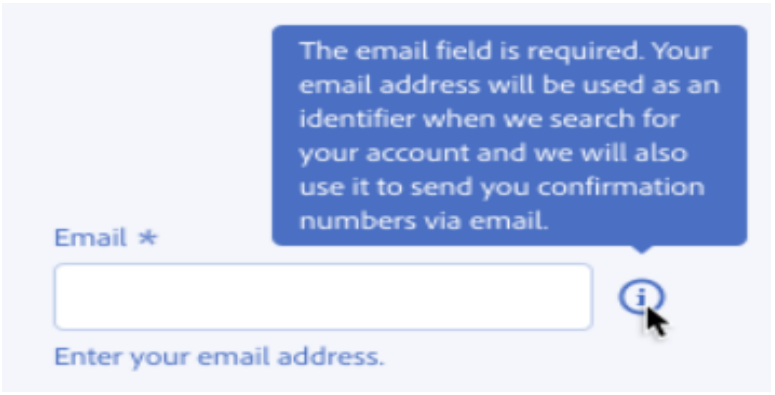
Αξιολογώντας τα αποτελέσματα της αναφοράς μας συμπεραίνουμε ότι τα σημεία που μπορούν να βελτιωθούν ώστε το σύστημα μας να πληρεί τους βασικούς παράγοντες ενός εύχρηστου UX είναι το σημείο της “Πραγματοποίηση κράτησης από τη διαχειριστική σελίδα” μέσω του ημερολογιακού καμβά. Προκειμένου να βελτιώσουμε αυτό το σημείο θα μπορούσαμε να εισάγουμε την προβολή ενημερωτικού μηνύματος με μορφή tooltip που να δίνει σύντομη περιγραφή των οδηγιών ή περιορισμών με κείμενο. Εν συνεχεία στις παραπάνω διορθώσεις είναι η εμφάνιση μηνύματος σφάλματος όσον αφορά τα πεδία με λάθος δεδομένα και ενδεχομένως κατάλληλος χρωματισμός τους. Τέλος είναι η εμφάνιση προσυμπληρωμένων πεδίων με γκρι χρωματισμό και απενεργοποιημένο δυνατότητα επεξεργασίας.

- Ένδειξη σφάλματος με κατάλληλο χρωματισμό και μήνυμα λάθους:



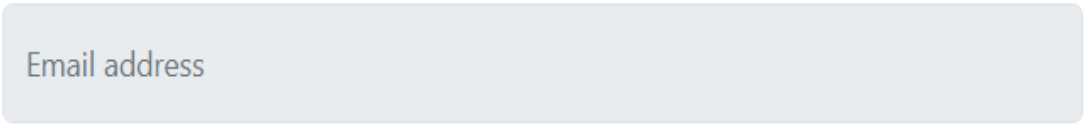
Εικόνα 9.6 Πεδίο φόρμας με υπόδειξη λάθους

- Εμφάνιση ενημερωτικού μηνύματος με κατάλληλες οδηγίες για συμπλήρωση του πεδίου:



Εικόνα 9.7 Οδηγίες συμπλήρωσης πεδίου φόρμας μέσω tooltip

- Εμφάνιση μη επεξεργάσιμου πεδίου ώς απενεργοποιημένο:



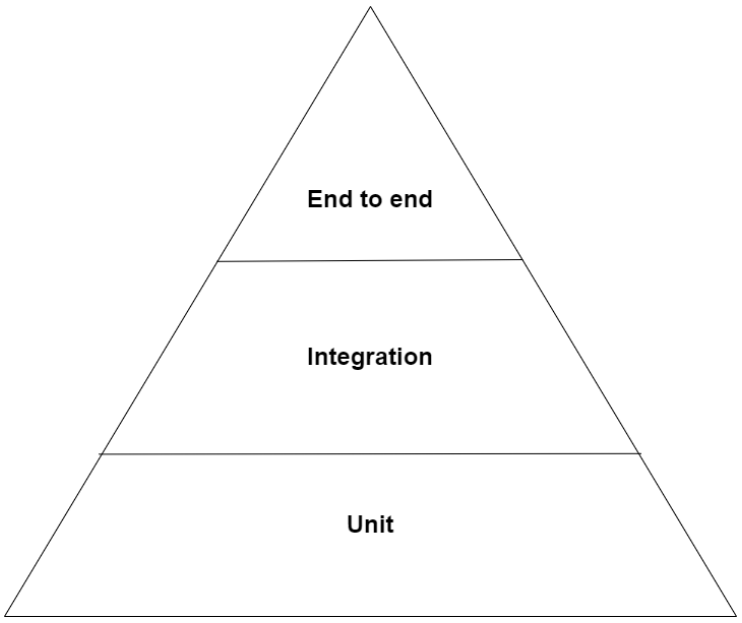
Εικόνα 9.8 Απενεργοποιημένο πεδίο φόρμας

Β) Δοκιμές της εφαρμογής

Οι δοκιμές (testing) μιας εφαρμογής αποτελούν ένα σημαντικό κομμάτι κατά την ανάπτυξη κώδικα και κατ' επέκταση μιας εφαρμογής καθώς είναι το μέσο που μας εξασφαλίζει την εγκυρότητα και την εμπιστοσύνη των λειτουργιών της εφαρμογής μας. Αν το σκεφτούμε καλύτερα οι δοκιμές αποτελούν κομμάτι της καθημερινότητας μας σε πολλές δραστηριότητες και εργασίες που κάνουμε, όπως για παράδειγμα η αλλαγή μιας λάμπας στο πορτατίφ πάντα συνοδεύεται και από δοκιμή για να διαπιστώσουμε αν δουλεύει και πετύχαμε τον αρχικό μας στόχο που ήταν να έχουμε φώς. Με το ίδιο σκεπτικό και κατά την ανάπτυξη του κώδικα κάνουμε τις απαραίτητες δοκιμές για να διαπιστωθεί ότι επετεύχθει ο αντικειμενικός σκοπός μιας συνάρτησης, μιας σελίδας ή ακόμα και ολόκληρης της εφαρμογής. Υπάρχουν διαφόρων ειδών δοκιμές ξεκινώντας από το βασικό που είναι οι χειροκίνητες δοκιμές και φτάνουμε στις αυτόματες όπου γράφουμε κώδικα για να ελέγχουμε την υλοποίηση της λειτουργικότητας της εφαρμογής μας. Οι λόγοι που στον κόσμο του προγραμματισμού επιλέγεται η δαπάνη χρόνου και χρήματος για την ανάπτυξη αυτόματων δοκιμών είναι κυρίως η εμπιστοσύνη που μπορεί να προσδώσει σε επερχόμενες αλλαγές στον κώδικα σε περίπτωση ανακατασκευής ή προσθήκης νέων χαρακτηριστικών, τη δημιουργία άτυπης τεκμηρίωσης (documentation) συνεχώς ενημερωμένης με τις τελευταίες αλλαγές και τέλος την αποφυγή επαναφοράς των ίδιων λαθών (bugs) στο κώδικα.

Παρακάτω βλέπουμε την πυραμίδα testing για το frontend κομμάτι, που σύμφωνα με τον Mike Cohn αποτελεί ίσως την πιο συχνή προσέγγιση για τον έλεγχο του λογισμικού. Στην κορυφή της πυραμίδας βρίσκονται τα End to end (E2E) tests, είναι τα πιο ακριβά για να αναπτυχθούν καθώς χρειάζονται να τρέξουν σε πραγματικό περιηγητή και ενδεχομένως με πραγματική βάση δεδομένων. Η βάση της πυραμίδας τα unit tests, είναι αυτά που βρίσκουμε σε μεγαλύτερες ποσότητες καθώς είναι τα πιο μικρά και οικονομικά για ανάπτυξη και ελέγχουν μεμονωμένα κομμάτια του κώδικα. Στην μέση βρίσκονται τα integration tests τα οποία συνδυάζουν ουσιαστικά τα unit tests μεταξύ τους σε μεγαλύτερη κλίμακα για να καλύψουν μία ολόκληρη ενότητα της εφαρμογής αλλά χωρίς καθόλου E2E. Η κάλυψη μιας εφαρμογής όσων αφορά τις δοκιμές(testing) δεν έχει νόημα πέρα από θεωρητικό επίπεδο να φτάσει στο 100% από πλευρά κόστους, χρόνου και αξιοπιστίας. Κάποια χαρακτηριστικά της εφαρμογής είναι δύσκολο να ελεγχθούν πλήρως, οι περιπτώσεις που αξίζει να καλυφθούν πλήρως είναι βιβλιοθήκες που προσφέρουν κρίσιμα στοιχεία και APIs καθώς και εφαρμογές

ανοιχτού κώδικα στις οποίες συμβάλλουν πολλοί προγραμματιστές και ενδεχομένως να μην είναι αρκετά εξοικειωμένοι με όλο το φάσμα της εφαρμογής.



Εικόνα 9.9 Πυραμίδα ελέγχου λογισμικού

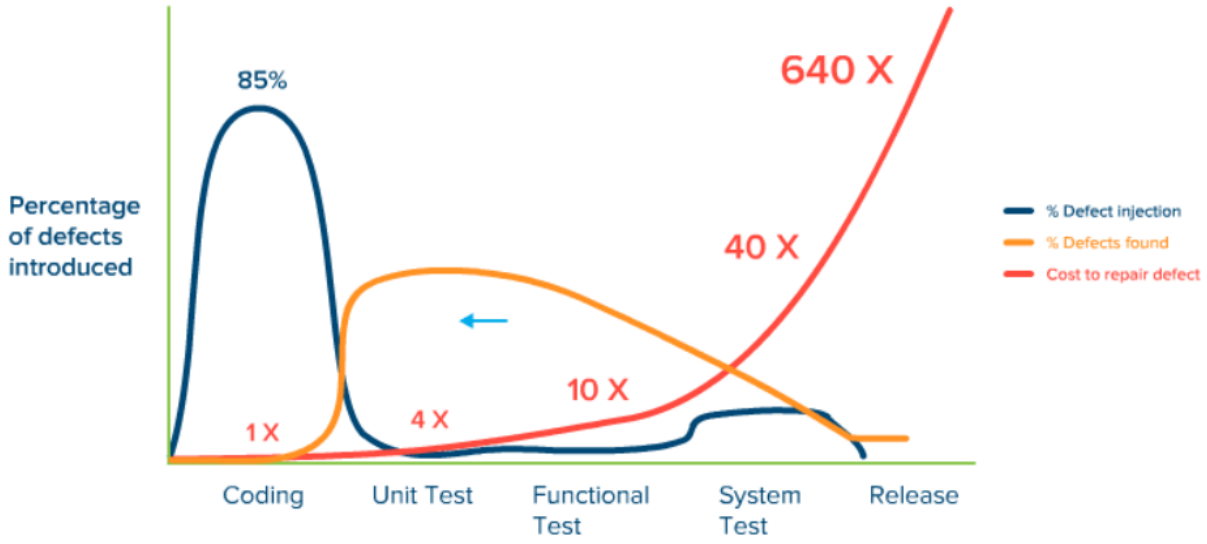
Είναι σημαντικό να αναπτύσσεται ο κώδικας ελέγχου αν όχι πριν τον κώδικα ανάπτυξης τουλαχιστον σε παράλληλο βαθμό ώστε να μειώσει μετέπειτα τα πιθανά λάθη που θα ανακύψουν και να προλάβει από σφάλματα που θα κοστίσουν πολύ περισσότερο από την ανάπτυξη κώδικα ελέγχου. Παρακάτω βλέπουμε σχηματικά ένα γράφημα για το πώς μπορεί να επηρεάσουν οι δοκιμές την συνολική εφαρμογή. Για τη βελτίωση της απόδοσης επένδυσης (ROI), θα πρέπει να χρησιμοποιηθεί η προσέγγιση "μετατόπιση προς τα αριστερά". Αυτή η προσέγγιση στοχεύει στη μετατόπιση των δοκιμαστικών λειτουργιών «προς τα αριστερά» που θα πραγματοποιηθούν όσο το δυνατόν νωρίτερα στον κύκλο ανάπτυξης. Πρέπει να αυξήσουμε την προσπάθεια για τον εντοπισμό σφαλμάτων στη φάση της δοκιμής, γεγονός που μας επιτρέπει να περιορίσουμε το κόστος που δημιουργείται στον υπόλοιπο κύκλο ανάπτυξης.

Κάποια βασικά χαρακτηριστικά που πρέπει να εισάγουμε στο κώδικα ελέγχου είναι:

- Τα tests πρέπει να είναι ντετερμινιστικά, να μην εξαρτώνται από το περιβάλλον.
- Θα πρέπει να αποφεύγονται σενάρια tests, χωρίς κάποια αξία στο κώδικα.
- Δεν χρειάζεται η εμμονική κάλυψη του κώδικα στο 100%.
- Να αναπτύσσεται όσο το δυνατόν περισσότερα tests ενσωμάτωσης(integration).

Εξειδικεύοντας τις κατηγορίες του frontend testing που έχουμε καταγράψει τις εξής:

- Manual tests
- End to end tests
- Integration tests
- Unit tests
- Static analysis



Εικόνα 9.10 Ανάλυση παραγωγικότητας, ποιότητας κώδικα και δοκιμών

(https://webbylab.com/blog/guide-of-testing-react-components-with-hooks-mocks/?fbclid=IwAR04ecwF8r000b9KXUzFklbhwYbsO_jwg71k6esu4zNmT5gd0LEduAVjPAQ)

9.1β Διαμόρφωση ρυθμίσεων Linter (Static Analysis)

Μια κατηγορία ελέγχου που μπορούμε να εντάξουμε κατα την ανάπτυξη κώδικα είναι πρόσθετα plugins που προσαρμόζουν τον κώδικα σε συγκεκριμένα φορματ βάση κανόνων ή που ελέγχουν τους τύπους μεταβλητών και την ροή του κώδικα. Φυσικά τέτοια εργαλεία δεν είναι ο κύριος τρόπος να για πραγματοποιούμε ελέγχους στο κώδικα μας αλλά αποτελούν έναν πολύ καλό οδηγό στον τρόπο συγγραφής καλού κώδικα. Αυτά τα εργαλεία όπως για παράδειγμα είναι το Prettier (Code formatter) και το ESLint (Linter) μπορούν να προστεθούν σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης κώδικα (IDE) και να προσαρμοστούν σύμφωνα με τις ανάγκες της εφαρμογής και της επιλογής ανάπτυξης κώδικα. Μέσω αυτών των πρόσθετων εργαλείων εξασφαλίζεται η συμβατότητα σε όλο το εύρος της εφαρμογής μας και

μας αποτρέπουν από την δημιουργία λαθών (συντακτικά) ή να εμφανίζουν κατάλληλα μηνύματα.

```
"rules": {
  "react/jsx-uses-react": "off", · · · 👍 [Best Practices]: Prevent React to be marked as unused
  "react/react-in-jsx-scope": "off", · · · 👍 [Possible Errors]: Prevent missing React when using JSX
  "consistent-return": "off", · · · Require `return` statements to either always or never specify values
  "react/jsx-one-expression-per-line": "off", · · · 🔧 [Stylistic Issues]: Limit to one expression per line in JSX
  "react/jsx-filename-extension": "off", · · · [Stylistic Issues]: Restrict file extensions that may contain JSX
  "jsx-a11y/label-has-associated-control": "off", · · · Enforce that a `label` tag has a text label and an associated control.
  "object-curly-newline": "off", · · · 🔧 Enforce consistent line breaks after opening and before closing braces
  "react/prop-types": "off", · · · 👍 [Best Practices]: Prevent missing props validation in a React component definition
  "max-len": [ · · · Enforce a maximum line length
    "off",
    {
      "code": 150
    }
  ],
  "import/order": [ · · · 🔧 eslint rule: import/order
    "error",
    {
      "groups": [
        "builtin",
        "external",
        "internal"
      ],
      "pathGroups": [
        {
          "pattern": "react",
          "group": "external",
          "position": "before"
        }
      ],
      "pathGroupsExcludedImportTypes": [
        "react"
      ],
      "newlines-between": "always",
      "alphabetize": {
        "order": "asc",
        "caseInsensitive": true
      }
    }
  ]
}
```

Εικόνα 9.11 Εφαρμογή κανόνων ανάπτυξης κώδικα σε Linter

Όπως βλέπουμε και στην παραπάνω εικόνα κατά την ανάπτυξη της εφαρμογής μας επιλέξαμε να ορίσουμε κάποιους κανόνες σύμφωνα με τις βέλτιστες πρακτικές ώστε να μας βοηθήσουν και να είναι ο οδηγός μας στην ανάπτυξη του συστήματος κρατήσεων. Για παράδειγμα ορίζουμε το πως θα χωρίζονται τα πακέτα (εξωτερικά, εσωτερικά, κλπ) που φορτώνονται στη React, με τι σειρά θα οργανώνονται, πόσους χαρακτήρες θα επιτρέπουμε ανά γραμμή για λόγους ευαναγνωσίας, κλπ. Αυτές οι ρυθμίσεις έδωσαν στον κώδικα μας ένα συγκεκριμένο μοτίβο, συνέπεια και σύμφωνα με τους περιορισμούς και συνθήκες που ορίσαμε αποφύγαμε σφάλματα που θα προέκυπταν από την ανεξέλεγκτη ανάπτυξη του.

9.2β Εγκατάσταση του Jest και React testing-library

Προκειμένου να ελέγχουμε τον κώδικα που αναπτύσσουμε για το σύστημα κρατήσεων μας επιλέξαμε να χρησιμοποιήσουμε δύο εργαλεία που μας προσφέρουν απλότητα και σύγχρονα χαρακτηριστικά ελέγχου καθώς και και να απευθύνονται στη Javascript και πιο συγκεκριμένα τη React. Το πρώτο μας εργαλείο είναι το Jest, είναι ένα πλαίσιο δοκιμών Javascript που φυσικά καλύπτει και τη βιβλιοθήκη της React. Μας επιτρέπει να γράφουμε και να εκτελούμε τα tests του κώδικα μας εύκολα και απλά. Το Jest μας παρέχει πολλά ισχυρά χαρακτηριστικά, όπως αυτόματη ανακάλυψη δοκιμών, mocking - spies και δοκιμές στιγμιοτύπων (snapshots). Το δεύτερο εργαλείο μας είναι το testing-library όπου είναι μία βιβλιοθήκη βασισμένη πάνω στο Jest και έχει σχεδιαστεί ειδικά για τον έλεγχο κώδικα React. Παρέχει πληθώρα βοηθητικών συναρτήσεων που διευκολύνουν σημαντικά τη δοκιμή της συμπεριφοράς των components της εφαρμογής μας, με τρόπο που να προσομοιάζει την χρήση και αλληλεπίδραση ενός χρήστη με την εφαρμογή. Αυτό είναι και το βασικότερο πλεονέκτημα της βιβλιοθήκης συγκριτικά με τον μεγάλο αντίπαλο του το Enzyme. Η χρήση των δύο παραπάνω εργαλείων συνδυαστικά θα μας δώσει την δυνατότητα να ελέγξουμε την εφαρμογή μας με εύκολο τρόπο και ισχυρά εργαλεία ώστε να δοκιμάσουμε σενάρια στα components της react, διασφαλίζοντας έτσι την σωστή λειτουργία και συμπεριφορά της εφαρμογής μας.

Προκειμένου να εγκαταστήσουμε τα εργαλεία μας θα μεταβούμε στην γραμμή εντολών/τερματικό του λειτουργικού μας ή του επεξεργαστής κώδικα μας (VS Code) και θα εγκαταστήσουμε αρχικά όλες τις εξαρτήσεις.

```
npm install --save-dev jest jest-environment-jsdom whatwg-fetch msw
@testing-library/react @testing-library/jest-dom
@testing-library/user-event
```

- Jest-dom: κάνει τους ελέγχους πιο ευανάγνωστους με την προσθήκη χρήσιμων ισχυρισμών.
- @testing-library/user-event: κάνει τους ελέγχους αλληλεπίδρασης του χρήστη να μοιάζουν πάρα πολύ με αυτές ενός πραγματικού χρήστη.
- Msw: είναι ένας πολύ καλός τρόπος για τη δοκιμή συστατικών που κάνουν αιτήσεις δικτύου.

Τέλος θα χρειαστούμε το babel-jest για το Babel και θα ανανεώσουμε το package.json αρχείο μας με τα παρακάτω στοιχεία.

```
"scripts": {
    "test": "jest",
    "test:watch": "jest --watch",
    "test:coverage": "jest --coverage"
  },
  "jest": {
    "setupFiles": ["whatwg-fetch"],
    "setupFilesAfterEnv":
["@testing-library/jest-dom/extend-expect"],
    "testEnvironment": "jest-environment-jsdom"
  }
```

9.3β Δημιουργία React Unit tests

Η ανάπτυξη unit tests είναι από τους πιο συνηθισμένους τρόπους για τον έλεγχο μιας συνάρτησης ή ενός μικρού component που κάνει μία μεμονωμένη εργασία. Τα unit tests απευθύνονται στον έλεγχο κομματιών του κώδικα που έχουν συνήθως μία συγκεκριμένη με μικρή σε έκταση αρμοδιότητα, ελέγχουν με λίγα λόγια τα επιμέρους σημεία μιας μεγαλύτερης οντότητας ώστε να προσέξουν λεπτομέρειες του κώδικα που δεν μπορούν να καλυφθούν με μεγαλύτερης κλίμακας δοκιμές. Σε αυτόν τον τύπο tests, δοκιμάζονται μεμονωμένες μονάδες ή συστατικά του λογισμικού, όπως για παράδειγμα μια μεμονωμένη συνάρτηση, μέθοδος, διαδικασία, ενότητα ή αντικείμενο. Ένα unit test απομονώνει ένα τμήμα κώδικα και επαληθεύει την ορθότητά του, προκειμένου να επικυρωθεί ότι κάθε μονάδα του κώδικα του λογισμικού λειτουργεί όπως αναμένεται. Για παράδειγμα, ο έλεγχος μιας συνάρτησης ή του κατά πόσον μια εντολή ή ένας βρόχος σε ένα πρόγραμμα λειτουργεί σωστά και ο στόχος της δοκιμής καθορίζει αν η μονάδα παράγει τα αναμενόμενα αποτελέσματα για μια δεδομένη είσοδο. Ένα παράδειγμα unit test που μπορούμε να αναλύσουμε και αναπτύχθηκε για το σύστημα κρατήσεων μας είναι ο έλεγχος μη αποθήκευσης λανθασμένης φόρμας. Προκειμένου να αναπτύξουμε τον έλεγχο χρησιμοποιήσαμε μια σειρά μεθόδων που παρέχει η Jest για την περιγραφή της δομής των δοκιμών. Όπως μπορούμε να διακρίνουμε στον κώδικα 9.1, η δοκιμή είναι αρκετά μικρή σε μέγεθος καθώς ελέγχει όπως προαναφέραμε μικρά μεμονωμένα κομμάτια του κώδικα. Για να το ορίσουμε καλούμε έτοιμες βοηθητικές συναρτήσεις του Jest και του Testing-library, και δίνουμε μία σύντομη περιγραφή για το ποιο είναι το αντικείμενο του ελέγχου ή τι περιμένουμε σαν αποτέλεσμα από αυτή τη δοκιμή. Όπως βλέπουμε το Testing-library είναι πολύ απλοϊκό και περιγραφικό και βοηθά πολύ στην συγγραφή δοκιμών πολύ κοντά στη φυσική γλώσσα.

Αφού καλέσουμε το component που θέλουμε να εξετάσουμε στη συγκεκριμένη περίπτωση, εδώ είναι το **<BookingForm/>**, δηλώνουμε τα αποτελέσματα που αναμένουμε από συγκεκριμένα πεδία της φόρμας να μην είναι έγκυρα. Αυτό το σενάριο είναι στην περίπτωση που ένας χρήστης πατήσει απευθείας το κουμπί υποβολής (submit) μιας φόρμας χωρίς να έχει συμπληρώσει τα απαραίτητα πεδία ή είναι συμπληρωμένα λάθος. Οπότε στο τέλος αναμένουμε σαν σωστή συμπεριφορά να μην είναι έγκυρη η συγκεκριμένη διαδικασία και να απορριφθεί από τον κώδικα μας με κάποιο μήνυμα λάθους. Στον κώδικα 9.2 θα παραθέσουμε ακόμη ένα σενάριο κατά το οποίο δίνουμε στα πεδία έγκυρες τιμές, κάνουμε υποβολή και αναμένουμε η συνάρτηση που θα πάρει αυτα τα δεδομένα να είναι τα ίδια και έγκυρα. Τέλος Στον κώδικα 9.3 θα δούμε ότι μπορούμε να ορίσουμε και βοηθητικές συναρτήσεις οι οποίες ενδεχομένως να καλεστούν πολλαπλές φορές από τις δοκιμές μας. Μέσω αυτών των συναρτήσεων καλούμαι συγκεκριμένα στοιχεία από το DOM ή από κομμάτια της εφαρμογής μας ώστε να τα εξετάσουμε λεπτομερώς. Ακόμη θα δούμε ότι μπορούμε να ορίσουμε και ενότητες δοκιμών σε ένα αρχείο test και αυτό μας βοηθά ώστε να χωρίζουμε σε ενότητες τα σενάρια που ασχολούνται οι εκάστοτε δοκιμές.

9.4β Δημιουργία React Integration tests

Τα Integration tests είναι ένας τύπος ελέγχου λογισμικού ο οποίος επαληθεύει τον τρόπο με τον οποίο τα unit tests ή ολόκληρες ομάδες unit tests αλληλεπιδρούν και συνεργάζονται μεταξύ τους. Αυτού του είδους οι έλεγχοι διασφαλίζουν ότι το συνολικό σύστημα λειτουργεί σωστά όσον αφορά τις αλληλεπιδράσεις μεταξύ των στοιχείων λογισμικού. Από την στιγμή που τα κομμάτια είναι διαφορετικά μεταξύ τους, οφείλουμε να ελέγχουμε και την ροή δεδομένων μεταξύ τους και αυτή η δοκιμή διεξάγεται με την χρήση integration tests. Με τα integration tests επί της ουσίας εξετάζουμε σενάρια μιας ολόκληρης διαδικασίας και αλληλεπίδρασης του χρήστη με το σύστημα μας, όπως είναι για παράδειγμα η διαδικασία της αυθεντικοποίησης ή η ολοκλήρωση μιας κράτησης.

Οι δοκιμές ενσωμάτωσης (integration tests) αποτελούν ένα σημείο καμπής μεταξύ κόστους και αξίας των δοκιμών. Σύμφωνα με τον **Kent C. Dodds** σε αναφορά του για την δημιουργία δοκιμών: "*Οι δοκιμές ολοκλήρωσης επιτυγχάνουν μια εξαιρετική ισορροπία στις ανταλλαγές μεταξύ εμπιστοσύνης και ταχύτητας/εξόδων. Γι' αυτό είναι σκόπιμο να ξοδεύετε το μεγαλύτερο μέρος της προσπάθειάς εκεί.*"

Τα πλεονεκτήματα των integration tests έναντι των unit tests αποτυπωνονται ώς εξής:

Unit test	**Integration tests**
Κάθε δοκιμή καλύπτει μόνο ένα σημείο ή συνάρτηση της εφαρμογής.	Κάθε δοκιμή καλύπτει ένα ολόκληρο χαρακτηριστικό ή σελίδα της εφαρμογής.
Συχνά χρειάζονται προσαρμογή κατόπιν ανακατασκευής του κώδικα.	Τις περισσότερες φορές επιβιώνει κατόπιν ανακατασκευής του κώδικα.
Δύσκολο να αποφευχθεί η δοκιμή λεπτομερειών υλοποίησης.	Προσομοιάζουν καλύτερα τον τρόπο με τον οποίο οι χρήστες αλληλεπιδρούν με την εφαρμογή.

Πίνακας 9.3 Διαφορές μεταξύ unit και integration tests

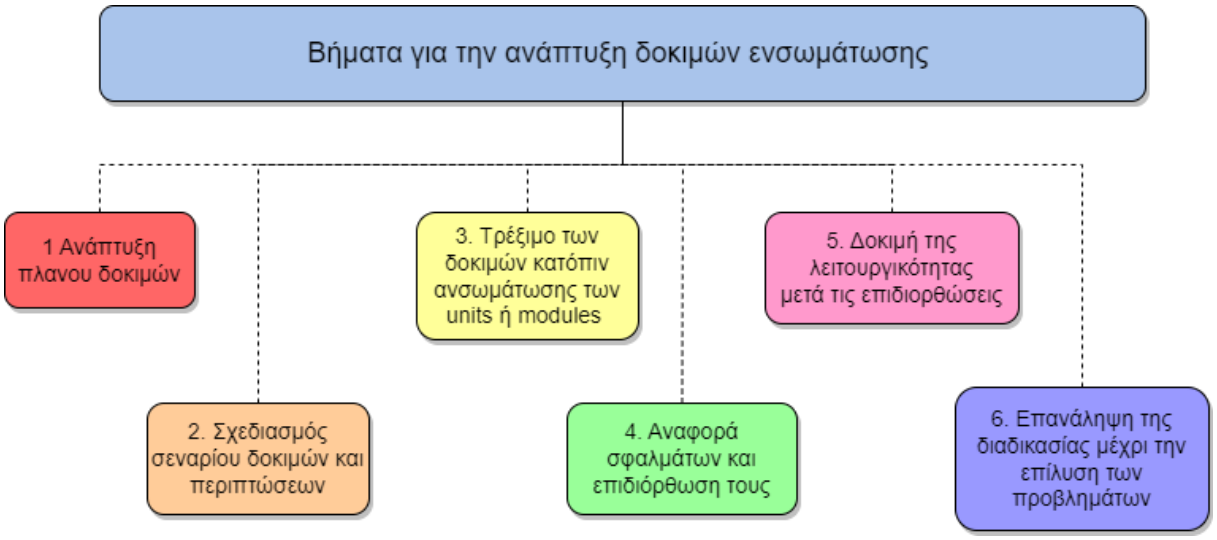
Τα integration tests χρησιμοποιούνται για να διασφαλιστεί ότι οι διάφορες ενότητες ή τα συστατικά στοιχεία ενός συστήματος είναι σε θέση να αλληλεπιδρούν σωστά μεταξύ τους και ότι το συνολικό σύστημα λειτουργεί όπως αναμένεται. Τα integration tests πραγματοποιούνται συχνά μετά τις δοκιμές μονάδας, οι οποίες επικεντρώνονται στη δοκιμή μεμονωμένων μονάδων κώδικα, και πριν από τις δοκιμές αποδοχής, οι οποίες επικεντρώνονται στη δοκιμή του συστήματος στο σύνολό του από τη σκοπιά ενός τελικού χρήστη. Τα integration tests είναι ιδιαίτερα σημαντικά για συστήματα που έχουν πολλές εξαρτήσεις ή που πρέπει να ενσωματωθούν με άλλα συστήματα. Για παράδειγμα, στην δική μας διαδικτυακή εφαρμογή, οι δοκιμές ελέγχουν ότι ένα στοιχείο του frontend επικοινωνεί σωστά με το backend-API ή ότι διάφορα μέρη της εφαρμογής αλληλεπιδρούν σωστά με την βάση δεδομένων μας.

Οι βασικότεροι τύποι δοκιμών ενσωμάτωσης είναι οι εξής:

- **Big-bang testing**: Αυτή η προσέγγιση δοκιμάζει όλα τα στοιχεία του συστήματος ταυτόχρονα, αντί να τα δοκιμάζει σταδιακά. Ουσιαστικά συνδέει όλες τις μεμονωμένες δοκιμές και παράγει ένα ολοκληρωμένο σύστημα, χάνοντας όμως σημεία ιδιαίτερης προσοχής και αντιμετώπισης.
- **Top-down testing**: Αυτή η προσέγγιση ξεκινά με τη δοκιμή των συστατικών του συστήματος σε ανώτερο επίπεδο και προχωρά προς τα χαμηλότερα συστατικά. Η δοκιμή για κάθε στοιχείο γίνεται ένα προς ένα, και στη συνέχεια τα στοιχεία ενσωματώνονται ώστε να διαπιστωθεί πώς λειτουργεί το σύνολο.
- **Bottom-up testing**: Αυτή η προσέγγιση ξεκινά με τη δοκιμή των συστατικών του συστήματος χαμηλότερου επιπέδου και φτάνει μέχρι τα συστατικά υψηλότερου

επιπέδου. Είναι η ακριβώς αντίθετη προσέγγιση με την παραπάνω και είναι κατάλληλη για τη δοκιμή όλων των μονάδων όταν είναι όλες διαθέσιμες.

- **Hybrid testing**: Αυτή η προσέγγιση δοκιμάζει στοιχεία του συστήματος σταδιακά, δηλαδή συνδυασμού και δοκιμής ξεχωριστών ενοτήτων λογισμικού σε μία οντότητα η δοκιμή ομάδων συναφών στοιχείων μαζί. Η δοκιμή ολοκλήρωσης χωρίζεται σε δύο φάσεις: διεπαφή και επικοινωνία. η δοκιμή ολοκλήρωσης γίνεται τόσο από πάνω προς τα κάτω όσο και από κάτω προς τα πάνω προσέγγιση.



Εικόνα 9.12 Βήματα ανάπτυξης δοκιμών ενσωμάτωσης

Καθώς αναπτύξαμε τις δοκιμές ολοκλήρωσης ακολουθήσαμε την τελευταία προσέγγιση κάνοντας συνδυασμό σεναρίων και στοιχείων για να καλύψουμε τις σημαντικότερες λειτουργίες του συστήματος κρατήσεων μας. Ένα πολύ σημαντικό στοιχείο σε αυτή την ανάπτυξη ήταν ότι ο στόχος είναι να ελέγχουμε τον τρόπο με τον οποίο τα διάφορα μέρη του συστήματός μας αλληλεπιδρούν μεταξύ τους με τα μεμονωμένα στοιχεία. Για να επιτευχθεί αυτό συχνά χρειάστηκε η δημιουργία δεδομένων δοκιμής (mock-spies) και περιβαλλόντων δοκιμής, καθώς και το χειρισμό τυχόν εξωτερικών εξαρτήσεων (http calls), παρακάτω θα δούμε ανάλογα παραδείγματα. Δοκιμάσαμε τις αναμενόμενες συμπεριφορές του συστήματος και όχι τις λεπτομέρειες υλοποίησης, καθώς αυτές μπορεί να αλλάζουν συχνά και δεν είναι ο στόχος των δοκιμών ενσωμάτωσης.

Ένα παράδειγμα integration test από την εφαρμογή μας που μπορούμε να εξετάσουμε είναι η διαδικασία της αυθεντικοποίησης ενός χρήστη. Η διαδικασία ξεκινάει από την εισαγωγή των στοιχείων του χρήστη, την αποστολή τους, την επιβεβαίωση μέσω του API, την αποθήκευση

ενός αναγνωριστικού και τέλος την ανακατεύθυνση του χρήστη στην αρχική σελίδα με τα δωμάτια του συστήματος κρατήσεων. Για όλες αυτές τις διαδικασίες υπάρχουν μεμονωμένες δοκιμές για κάθε component ή συνάρτηση των παραπάνω λειτουργιών.

```
describe('A complete flow until rendered all the available games (integration)', () => {
  beforeEach(() => {
    service.auth.useLogin.mockImplementation(() => initialServiceValues);
  });

  it('should complete login and rendered all games', async () => {
    const { user } = renderNode(wrappedLogin);

    await user.type(getUserName(), mockCredentials.body.username);
    await user.type(getPassword(), mockCredentials.body.password);
    await user.click(getRememberMe());

    expect(screen.getByTestId('loginform')).toBeValid();
    expect(getRememberMe()).toBeTruthy();
    await user.click(getLoginButton());

    expect(trackRequest).toBeCalledWith(mockCredentials);

    service.auth.useLogin.mockImplementation(() => ({ ...initialServiceValues, ok: true, data: mockToken }));
    renderNode(wrappedLogin);

    expect(trackSetToken).toBeCalledWith(mockToken.token, false);
    expect(mockedNavigator).toHaveBeenCalled();
    expect(mockedNavigator).toHaveBeenCalledWith('/games', { replace: true });

    expect(trackRequest).toBeCalled();
    service.games.useAll.mockImplementation(() => ({ ...initialServiceValues, ok: true, data: mockGames }));
    renderNode(<Games />);

    expect(getGameOne()).toBeInTheDocument();
    expect(getGameTwo()).toBeInTheDocument();
    expect(mockGames.length).toBe(2);
  });
});
```

Εικόνα 9.13 Δοκιμή ενσωμάτωσης της αυθεντικοποίησης ενός χρήστη

Σε αυτό το σημείο έρχεται να συνδράμει η δοκιμή ενσωμάτωσης η οποία θα δοκιμάσει το σενάριο της εισόδου ενός χρήστη στη διαχειριστική σελίδα του συστήματος μας, χωρίς φυσικά να εξετάσει λεπτομέρειες των επιμέρους λειτουργιών. Στην παραπάνω εικόνα 9.5 βλέπουμε την έναρξη μιας ενότητας δοκιμών (describe) και στην συνέχεια για κάθε δοκιμή την αρχικοποίηση της mock συνάρτησης κλήσης του HTTP Agent και κατά συνέπεια και την επιστρεφόμενη τιμή του. Κατόπιν ξεκινάει η δοκιμή μας με την συμπλήρωση των στοιχείων του χρήστη από τα mock δεδομένα, την υποβολή τους και την κλήση του mock API μας με τα επιστρεφόμενα αποτελέσματα. Εφόσον όλα έχουν πάει όπως αναμενόταν αποθηκευεται το αναγνωριστικό του χρήστη και γίνεται ανακατεύθυνση στο /games. Τέλος μέσω της σύγκρισης των αναμενόμενων αποτελεσμάτων επιβεβαιώνεται και ο στόχος της δοκιμής μας.

```
const mockToken = { token: '123ewq!@#EWQ' };
const mockError401 = { status: 401, message: 'Unauthorized' };
const mockCredentials = {
  body: {
    username: 'username',
    password: 'password',
  },
};
```

```
jest.mock('../../hooks/service', () => ({
  auth: { useLogin: jest.fn() },
  games: { useAll: jest.fn() },
}));

const mockedNavigator = jest.fn();
const mockUseLocationValue = {
  state: {
    from: {
      pathname: '/games',
    },
  },
};
jest.mock('react-router-dom', () => ({
  ...jest.requireActual('react-router-dom'),
  useNavigate: () => mockedNavigator,
  useLocation: () => mockUseLocationValue,
}));

const trackRequest = jest.fn();
const initialServiceValues = {
  ok: false,
  data: null,
  error: null,
  loading: false,
  request: trackRequest,
};
```

Εικόνα 9.14 Ανάπτυξη mock συναρτήσεων και δεδομένων

Στην παραπάνω εικόνα 9.6 βλέπουμε τα mock δεδομένα που εισάγαμε στις δοκιμές μας ώστε να μπορούμε μέσω των βοηθητικών συναρτήσεων που μας παρέχει το Jest με το React testing-library, να προσομοιάσουμε εξωγενείς λειτουργίες και επικοινωνίες του front-end με το υπόλοιπο σύστημα (HTTP calls, backend και βάση δεδομένων). Μέσω της δημιουργίας συναρτήσεων προσομοίωσης και συναρτήσεων “κατάσκοπους” κερδίζουμε πολύτιμο χρόνο στις δοκιμές μας και δεν χρειάζεται να κάνουμε πολύπλοκες διαδικασίες ώστε να επικοινωνεί η εφαρμογή μας με τον έξω κόσμο. Η δοκιμή του συνόλου της εφαρμογής αποτελεί ένα διαφορετικό τύπο δοκιμών (E2E) και δεν εξετάζεται στα integration tests ώστε να μας απασχολεί η εύρυθμη λειτουργία και των εξωγενών συστημάτων και παραγόντων. Τέλος στην τελευταία εικόνα 9.7 παρουσιάζονται τα επιτυχημένα αποτελέσματα όλων των δοκιμών που έγιναν στο front-end της εφαρμογής μας, συνδυασμός μεμονωμένων δοκιμών και ενσωμάτωσης. Μέσω αυτών των δοκιμών καλύψαμε ένα σημαντικό ποσοστό

της εφαρμογής μας σε κρίσιμα σημεία που απαιτούσαν έλεγχο για να διαπιστωθεί η εύρυθμη λειτουργία της εφαρμογής.

Εικόνα 9.15 Αποτελέσματα των δοκιμών του front-end της εφαρμογής μας

9.5β Testing και Code Validation στην GO

Η Go παρέχει το testing package για την δημιουργία δοκιμών του κώδικα καθώς και το εργαλείο το gopls, έναν Language Server που παρέχει στο IDE λειτουργίες για την ανάλυση του κώδικα κατα την συγγραφή του.

9.5.1β Code Validation

Language Server είναι ένας διακομιστής που έχει ρυθμιστεί για να χρησιμοποιεί το πρωτόκολλο διακομιστή γλώσσας προκειμένου να παρέχει υποστήριξη εργαλείων για τον επεξεργαστή κειμένου ή το IDE που χρησιμοποιείται για την ανάπτυξη κώδικα. Το Language Server Protocol (LSP) ορίζει το πρωτόκολλο που χρησιμοποιείται μεταξύ ενός επεξεργαστή κειμένου ή ενός IDE και ενός διακομιστή γλώσσας που παρέχει λειτουργίες γλώσσας όπως αυτόματη συμπλήρωση, μετάβαση στον ορισμό, επαλήθευση σύνταξης και άλλων χρήσιμών λειτουργιών. Το LSP αναπτύχθηκε για να μειώσει το κόστος υλοποίησης εργαλείων ανάπτυξης καθώς διαφορετικά IDEs απαιτούν επιπλέων προσπάθεια για την ανάπτυξη τέτοιον εργαλείων για αυτά. Οποιαδήποτε ενημέρωση στις προδιαγραφές εργαλείων σημαίνει πολλαπλές ενημερώσεις σε διαφορετικά εργαλεία. Οι Language Server προωθούν την ιδέα της υποστήριξης της ανάπτυξής κώδικα από το νέφος (cloud). Αυτό σημαίνει ότι:

- Δεν υπάρχει περίπλοκη ρύθμιση για εργαλεία στο τοπικό περιβάλλον ανάπτυξης. Αφού εγκατασταθεί ο Language Server, θα πρέπει να χειρίζεται τα υπόλοιπα εργαλεία που χρειάζονται για να υποστήριξη της συγκεκριμενεις γλώσσας προγραμματισμού.

  Για παράδειγμα, το vscode-go απαιτεί συνήθως την εγκατάσταση πολλαπλών πακέτων όπως gocode, gopkg, gosymbols, go-outline, κλπ. Όλα αυτά αντικαθίστανται από ένα μόνο πακέτο που ονομάζεται gopls.

- Είναι εύκολο ο προγραμματιστής να ακολουθεί τις ενημερωμένες βέλτιστες πρακτικές. Καθώς χρησιμοποιώντας έναν τυπικό Language Server που λαμβάνει όλες τις ενημερώσεις μπορεί να χρησιμοποιηθεί για τον έλεγχο διαφορετικών έργων.

Το gopls είναι το εργαλείο LSP που αναπτύσει και παρέχει η ομάδα της Golang για την υποστήριξη στην ανάπτυξη κώδικα Go. Το gopls σε συνδυασμό με ένα σύγχρονο IDE ή επεξεργαστή κειμένου μπορεί να αναγνωρίζει τα αρχεία go και να τα αναλύει για να επαληθεύει οτι δεν υπάρχουν συντακτικά λάθη που δεν θα επιτρέπουν το compile του πηγαίου κώδικα καθώς και να εντοπίζει ορισμένα λογικά σφάλματα που μπορεί να προκύψουν κατα την εκτελεση του προγραμματος. Επίσης εφαρμόζει ορισμένους κανόνες για την μορφή του κώδικα σε ένα αρχείο και μπορεί να τροποποιεί τα αρχεία για να πληρούν τους συγκεκριμένους κανόνες. Επίσης επιτρέπει την ευκολότερη περιήγηση στον πηγαίο κώδικα καθώς μπορεί να αναγνωρίσει τα στοιχεία της γλώσσας και να παρέχει πληροφορίες για αυτά όπως τους ορισμούς τους και την προέλευσή τους. Η χρήση του gopls αποτρέπει τον προγραμματιστη από το να γράφει κώδικα με λάθη, βοηθάει να εντοπίζει τα σφάλματα γρηγορότερα, να ανατρέχει στο Documentation κάποιου module ή package και να τηρεί τις βέλτιστες πρακτικές. Επίσης η αυστηρή δομή του κώδικα που επιβάλλει έχει σαν αποτέλεσμα ο πηγαίος κώδικας να είναι γραμμένος με κοινό τρόπο ώστε να μπορεί να ελεγχθεί ή να τροποποιηθεί πιο εύκολα και μετά την ολοκλήρωση του. [39]

Παρακάτω παραθέτουμε ορισμένα παραδείγματα χρήσης του gopls στον επεξεργαστή κειμένου VSCode.

Μορφοποίηση κώδικα

Αριστερά βλέπουμε τα imported packages πριν την εφαρμογή του gopls και δεξιά μετά την εφαρμογή του. Το gopls ταξινόμησε τα packages σε αλφαβητική σειρά και τα οργάνωσε βάζοντας πρώτα αυτά που προέρχονται από την βασική βιβλιοθήκη της Go και μετά τα packages που προέρχονται από τρίτες ομάδες ή το ίδιο το module.

```
import (
"encoding/json"
"net/http"
"strings"
"github.com/labstack/echo/v4"
"gitlab.com/khakibee/khakibee/api/game"
"testing"
"time"
"gitlab.com/khakibee/khakibee/api/model"
"gitlab.com/khakibee/khakibee/api/testUtils"
"strconv"
)
```

```
import (
"encoding/json"
"net/http"
"strconv"
"strings"
"testing"
"time"

"github.com/labstack/echo/v4"
"gitlab.com/khakibee/khakibee/api/game"
"gitlab.com/khakibee/khakibee/api/model"
"gitlab.com/khakibee/khakibee/api/testUtils"

)
```

Εντοπισμός Σφαλμάτων

Στην εικόνα 9.8 φαίνεται το VSCode να έχει σημειώσει το σφάλμα που εντόπισε αλλα και να παρέχει το μήνυμα σφάλματος και τρόπους διόρθωσης αν υπάρχουν. Χωρίς το gopls το συγκεκριμένο σφάλμα θα μπορούσε να περάσει απαρατήρητο και να οδηγήσει σε αποτυχημένη προσπάθεια compile του κώδικα και την ανάγκη αποσφαλμάτωσης του.

Εικόνα 9.16 Εντοπισμός σφάλματος από το gopls

Προβολή και μετάβαση σε ορισμούς

Στην εικόνα 9.9 φαίνεται ο ορισμός της συνάρτησης NoContent καθώς και το σχόλιο του προγραμματιστή που αφορά την λειτουργία της αλλα και από ποιό package προέρχεται. Επίσης το NoContent εμφανίζεται με την τυπική μορφή συνδέσμου καθώς λειτουργεί σαν

σύνδεσμος για το αρχείο που περιέχεται ο ορισμός της για την ευκολότερη πλοήγηση στα αρχεία του πηγαίου κώδικα.

Εικόνα 9.17 Προβολή ορισμού συνάρτησης

9.5.2β Testing

Το package testing της Go παρέχει υποστήριξη για αυτοματοποιημένες δοκιμές των Go packages. Το package testing προορίζεται να χρησιμοποιηθεί σε συνδυασμό με την εντολή "go test", η οποία αυτοματοποιεί την εκτέλεση οποιασδήποτε testing function του package.

Test function

Τα Test functions πρέπει να έχουν την μορφή **func TestXxx(*testing.T)** όπου το Xxx δεν ξεκινά με πεζό γράμμα. Το όνομα της συνάρτησης χρησιμεύει για τον προσδιορισμό της ρουτίνας test. Σε αυτές τις λειτουργίες, χρησιμοποιούμε τις μεθόδους Error, Fail ή σχετικές μεθόδους για να σηματοδοτήσουμε την αποτυχία. Για να γράψουμε μια νέα σουίτα testing, δημιουργούμε ένα αρχείο που περιέχει τις λειτουργίες TestXxx όπως περιγράφονται εδώ και δίνουμε σε αυτό το αρχείο ένα όνομα που τελειώνει σε "_test.go". Το αρχείο θα εξαιρεθεί από τις κανονικές εκδόσεις πακέτων, αλλά θα συμπεριληφθεί όταν εκτελείται η εντολή "go test". Το αρχείο test μπορεί να βρίσκεται στο ίδιο package με αυτό που δοκιμάζεται ή σε αντίστοιχο package με το επίθημα "_test". Εάν το αρχείο δοκιμής βρίσκεται στο ίδιο πακέτο, μπορεί να αναφέρεται σε μη εξαγόμενα αναγνωριστικά εντός του package, ενώ εάν το αρχείο βρίσκεται σε ξεχωριστό package "_test", το package που ελέγχεται πρέπει να εισαχθεί ρητά και μπορούν να χρησιμοποιηθούν μόνο τα εξαχθέντα αναγνωριστικά του. Αυτό είναι γνωστό ως "black box" testing. Το "black box" testing μας υποχρεώνει να συμπεριφερθούμε στο package προς δοκιμή όπως όταν το χρησιμοποιούμε σε ένα module οπότε τα tests θα

ανταποκρίνονται περισσότερο στην πραγματική λειτουργία του. Στον κώδικα 9.4 βλέπουμε την συνάρτηση TestAbs που υλοποιεί μία δοκιμή για την ρουτίνα Abs του package abs και καλεί την μέθοδο errorF στην περίπτωση που η συνάρτηση abs δεν φέρει το αναμενόμενο αποτέλεσμα.

Το T είναι ένας τύπος που μεταβιβάζεται στις Test functions για τη διαχείριση του test state και την υποστήριξη μορφοποιημένων test logs. Μία δοκιμή (Test) τελειώνει όταν η Test function επιστρέφει ή καλεί οποιαδήποτε από τις μεθόδους **FailNow**, **Fatal**, **Fatalf**, **SkipNow**, **Skip** ή **Skipf**. Αυτές οι μέθοδοι, καθώς και η μέθοδος **Parallel**, πρέπει να καλούνται μόνο από το goroutine που εκτελεί τη λειτουργία Test. Οι άλλες μέθοδοι αναφοράς, όπως οι παραλλαγές των Log και Error, μπορούν να καλούνται ταυτόχρονα από πολλαπλές γορουτίνες. Εν κατακλείδι μια δοκιμή (Test function) θα είναι υπεύθυνη για την δοκιμή μιας ρουτίνας, η οποία μπορεί να είναι ένα unit test ή ένα integration test. Οι λειτουργίες που πρέπει να κάνει μία δοκιμή είναι:

- Να δημιουργεί και να αρχικοποιεί τα στιγμιότυπα και τις μεταβλητές που χρειάζονται.
- Να ορίζει τις αναμενόμενα τιμές για κάθε περίπτωση χρήσης.
- Να εκτελεί τις λειτουργίες προς δοκιμή με τις κατάλληλες παραμέτρους για κάθε περίπτωση.
- Να επαληθεύει το αποτέλεσμα με βάση τις αναμενόμενες τιμές που έχουν οριστεί και να καταγράφει το αποτέλεσμα της δοκιμής σαν επιτυχία ή σφάλμα.

Test Utilities

Το package testUtils περιέχει βοηθητικές συναρτήσεις και τιμές που επαναχρησιμοποιούνται ώστε να παραμένουν τα Test functions καθαρά και να υλοποιουν μόνο τα σενάρια των δοκιμών. Στο testUtils υπάρχουν οι συναρτησεις assert. Αυτές οι συναρτήσεις ονομάζονται AssertXxx, όπου στο Xxx θα μπαίνει η περιγραφή της λειτουργίας τις, και σκοπός τους είναι η σύγκρισή του αποτελέσματος της ρουτίνας test με την αναμενόμενη τιμή και η καταγραφή του σφάλματος σε περίπτωση που το αποτέλεσμα δεν είναι το αναμενόμενο. Στον κώδικα 9.5 βλέπουμε την συναρτηση **AssertHTTPStatus** που συγκρίνει το αποτέλεσμα με την αναμενόμενη τιμή και εμφανίζει κατάλληλο μήνυμα στην περίπτωση που δεν ταιριάζουν. Η μέθοδος Helper ορίζει την συνάρτηση σαν βοηθητική ώστε να καταγραφεί το σημείο κλίσης της συνάρτησης assert σαν σφάλμα και όχι η κλήση της μεθόδου Errorf που θα σημειονόταν αρχικά.

Επίσης στις βοηθητικές συναρτήσεις υπάρχουν συναρτήσεις όπως η NewGetRequest που προετοιμάζει ένα get HTTP request για την εκτέλεση ενος HTTP handler. Η υλοποίησή της φαίνεται στον κώδικα 9.6

Και τέλος, στο testUtils υπάρχει το struct DB που περιέχει ορισμένα εικονικά δεδομένα που θα χρησιμοποιηθούν από τις δοκιμές αντι της βάσης δεδομένων. Στον κώδικα 9.7 φαίνεται το struct DB και τα εικονικά δεδομένα που αφορούν το Schedule.

Dependency injection

Στη τεχνολογία λογισμικού, dependency injection είναι ένα design pattern στο οποίο ένα αντικείμενο ή μια συνάρτηση λαμβάνει άλλα αντικείμενα ή λειτουργίες από τις οποίες εξαρτάται. Το dependency injection στοχεύει να διαχωρίσει τις ανησυχίες της κατασκευής αντικειμένων και της χρήσης τους, οδηγώντας σε χαλαρά συνδεδεμένα προγράμματα.Το μοτίβο διασφαλίζει ότι ένα αντικείμενο ή συνάρτηση που θέλει να χρησιμοποιήσει μια δεδομένη υπηρεσία δεν πρέπει να γνωρίζει πώς να κατασκευάσει αυτές τις υπηρεσίες. Αντίθετα, ο «πελάτης» λήψης (αντικείμενο ή συνάρτηση) παρέχεται σαν dependency από εξωτερικό κωδικά (ένας «injector»), τον οποίο δεν γνωρίζει. Το dependency injection βοηθά κάνοντας σαφείς τις υπονοούμενες εξαρτήσεις (dependencies) και βοηθά στην απομόνωση των λειτουργιών και στην δυνατότητα αντικατάστασης των εξαρτήσεων χωρίς να επηρεάζουν το αντικείμενο ή συνάρτηση που τα χρησιμοποιεί.[40]

Μέσω του dependency injection μπορούμε να αντικαταστήσουμε τις εξαρτήσεις (dependencies) που δεν αφορούν την δοκιμή με εικονικές υλοποιήσεις αυτών των λειτουργιών. Για την λειτουργία του HTTP server, του database server και του smtp server θα χρησιμοποιήσουμε τα packages net/http/httptes, go-sqlmock και go-smtp-mock που υλοποιούν μία εικονική λειτουργία των διακομιστών. Με αυτόν τον τρόπο μπορούμε να επιβεβαιώνουμε την σωστή λειτουργία του κώδικα απομονώνοντας τον από εξωτερικούς παράγοντες. Επίσης θα χρειαστεί να φτιάξουμε mock stores και email methods που θα υλοποιούν τα αντίστοιχα interfaces. Με αυτον τον τρόπο θα μπορούμε να στοχεύουμε στην λειτουργικότητα που μας ενδιαφέρει χωρίς επιρροές από dependencies.

Στον κώδικα 9.8 βλέπουμε το struct MockGameStore που υλοποιεί το interface GameStore και στον κώδικα 9.9 την υλοποίηση της εικονικής InsertGame. Το struct περιέχει τα δεδομένα που χρειάζεται το συγκεκριμένο store και κάποια βοηθητικά πεδία για την επιβεβαίωση

ορισμένων λειτουργιών. Η μέθοδος InsertGame ενημερώνει το counter InsertCalled ότι εκτελέστηκε ακόμα μια φορά και αν έχει οριστεί να επιστρέψει error, το επιστρέφει. Καλώντας την εικονική μέθοδο InsertGame μπορούμε να δοκιμάσουμε ότι κάποιος handler την εκτελεί όταν χρειάζεται και ότι μπορεί να διαχειριστεί το σφάλμα αν προκύψει.

Παράδειγμα

Στον κώδικα 9.10 βλέπουμε ένα παράδειγμα unit testing για τον handler getGame. Η δοκιμή (Test function) ονομάζεται TestGetGame. Αφού δημιουργήσει τα στιγμιότυπα που χρειάζεται τρέχει ορισμένα σενάρια στον handler getGame. Τα σενάρια αφορούν την επιτυχή ανάκτηση του game, την περίπτωση που το game που ζητείται δεν υπάρχει και την περίπτωση που προκύπτει κάποιο σφάλμα στην εκτέλεση του store. Για κάθε σενάριο ορίζει το αναμενόμενο αποτέλεσμα, τρέχει την getGames με τα αντίστοιχα ορίσματα και επαληθεύει το αποτέλεσμα μέσω των assert συναρτήσεων. Εκτελώντας την εντολή go test μπορούμε να επιβεβαιώσουμε την σωστή λειτουργία του package game ή να δούμε τα σφάλματα που προέκυψαν.

Αποτέλεσμα Go test χωρίς σφάλματα:

```
PASS
ok      gitlab.com/khakibee/khakibee/api/game   0.141s
```

Αποτέλεσμα Go test με σφάλματα:

```
-- FAIL: TestGetGame (0.00s)
    --- FAIL: TestGetGame/Get_game_not_found (0.00s)
        game_test.go:150: wanted http status 404 but got http status 422
FAIL
exit status 1
FAIL    gitlab.com/khakibee/khakibee/api/game   0.246s
```

Κεφάλαιο 10ο

Μελλοντικές Επεκτάσεις

Οι επεκτάσεις των εφαρμογών αποτελούν ένα σύνηθες φαινόμενο καθώς στον χώρο της τεχνολογίας συνεχώς προκύπτουν νέες ανάγκες και προκλήσεις. Προκειμένου ένα προϊόν να επιβιώσει στον κόσμο της τεχνολογικής επανάστασης οφείλει να προσαρμόζεται και να πρωτοπορεί σε χαρακτηριστικά και λειτουργίες. Στα πλαίσια της διπλωματικής εργασίας αναπτύχθηκε το υπάρχον σύστημα κρατήσεων ως το Ελάχιστο Βιώσιμο Προϊόν (MVP) στην κατηγορία των κρατήσεων. Για την ανάπτυξη του συστήματος κρατήσεων έγινε σχεδίαση και πρόβλεψη για περαιτέρω μελλοντική ανάπτυξη και επέκταση του συστήματος με επιπλέον λειτουργίες. Κατά την ανάπτυξη των λειτουργιών επιλέχθηκαν οι βασικότερες ώστε το προϊόν να αποτελεί από μόνο του ένα αξιοποιήσιμο εργαλείο για επιχειρήσεις ψυχαγωγίας. Η διαδικτυακή μας εφαρμογή ως υπηρεσία εκπληρώνει τους αρχικούς της στόχους όπως η δημιουργία και διαχείριση υπηρεσιών, η δημιουργία και διαχείριση κρατήσεων από διαχειριστές και πελάτες, η δημιουργία και διαχείριση προγραμμάτων λειτουργίας, ενημέρωση πελατών και χρηστών μέσω email κλπ. Κατά την σχεδίαση προέκυψαν διάφορες χρήσιμες λειτουργίες που μπορούν να ενταχθούν στην εφαρμογή μας με σκοπό την αναβάθμιση και εξέλιξη της στο χώρο των κρατήσεων.

10.1 Εγγραφή στο σύστημα και αυτόματη έναρξη της εφαρμογής

Μια πολύ χρήσιμη λειτουργία που βοηθά καθοριστικά στην επεκτασιμότητα και αυτοματοποίηση της εφαρμογής μας είναι να μπορούμε να ανεβάζουμε ένα αντίγραφο της εφαρμογής μας σε κάποιο διακομιστή σύμφωνα με τις ανάγκες μας κατόπιν εγγραφής ενός πελάτη του συστήματος αγοράζοντας το σύστημα κρατήσεων ως υπηρεσία. Ο πελάτης που ενδιαφέρεται για το σύστημα κρατήσεων θα συμπληρώνει μια φόρμα με τα απαραίτητα πεδία που χρειάζεται η εφαρμογή και αφού επιλέξει-πληρώσει το πακέτο των υπηρεσιών που τον ενδιαφέρει, στην συνέχεια το σύστημα μας κυρίως σε συστημικό επίπεδο (DevOps) θα αναλαμβάνει μέσω χρήσης κατάλληλων αντιγράφων (Instances) του docker-kubernetes να ανεβάζει την εφαρμογή σύμφωνα με τα κριτήρια που έδωσε ο χρήστης και καταλαμβάνοντας μονο τους πόρους που απαιτούνται. Ο ενδιαφερόμενος πελάτης θα λαμβάνει τα απαραίτητα

στοιχεία και διαπιστευτήρια μέσω email και φυσικά θα γίνεται επιβεβαίωση της ύπαρξης του πελάτη για αποφυγή σπατάλης πόρων. Η εφαρμογή του πελάτη σύμφωνα με τις ανάγκες του θα ανεβαίνει στο διαδίκτυο σε μερικά λεπτά και θα του παρέχεται το σύστημα κρατήσεων με τις ιδιαίτερες απαιτήσεις που επέλεξε ώς υπηρεσία (SaaS). Κατόπιν ο πελάτης θα μπορεί να παραμετροποιήσει το σύστημα σύμφωνα με τις ανάγκες του χωρίς να χρειαστεί να εμπλακεί σε πολύπλοκες και χρονοβόρες διαδικασίες επικοινωνίας με την επιχείρηση κατασκευής ή μεταπώλησης της εφαρμογής.

10.2 Διακομιστής επικοινωνιών (SMS/Email/Push notifications)

Η επικοινωνία της εφαρμογής με τους χρήστες είναι μία πολύ σημαντική λειτουργία που στις μέρες μας θεωρείται αναγκαιότητα καθώς ενημερώνει τους πελάτες για τα ζητήματα που τους αφορούν και επιπλέον προσωποποιεί τα μηνύματα της εφαρμογής δίνοντας στον πελάτη μεγαλύτερη αξία. Τα μέσα επικοινωνίας μιας διαδικτυακής εφαρμογής μπορεί να είναι πολλών ειδών αλλά τα σημαντικότερα είναι τα μηνύματα στο κινητό τηλέφωνο (SMS), μηνύματα στο ηλεκτρονικό ταχυδρομείο (Emails) και προωθητικές ειδοποιήσεις. Για την βέλτιστη διαχείριση και επεκτασιμότητα των επικοινωνιών ενός συστήματος πολύ καλή πρακτική είναι η μελλοντική δημιουργία ενός ξεχωριστού διακομιστή που θα έχει σαν αρμοδιότητα τον χειρισμό της επικοινωνίας του συστήματος με τους πελάτες. Αυτή η ξεχωριστή οντότητα θα είναι πλήρως υλοποιημένη από την ομάδα και δεν θα εναπόκειται σε εξωτερικές εξαρτήσεις που κλειδώνουν πολλές φορές μια επιχείρηση για μελλοντικές αλλαγές και επεκτάσεις. Ο διακομιστής αυτός θα επικοινωνεί μόνο με όσα κομμάτια της κύριας εφαρμογής χρειάζεται όπως είναι η βάση δεδομένων και πιθανόν κάποια APIs που σχετίζονται με αλλαγές που μπορεί να προκύπτουν οι χρονισμένα μηνύματα που πρέπει να αποστέλλονται σε πελάτες. Τα SMS θα χρησιμοποιούνται από την εφαρμογή για πιο υψηλού επιπέδου ενημερωτικά στοιχεία καθώς είναι μία υπηρεσία που κοστίζει και χρησιμοποιείται κυρίως για περιπτώσεις αυθεντικοποίησης, επιβεβαίωση κράτησης ή άλλα κρίσιμα κομμάτια της εφαρμογής. Τα emails θα χρησιμοποιούνται σε πιο εκτεταμένη χρήση καθώς είναι δωρεάν και αποτελούν ένα σύνηθες τρόπο για ενημέρωση πελατών κατόπιν κρατήσεων ή επεξεργασίας της κατάστασης τους, εγγραφής στην εφαρμογή, ετεροχρονισμένων μηνυμάτων υπενθύμισης, κλπ. Για την χρήση αυτού του τρόπου επικοινωνίας επιβάλλεται η χρήση ενός απλού και ευανάγνωστου πλαισίου email που θα παρουσιάζει στον πελάτη μια καλαίσθητη εικόνα για την επιχείρηση και τον χειρισμό των πελατών. Τέλος η χρήση προωθητικών μηνυμάτων θα έχει ένα βοηθητικό ρόλο όσον αφορά την διαχειριστική σελίδα ώστε να ενημερώνει τους διαχειριστές δωματίων αλλά και της επιχείρησης για αλλαγές πάνω

σε κρατήσεις. Με αυτόν τον τρόπο θα έχουν ένα γρήγορο τρόπο να ενημερώνονται για τις τελευταίες εξελίξεις και αλλαγές στα δωμάτια όσο απουσιάζουν από την εφαρμογή.

10.3 Διαχείριση χρηστών και έλεγχος πρόσβασης

Οι χρήστες είναι το πιο βασικό στοιχείο κάθε εφαρμογής έτσι και στην δική μας περίπτωση η διαχείριση των χρηστών της εφαρμογής αποτελεί σημαντική λειτουργία. Οι χρήστες μας χωρίζονται σε δύο βασικά είδη είναι οι χρήστες της εφαρμογής ώς πελάτες της επιχείρησης και οι χρήστες του διαχειριστικού συστήματος. Ξεκινώντας από τους χρήστης πελάτες αποτελούν την πελατεία της επιχείρησης και την πηγή εσόδων της συνεπώς, μία σημαντική λειτουργία που θα μπορούσε μελλοντικά να προστεθεί στην εφαρμογής είναι πέραν της καταγραφής τους, η απεικόνιση των πελατών σε λίστα με οικονομικά στοιχεία, συνέπεια απέναντι στις κρατήσεις τους, κριτικές κλπ. Αυτές οι συσσωρευμένες πληροφορίες θα μπορούν να αποτυπώσουν για την επιχείρηση μια καλύτερη εικόνα για το είδος και την ποιότητα των πελατών τους, ώστε να γίνεται καλύτερη διαχείριση και στοχευμένες προωθητικές ενέργειες. Στο δεύτερο είδος των χρηστών που είναι αυτοί του συστήματος, μελλοντικές λειτουργίες που μπορούν να προστεθούν και έχουν προστιθέμενη αξία στο σύστημα κρατήσεων μας είναι η διαχείριση του προγράμματος εργασίας των υπαλλήλων της επιχείρησης καθώς και του ρόλου και των αρμοδιοτήτων-δυνατοτήτων πάνω στο σύστημα. Πιο συγκεκριμένα η προσθήκη χρηστών από τον πλήρη διαχειριστή ή άλλο αρμόδιο του συστήματος και η διαχείριση τους αναλόγως του ρόλου που τους δίνεται αποτελεί ένα μέσο για τον επιχειρηματία για την διαχείριση των υπαλλήλων του, των αρμοδιοτήτων και λοιπών λεπτομερειών στα δωμάτια απόδρασης. Επιπρόσθετα η δημιουργία και διαχείριση του προγράμματος εργασίας των υπαλλήλων της επιχείρησης πάνω στο πρόγραμμα λειτουργίας είναι πολύ σημαντική λειτουργία και σε συνδυασμό με την ενημέρωση των υπαλλήλων με το πρόγραμμα εργασίας τους επιλύονται καθημερινά προβλήματα λειτουργίας, συγχύσεων και προγραμματισμού.

10.4 Τιμές και Προσφορές

Στον σύγχρονο κόσμο όπου η ανταγωνιστικότητα αποτελεί βασικό στοιχείο της αγοράς μια σημαντική λειτουργία για την επιχείρηση είναι η διαχείριση της τιμολογιακής πολιτικής και η δημιουργία και διαχείριση προσφορών για μεγαλύτερη προσέλκυση πελατών. Ανά τακτά χρονικά διαστήματα οι επιχειρήσεις προχωρούν σε αξιολόγηση της πολιτικής της επιχείρησης

και μέσα σε αυτά τα πλαίσια εντάσσεται και η τιμολογιακή της πολιτική. Οπότε ένα συχνό φαινόμενο είναι η διακύμανση των τιμών των υπηρεσιών των δωματίων ψυχαγωγίας, αυτή ή λειτουργία φυσικά έχει πολλές επιπτώσεις στο σύστημα μας καθώς δεν θα πρέπει να διαταράσσονται οι υπάρχουσες κρατήσεις του συστήματος. Η επόμενη λειτουργία που αποτελεί και το μεγαλύτερο κομμάτι πρόκλησης από πλευράς σχεδίασης και υλοποίησης για το σύστημα μας είναι οι προσφορές. Η δημιουργία προσφορών μπορεί να γίνει οριζόντια με απλή λογική εκπτώσεις στους πελάτες για τις ώρες που η ζήτηση είναι συγκριτικά χαμηλότερη με αυτές μεγαλύτερης κίνησης. Σε μια μελλοντική όμως καλά σχεδιασμένη αναβάθμιση σε αυτό το κομμάτι θα μπορούσε να ενταχθεί τεχνητή νοημοσύνη (AI) ώστε να γίνεται στοχευμένη προώθηση προσφορών. Η ανταμοιβή καλών πελατών με γενναίες προσφορές, οι προσφορές σε πελάτες σε ώρες χαμηλής ζήτησης, που έχουν δείξει ενδιαφέρον (π.χ μέσω λιστών αναμονής) για υψηλής ζήτησης ώρες και οι προσφορές σε πελάτες που ακύρωσαν τις κρατήσεις τους είναι μερικά μόνο παραδείγματα από τα πεδία εφαρμογής που μπορούν να χρησιμοποιηθούν οι προσφορές με τελικό σκοπό την αύξηση του πελατολογίου της επιχείρησης.

10.5 Ηλεκτρονικές Πληρωμές

Ένα μεγάλο κομμάτι των υπηρεσιών και προϊόντων που πωλούνται σήμερα στο διαδίκτυο διαθέτουν μια επιβεβλημένη ανάγκη που είναι οι ηλεκτρονικές πληρωμές. Στις μέρες μας η χρήση του διαδικτύου και της διάθεσης των προϊόντων είναι κομμάτι της καθημερινότητας μας καθώς προσφέρει ευκολία στην αγορά, ταχύτητα και πολλαπλές επιλογές. Συνεπώς η προσθήκη ηλεκτρονικής πληρωμής για την αγορά των υπηρεσιών της εκάστοτε επιχείρησης του συστήματος κρατήσεων είναι μια επιβεβλημένη ανάγκη. Μέσω αυτής της μελλοντικής προσθήκης το σύστημα μας θα προσφέρει στους χρήστες την ευκολία της πληρωμής με ηλεκτρονικό χρήμα, του άμεσου ελέγχου των οικονομικών στοιχείων και την αποτροπή ακυρώσεων από τους πελάτες που επιφέρει ζημία για την επιχείρηση. Φυσικά οι ηλεκτρονικές πληρωμές ενέχουν κινδύνους καθώς εκτίθενται στο διαδίκτυο όπως είναι υποκλοπές των στοιχείων ηλεκτρονικών καρτών και προσωπικών στοιχείων. Γι αυτό το λόγο η μελλοντική αυτή προσθήκη θα πρέπει να γίνει στηριζόμενη στις βέλτιστες πρακτικές και εξασφαλίζοντας την ανωνυμία των πελατών και των στοιχείων τους πράγμα που αντικατοπτρίζει στην αξιοπιστία της επιχείρησης και την εικόνα της απέναντι στους πελάτες.

Συνολικά στην διαδικτυακή εφαρμογή μας πέραν από ένα απλό σύστημα κρατήσεων δόθηκε η κατεύθυνση για την τελική δημιουργία ενός εργαλείου για τον υπεύθυνο διαχειριστή των δωματίων ψυχαγωγίας. Μέσω της ένταξης των υπηρεσιών που θα του προσφέρονται ως υπηρεσίες (SaaS) και του πλάνου που θα επιλέγει από τις προσφερόμενες λειτουργίες του συστήματος θα μπορεί να παραμετροποιεί το σύστημα κρατήσεων στις ανάγκες της επιχείρησης. Όλα τα χαρακτηριστικά και οι λειτουργίες του συστήματος κρατήσεων έχουν σαν στόχο την παροχή στους τελικούς πελάτες της μέγιστης διαδικτυακής εμπειρίας και ευκολίας στην πραγματοποιήση των κρατήσεων τους αλλα και την βέλτιστη διαχείριση της συνολικής λειτουργίας των δωματίων ψυχαγωγίας από τους διαχειριστές της επιχείρησης. Ένα από τα βασικά ζητούμενα είναι η καλύτερη διαχείριση με όσο το δυνατόν λιγότερες ενέργειες γίνεται από έναν επιχειρηματία-διαχειριστή ώστε να μπορεί να αξιοποιεί την προσπάθεια σου σε άλλους τομείς της επιχείρησης. Αυτό είναι άλλωστε και το μεγάλο πλεονέκτημα και όφελος από την τεχνολογία, η δημιουργία ευκολιών και επίλυσης προβλημάτων προς όφελος μας.

Κεφάλαιο 11ο

Συμπεράσματα

Οι χώροι ψυχαγωγίας αποτελούν μία σχετικά πρόσφατη δραστηριότητα για το κοινό και όπως είναι φυσικό σε κάθε νέο κλάδο προκύπτουν νέες ανάγκες και ελλείψεις. Μία βασική έλλειψη που υπάρχει για τις συγκεκριμένες υπηρεσίες είναι ένα ηλεκτρονικό σύστημα κρατήσεων που να καλύπτει τις ιδιαίτερες απαιτήσεις τέτοιων χώρων. Αυτό το οργανωτικό εργαλείο θα επιτρέπει για πρώτη φορά τον συνδυασμό της δημιουργίας προκαθορισμένου ωραρίου λειτουργίας, της δυνατότητας κρατήσεων απευθείας από τους πελάτες, της ομαλής εναλλαγής του ωραρίου λειτουργίας σύμφωνα με τις ανάγκες της επιχείρησης, καθώς και την δυνατότητα διανομής του λογισμικού μέσω του διαδικτύου. Προκειμένου να επιλυθούν αυτά τα ζητήματα αποφασίσαμε να κάνουμε μία ολιστική και σφαιρική προσέγγιση των παραπάνω προβλημάτων, αναπτύσσοντας μία διαδικτυακή εφαρμογή για την διαχείριση κρατήσεων σε αυτού του τύπου εγκαταστάσεις, που θα διανέμεται σαν υπηρεσία. Η ανάπτυξη της δεν επικεντρώθηκε μονάχα στην ικανοποίηση των προαναφερθεισών αναγκών αλλά συνολικά στην σχεδίαση και ανάπτυξη λογισμικού. Σύμφωνα με τις τυποποιημένες μεθοδολογίες τεχνολογίας λογισμικού καταφέραμε την ταχεία ανάπτυξη μιας εφαρμογής που να πληρεί τις ελάχιστες απαιτούμενες λειτουργίες ώστε να είναι άμεσα διαθέσιμη προς χρήση. Κεντρικοί στόχοι της σχεδίασης ήταν, πέραν της κάλυψης των ιδιαιτεροτήτων των χώρων ψυχαγωγίας, η εφαρμογή να μπορεί να επαληθευτεί και να επικυρωθεί, να μπορεί να εξελιχθεί, συντηρηθεί και διανεμηθεί εύκολα, σημαντικό ζητούμενο από τις σύγχρονες εφαρμογές. Τέλος, ένα ακόμη κύριο σημείο ενδιαφέροντος ήταν η ανάπτυξή της με τεχνολογίες και πρακτικές που να παρέχουν στους πελάτες εύκολη προσβασιμότητα από όλες τις σύγχρονες συσκευές διασύνδεσης στο Διαδίκτυο και να προσφέρουν τελικά στον χρήστη την μέγιστη δυνατή εμπειρία. Συνοψίζοντας, καταλήγουμε στο συμπέρασμα ότι η επιτυχημένη επίλυση πολυπαραγοντικών προβλημάτων, όπως αυτών που εμπεριέχονται στις κρατήσεις ειδικών χώρων ψυχαγωγίας, και η ανάπτυξη λογισμικού που τους αφορά, αποτελεί απαραίτητη, αν και περίπλοκη διαδικασία, καθώς πρέπει να βασίζεται στην υπάρχουσα γνώση σε συνδυασμό με προγραμματιστικές πρωτοβουλίες. Με αυτόν τον τρόπο, καθίσταται εφικτή η παράδοση ενός άρτιου λογισμικού που να διαχειρίζεται, αναπτύσσεται, διανέμεται και επεκτείνεται κατά το δυνατόν αποδοτικότερα.

Κεφάλαιο 12ο

Πηγές και Βιβλιογραφία

1. IBM Technology corporation. Topic IaaS vs. PaaS vs. SaaS. https://www.ibm.com/topics/iaas-paas-saas (τελευταία πρόσβαση 22. 2. 2023).
2. IOSR Journal of Computer Engineering (IOSR-JCE) 2014. Client-Server Model. e-ISSN: 2278-0661, p- ISSN: 2278-8727 Volume 16, Issue 1, Ver. IX (Feb. 2014), PP 67-71.. Εθνολογία (4): 107-126
3. “Computer reservation system” Wikipedia, Wikimedia Foundation, 29 July 2019, https://en.wikipedia.org/wiki/Computer_reservation_system (τελευταία πρόσβαση 22. 2. 2023).
4. AltexSoft - Technology & Solution Consulting Company. Central Reservation System for Hotels: CRS Functionality and Software Explained. https://www.altexsoft.com/blog/central-reservation-system-hotel/ (τελευταία πρόσβαση 22. 2. 2023).
5. Ably - Software company in London, England. Socket.IO vs. WebSocket: A comparison. An introduction to Socket.IO. Δημοσιεύτηκε στις 7. 9. 2022. https://ably.com/topic/socketio-vs-websocket#an-introduction-to-socket-io (τελευταία πρόσβαση 22. 2. 2023).
6. Gökhan Ayrancıoğlu. Medium - American online publishing platform. What is Server-Sent Events (SSE) and how to implement it?. Δημοσιεύτηκε στις 7. 2. 2022. https://medium.com/yemeksepeti-teknoloji/what-is-server-sent-events-sse-and-how-to-implement-it-904938bffd73 (τελευταία πρόσβαση 22. 2. 2023).
7. I. Sommerville. Software Engineering, 9th English Edition (2011).
8. Sommerville, I. (2006) Βασικές αρχές Τεχνολογίας Λογισμικού, 8η Αγγλική έκδοση, Μτφρ. Δ. Τσιλογιάννης. Αθήνα: Εκδόσεις Κλειδάριθμος.
9. Elliotte Rusty Harold, “Προγραμματισμός διαδικτυακών εφαρμογών με Java”, Επιστημονική επιμέλεια ελληνικής έκδοσης: Στέργιος Παπαδημητρίου, ΤΕΙ ΑΜΘ, 4η Αμερικανική έκδοση.

10. Sven Ziemer. An Architecture for Web Applications. Essay in DIF 8914 Distributed Information Systems. November 28, 2002

11. Anuupadhyay. Geeksforgeeks web platform. How to Design a Web Application – A Guideline on Software Architecture. Τελευταία αλλαγή στις 28.12. 2022.https://www.geeksforgeeks.org/how-to-design-a-web-application-a-guideline-on-software-architecture/ (τελευταία πρόσβαση 22. 2. 2023).

12. Web Application Architecture, (Δημοσίευση 13 Σεπτ, 2021), διαθέσιμο στη διεύθυνση:https://medium.com/geekculture/web-application-architecture-800d3ecd8019.

13. Viplove Prakash. Medium - American online publishing platform. Web Application Architecture. Δημοσιεύτηκε στις 13. 9. 2021. https://medium.com/geekculture/web-application-architecture-800d3ecd8019 (τελευταία πρόσβαση 22. 2. 2023).

14. Χάρης Μανιφάβας. Τμήμα Εφ. Πληροφορικής & Πολυμέσων, ΤΕΙ Κρήτης. Κατανεμημένα Συστήματα - Επικοινωνία - Client/Server, 2012-2013 (Ε)

15. Haroon Shakirat Oluwatosin, School of Computing University Utara Malaysia Kedah, Malaysia (IOSR Journal of Computer Engineering). Client-Server Model, (Δημοσίευση Φεβ 2014).

16. Interviewbit web platform. Client Server Model (Δημοσίευση 22 Απρ, 2022). https://www.interviewbit.com/blog/client-server-model/ (τελευταία πρόσβαση 22. 2. 2023).

17. Mozilla foundation. HTTP request methods, Specifications (Τελευταία αλλαγή στις 9. 9.2022). https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods#specifications (τελευταία πρόσβαση 22. 2. 2023).

18. Stuart Schechter, Murali Krishnan, Michael D. Smith. Computer Networks and ISDN Systems, Volume 30, Issues 1–7, April 1998, Pages 457-467. Using path profiles to predict HTTP requests. https://www.sciencedirect.com/science/article/abs/pii/S0169755298001068 (τελευταία πρόσβαση 22. 2. 2023).

19. Mozilla foundation. HTTP response status codes, Specifications (Τελευταία αλλαγή στις 9. 9.2022). https://www.tutorialspoint.com/http/http_status_codes.htm (τελευταία πρόσβαση 12. 2. 2023).

20. Tutorialspoint.com - Online Library. What are Web Services? https://www.tutorialspoint.com/webservices/what_are_web_services.htm (τελευταία πρόσβαση 12. 2. 2023).

21. Alyssa Walker, Guru99.com. RESTful Web Services Tutorial: What is REST API with Example (Τελευταία αλλαγή στις 24. .12.2022). https://www.guru99.com/restful-web-services.html (τελευταία πρόσβαση 12. 2. 2023).

22. W3Schools.com - Online Web Tutorials. HTML Introduction. https://www.w3schools.com/html/html_intro.asp (τελευταία πρόσβαση 12. 2. 2023).

23. World Wide Web Consortium (W3C). HTML & CSS, What is CSS?. https://www.w3.org/standards/webdesign/htmlcss#whatcss (τελευταία πρόσβαση 12. 2. 2023).

24. Bootstrap open-source framework community. Get started with Bootstrap. https://getbootstrap.com/docs/5.3/getting-started/introduction/ (τελευταία πρόσβαση 12. 2. 2023).

25. Official Javascript web page. https://www.javascript.com/ (τελευταία πρόσβαση 12. 2. 2023).

26. Official ReactJS web page. https://reactjs.org/ (τελευταία πρόσβαση 12. 2. 2023).

27. Official Jest web page. https://jestjs.io/ (τελευταία πρόσβαση 12. 2. 2023).

28. Official Testing library web page. https://testing-library.com/ (τελευταία πρόσβαση 12. 2. 2023).

29. Official JSON web page. https://www.json.org/json-en.html (τελευταία πρόσβαση 12. 2. 2023).

30. Official Golang web page. https://go.dev/ (τελευταία πρόσβαση 12. 2. 2023).

31. Julien Etienne. Medium - American online publishing platform. Why Go: The benefits of Golang, (Δημοσίευση 18 Απριλίου, 2022) https://medium.com/@julienetienne/why-go-the-benefits-of-golang-6c39ea6cff7e (τελευταία πρόσβαση 12. 2. 2023).

32. Samuel Olusola, LogRocket Boston. Server-sent events vs. WebSockets, (Δημοσίευση 30 Μαρτίου, 2022, Samuel Olusola). https://blog.logrocket.com/server-sent-events-vs-websockets/ (τελευταία πρόσβαση 12. 2. 2023).

33. Official PostgreSQL web page. https://www.postgresql.org/ (τελευταία πρόσβαση 12. 2. 2023).

34. Official JWT web page. https://jwt.io/ (τελευταία πρόσβαση 12. 2. 2023).

35. Atlassian - Software company. What is version control. https://www.atlassian.com/git/tutorials/what-is-version-control (τελευταία πρόσβαση 12. 2. 2023).

36. Γιάννης Τζίτζικας, Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών, Φθινόπωρο 2006-2007. Γενικές μεθοδολογίες ανάπτυξης λογισμικού, Διάλεξη 4, 12. 10 .2006.

37. Nginx - Software Company. What Is a Reverse Proxy Server? https://www.nginx.com/resources/glossary/reverse-proxy-server (τελευταία πρόσβαση 12. 2. 2023).

38. IBM Technology corporation. What is Infrastructure as Code (IaC)? https://www.ibm.com/topics/infrastructure-as-code (τελευταία πρόσβαση 22. 2. 2023).

39. Bruce Bigirwenkya. Medium - American online publishing platform. Gopls Language Server setup for Go Projects. Δημοσιεύτηκε στις 13. 12. 2021. https://medium.com/the-andela-way/gopls-language-server-setup-for-go-projects-3ee79dcac123 (τελευταία πρόσβαση 22. 2. 2023).

40. w3sDesign.com. "The Dependency Injection design pattern - Problem, Solution, and Applicability". (τελευταία πρόσβαση 22. 2. 2023).

41. Nicola Jones, CC BY-ND 2.0, How to Conduct a Cognitive Walkthrough, Δημοσιεύτηκε στις 16. 03. 2022.

https://www.interaction-design.org/literature/article/how-to-conduct-a-cognitive-walkthrough

Παράρτημα - Κώδικας

Κώδικας 5.1 Δημιουργία table main.gme σε PSQL

```
CREATE TABLE main.gme (
    gme_id integer NOT NULL,
    status text DEFAULT 'inactive'::text NOT NULL,
    img_url text NOT NULL,
    map_url text NOT NULL,
    plrs text NOT NULL,
    age_rng text NOT NULL,
    nme text NOT NULL,
    descr text NOT NULL,
    addr text NOT NULL,
    dur integer NOT NULL
);
```

Κώδικας 5.2 Δημιουργία table main.gme_lcl σε PSQL

```
CREATE TABLE main.gme_lcl (
    gme_lcl_id integer NOT NULL,
    gme_id integer NOT NULL,
    lcl text NOT NULL,
    nme text NOT NULL,
    descr text NOT NULL,
    addr text NOT NULL
);
```

Κώδικας 5.3 Δημιουργία table main.schdl σε PSQL

```
CREATE TABLE main.schdl (
    schdl_id integer NOT NULL,
    gme_id integer NOT NULL,
    schedule jsonb DEFAULT '{}'::jsonb NOT NULL,
    active_by date
);
```

Κώδικας 5.4 Δημιουργία table main.bkng σε PSQL

```
CREATE TABLE main.bkng (
    bkng_id integer NOT NULL,
    status text DEFAULT 'booked'::text NOT NULL,
    dte timestamp without time zone NOT NULL,
    gme_id integer NOT NULL,
```

```
    nme text NOT NULL,
    mob_num text NOT NULL,
    email_addr text NOT NULL,
    tmstmp timestamp without time zone DEFAULT CURRENT_TIMESTAMP NOT NULL,
    canceled_at timestamp without time zone,
    notes text,
    bked_by integer,
    lst_edted_by integer
);
```

Κώδικας 5.5 Δημιουργία table main.usr σε PSQL

```
CREATE TABLE main.usr (
    usr_id integer NOT NULL,
    usrnme text NOT NULL,
    psswrd text NOT NULL,
    salt text NOT NULL
);
```

Κώδικας 6.1 Υλοποίηση HTTP handler method GetGame σε Go

```
func (g *GameHandler) GetGame(c echo.Context) error {
 var gm model.Game
 req := new(getGameReq)
 if err := req.bind(c, &gm); err != nil {
   return logger.EchoHTTPError(http.StatusUnprocessableEntity, err.Error())
 }
 game, err := g.store.SelectGameByID(c.Request().Context(), gm.GameID)
 if err != nil {
   return logger.EchoHTTPError(http.StatusInternalServerError, err.Error())
 }
 if game == nil {
   return logger.EchoHTTPError(http.StatusNotFound, "game not found")
 }
 return c.JSON(http.StatusOK, game)
}
```

Κώδικας 6.2 Ορισμός του GameHandler struct και οι μέθοδοι του σε Go

```
type GameHandler struct {
    store          store.GameStore
    bkngStore      store.BookingStore
    schdlStore     store.ScheduleStore
    emailHandler   email.EmailHandler
    bookingChanged chan struct{}
}

func (*GameHandler).BookGame(c echo.Context) error
func (*GameHandler).CreateGame(c echo.Context) error
```

```
func (*GameHandler).DispatchBookingUpdate()
func (*GameHandler).EditBooking(c echo.Context) error
func (*GameHandler).EditGame(c echo.Context) error
func (*GameHandler).GetBooking(c echo.Context) error
func (*GameHandler).GetBookings(c echo.Context) error
func (*GameHandler).GetCalendar(c echo.Context) error
func (*GameHandler).GetCalendarSSE(c echo.Context) error
func (*GameHandler).GetGame(c echo.Context) error
func (*GameHandler).GetGames(c echo.Context) error
func (GameHandler).GetSchedule(c echo.Context) error
func (GameHandler).PutNewSchedule(c echo.Context) error
func (GameHandler).PutSchedule(c echo.Context) error
func (*GameHandler).Register(api *echo.Group)
func (*GameHandler).RemoveBooking(c echo.Context) error
```

Κώδικας 6.3 Υλοποίηση της συνάρτησης constructor NewGameHandler σε Go

```
func NewGameHandler(store store.GameStore, bkngStore store.BookingStore,
schdlStore store.ScheduleStore, emailHandler email.EmailHandler)
(gameHandler *GameHandler) {
 gameHandler = &GameHandler{
   store,
   bkngStore,
   schdlStore,
   emailHandler,
   make(chan struct{}),
 }
 return
}
```

Κώδικας 6.4 Ορισμός του struct bookGameReq και υλοποίηση της μεθόδου bind σε Go

```
type bookGameReq struct {
      Dte    string `json:"date"  validate:"required,datetime=2006-01-02
15:04"`
      GameID int    `param:"gid"`
      Name   string `json:"name"  validate:"required"`
      MobNum string `json:"mob_num"  validate:"required,e164"`
      Email  string `json:"email" validate:"required,email"`
      Notes *string `json:"notes"`
}

func (r *bookGameReq) bind(c echo.Context, b *model.Booking, location
*time.Location) error {
      if err := c.Bind(r); err != nil {
            return err
      }
```

```
	if err := c.Validate(r); err != nil {
		return err
	}
	b.Dte = r.Dte
	b.GameID = r.GameID
	b.Name = r.Name
	b.MobNum = r.MobNum
	b.Email = r.Email
	b.Notes = r.Notes
	*location = common.GetHeaderLocation(c)

	return nil
}
```

Κώδικας 6.5 Υλοποίηση της συνάρτησης InsertGame σε Go

```
func (g *PSQLGameStore) InsertGame(ctx context.Context, game model.Game)
(err error) {
 query := `insert into gme (status, img_url, map_url, plrs, dur, age_rng,
nme, descr, addr)
 values ($1, $2, $3, $4, $5, $6, $7, $8, $9);`

 args := []interface{}{game.Status, game.ImgURL, game.MapURL, game.Plrs,
game.Dur, game.AgeRange, game.Name, game.Descr, game.Addr}

 _, err = g.db.ExecContext(ctx, query, args...)
 if err != nil {
   log.Logger().WithFields(logrus.Fields{"query": query, "args":
args}).Error(err)
 }

 return
}
```

Κώδικας 6.6 Ορισμός του interface GameStore σε Go

```
type GameStore interface {
	SelectGames(ctx context.Context, status string) (games []model.Game,
err error)
	SelectGameByID(ctx context.Context, gameID int) (game *model.Game,
err error)
	InsertGame(ctx context.Context, game model.Game) (err error)
	UpdateGame(ctx context.Context, game model.Game) (ok bool, err error)
}
```

Κώδικας 6.7 Ορισμός του PSQLGameStore struct και οι μέθοδοι του σε Go

```
type PSQLGameStore struct {
    db *sql.DB
```

```
}

func (*PSQLGameStore).InsertGame(ctx context.Context, game model.Game) (err
error)
func (*PSQLGameStore).SelectGameByID(ctx context.Context, gameID int) (game
*model.Game, err error)
func (*PSQLGameStore).SelectGames(ctx context.Context, status string)
(games []model.Game, err error)
func (*PSQLGameStore).UpdateGame(ctx context.Context, game model.Game) (ok
bool, err error)
```

Κώδικας 6.8 Ορισμός του interface EmailHandler σε Go

```
type EmailHandler interface {
      SendEmail(bkngID int, tmpltCode string, timezone string) (err error)
}
```

Κώδικας 6.9 Ορισμός του SMTPEmailHandler struct και η μέθοδος του σε Go

```
type SMTPEmailHandler struct {
    emailStore  store.EmailStore
    smtpDtls    config.SMTPDtls
    authEnabled bool
}

func (*SMTPEmailHandler).SendEmail(bkngID int, tmpltCode string,
timezone string) (err error)
```

Κώδικας 6.10 Υλοποίηση της συνάρτησης constructor NewSMTPEmailHandler σε Go

```
func NewSMTPEmailHandler(emailStore store.EmailStore, smtpDtls
config.SMTPDtls) (*SMTPEmailHandler, error) {
      email, err := generateEmail(smtpDtls.Username, model.EmailTmpl{Body:
"Verifing SMTP configuration", Subject: "Verifing SMTP configuration"},
model.EmailDtls{})
      if err != nil {
            return nil, err
      }

      err = send([]string{smtpDtls.Username}, email, smtpDtls, true)
      return &SMTPEmailHandler{emailStore, smtpDtls, true}, err
}
```

Κώδικας 6.11 Υλοποίηση της μεθόδου SendEmail σε Go

```
func (e *SMTPEmailHandler) SendEmail(bkngID int, tmpltCode string, timezone
string) (err error) {
      ctc := context.Background()
```

```
	ctx, cancel := context.WithTimeout(ctc, 2*time.Second)
	defer cancel()
	dtls, err := e.emailStore.SelectEmailDtls(ctx, bkngID, timezone)
	if err != nil {
		return
	}
	tmplt, err := e.emailStore.SelectEmailTemplate(ctx, tmpltCode)
	if err != nil {
		return
	}
	emailMsg, err := generateEmail(e.smtpDtls.Username, tmplt, dtls)
	if err != nil {
		return
	}
	to := []string{dtls.Email}
	err = send(to, emailMsg, e.smtpDtls, e.authEnabled)

	e.emailStore.InsertEmailReport(ctx, bkngID, tmpltCode, err)
	return
}
```

Κώδικας 6.12 Υλοποίηση του SSE HTTP handler GetCalendarSSESendEmail σε Go

```
func (g *GameHandler) GetCalendarSSE(c echo.Context) error {
	var (
		gm     model.Game
		offset int
		loc    time.Location
	)
	req := new(getCalendarReq)

	if err := req.bind(c, &gm, &offset, &loc); err != nil {
		return logger.EchoHTTPError(http.StatusUnprocessableEntity,
err.Error())
	}
	userID := c.Get(user.UserIDCtxKey)
	openBkngsOnly := userID == user.UserIDLimitedAccess
	clndr, echoErr := getCalendar(c.Request().Context(), g, gm.GameID,
loc, offset, openBkngsOnly)
	if echoErr != nil {
		return echoErr
	}
	c.Response().Header().Set("Content-Type", "text/event-stream")
	c.Response().Header().Set("Cache-Control", "no-cache")
	c.Response().Header().Set("Connection", "keep-alive")
	clndrBt, err := json.Marshal(clndr)
	if err != nil {
```

```
            return logger.EchoHTTPError(http.StatusInternalServerError,
err)
      }
      fmt.Fprintf(c.Response().Writer, "data: %v\n\n", string(clndrBt))
      c.Response().Flush()
      for {
            select {
            case <-g.bookingChanged:
                  clndr, echoErr := getCalendar(c.Request().Context(), g,
gm.GameID, loc, offset, openBkngsOnly)
                  if echoErr != nil {
                        return echoErr
                  }
                  clndrBt, err := json.Marshal(clndr)
                  if err != nil {
                        return
logger.EchoHTTPError(http.StatusInternalServerError, err)
                  }
                  fmt.Fprintf(c.Response().Writer, "data: %v\n\n",
string(clndrBt))
                  c.Response().Flush()
            case <-c.Request().Context().Done():
                  return nil
            }
      }
}
```

Κώδικας 6.13 Ορισμός του struct listener και της συνάρτησης constructor σε Go

```
type listener struct {
      *pq.Listener
      failed chan error
}

// NewListener creates a new listener for given PostgreSQL credentials.
func NewListener(connString string) *listener {
      l := &listener{failed: make(chan error, 2)}

      listener := pq.NewListener(
            connString,
            10*time.Second, time.Minute,
            l.LogListener)

      l.Listener = listener
```

```
      return l
}
```

Κώδικας 6.14 Υλοποίηση της μεθόδου ListenAndNotify σε Go

```
func (l *listener) ListenAndNotify(channelName string, callback func()) {
 if err := l.Listen(channelName); err != nil {
   l.Close()
   log.Fatal(err)
 }


 if err := l.listen(callback); err != nil {
   logger.EchoLogger.WithField("channel name", channelName).Error(err)
 }
}

// listen is the main loop of the listener to listen for notifications from
the database.
func (l *listener) listen(callback func()) error {
 for {
   select {
   case <-l.Notify:
     callback()
   case err := <-l.failed:
     return err
   case <-time.After(time.Minute):
     go l.Ping()
   }
 }
}
```

Κώδικας 6.15 Υλοποίηση της συνάρτησης notify_bkng_changes_func σε PostrgeSQL

```
CREATE OR REPLACE FUNCTION public.notify_bkng_changes_func()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
                BEGIN

                IF TG_OP='INSERT' or OLD.dte != NEW.dte or
NEW.canceled_at is not null THEN
                PERFORM pg_notify('bkng_changes_chan', 'changed');
                END IF;

                RETURN NEW;
                END;
```

```
                $function$
;
```

Κώδικας 6.16 Υλοποίηση της συνάρτησης generateAuthToken σε Go

```
// generateAuthToken return authentication JWT
func generateAuthToken(userID int, secret string) (signedToken string, err
error) {
 // Create the Claims
 claims := &AuthUserJWTClaims{
   &jwt.StandardClaims{
     ExpiresAt: time.Now().Add(time.Hour * 24 * 365).Unix(),
   },
   userID,
 }
 token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
 signedToken, err = token.SignedString([]byte(secret))

 return
}
```

Κώδικας 6.17 Χρήση του echo middleware JWTWithConfig και υλοποίηση της συνάρτησης ParseJWTFromCTX σε Go

```
api.Use(middleware.JWTWithConfig(middleware.JWTConfig{
      Claims:         &user.AuthUserJWTClaims{},
      SigningKey:     []byte(cnfg.JWTSecret),
      SuccessHandler: user.ParseJWTFromCTX,
      Skipper:        AuthWhitelistSkipper,
}))
// ParseJWTFromCTX parses jwt claims from context and set values to new
context keys

func ParseJWTFromCTX(c echo.Context) {
      userJWT := c.Get(middleware.DefaultJWTConfig.ContextKey)
      claims := userJWT.(*jwt.Token).Claims.(*AuthUserJWTClaims)
      userID := claims.UserID
      c.Set(UserIDCtxKey, userID)
}
```

Κώδικας 6.18 Υλοποίηση του package config σε Go

```
type Config struct {
	Port         string
	SQLString    string
	JWTSecret    string
	AllowOrigins []string
	SMTPDtls     SMTPDtls
}
type SMTPDtls struct {
	Username string
	Password string
	Host     string
	Port     string
}
// LoadEnv loads the .env file and returns config object
func LoadEnv() (config Config) {
	err := godotenv.Load(EnvFilePath)
	if err != nil {
		log.Logger().Warning(EnvFileWarning)
	}
	config.Port = os.Getenv(Port)
	if len(config.Port) == 0 {
		config.Port = DefaultPort
	}

	config.SQLString = os.Getenv(SQLString)
	config.JWTSecret = os.Getenv(JWTSecret)
	allowOrigins := os.Getenv(AllowOrigins)
	config.AllowOrigins = strings.Split(allowOrigins, ",")

	config.SMTPDtls.Username = os.Getenv(SMTPUsername)
	config.SMTPDtls.Password = os.Getenv(SMTPPassword)
	config.SMTPDtls.Host = os.Getenv(SMTPHost)
	config.SMTPDtls.Port = os.Getenv(SMTPPort)

	return
}
```

Κώδικας 6.19 Ανάθεση των μεθόδων HTTP handler σε αναγνωριστικά πόρων για τον διακομιστή HTTP echo σε Go.

```
gg := api.Group("/games")
gg.GET("", g.GetGames)
gg.GET("/:gid", g.GetGame)
gg.POST("", g.CreateGame)
gg.PUT("/:gid", g.EditGame)
```

Κώδικας 7.1 Χρήση του npm package Route σε JSX

```
<Routes>
   <Route path="/login" element={<Login />} />
   <Route
     path="*"
     element={(
       <PrivateRoute>
         <Layout>
           <React.Suspense fallback={loading()}>
           <Routes>
           <Route index path="*" element={<Navigate to="/games" />} />
             <Route path="/games" element={<Games />} />
             <Route path="/games/create" element={gameCreate} />
             <Route path="/games/:gid" element={gamePanel}>
               <Route path="booking/:bid" element={booking} />
               <Route path="booking/create" element={bookingCreate} />
                </Route>
             </Routes>

           </React.Suspense>
         </Layout>
       </PrivateRoute>
     )}
   />
 </Routes>
```

Κώδικας 7.2 Υλοποίηση της συνάρτησης agentSSE σε JS

```
const agentSSE = async (url, method, headers, body, setOk, setData, signal)
=> {
  await fetchEventSource(url, {
    method,
    headers,
    body: body ? JSON.stringify(body) : undefined,
    signal,
    onopen(response) {
      setOk(response.ok && response.status === 200);
      if (!response.ok) throw response;
    },
    onmessage(event) {
      const parsedData = JSON.parse(event.data);
      setData(parsedData);
    },
    onerror(err) {
      throw err;
    },
```

```
  });
};
```

Κώδικας 8.1 Nginx server block configuration

```
server {
    listen       80;
    listen  [::]:80;
    server_name  admin.${HOST};

    location / {
       proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://ui-admin:80;
    }

    location /api {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://api-admin:80;
    }

    location ~* /api/.*/sse {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://api-admin:80;

        proxy_set_header Connection '';
        proxy_http_version 1.1;
        chunked_transfer_encoding off;
        proxy_buffering off;
        proxy_cache off;
    }
}.
```

Κώδικας 8.2 Dockerfile για την δημιουργία του docker image ui-admin σε YAML

```
# pull official base image
FROM node:18.10.0-alpine as builder

# set working directory
WORKDIR /app

# install app dependencies
COPY package.json ./
COPY package-lock.json ./
RUN npm install
```

```
# add app
COPY . ./

# fixes JS out of memory error
ENV NODE_OPTIONS --max-old-space-size=3072

# build project
RUN npm run build

# Start a new stage from scratch
FROM nginx:1.23.2-alpine

COPY --from=builder /app/nginx.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Κώδικας 8.3 Environment variables για χρήση από το Docker compose σε YAML

```
ENVIRONMENT=1.0.0
HOST=bee.com
SQLSTRING=postgres://vxvtteck:saMOJZhj0sWXr5PWTfSDHf1RkZrcXnLLh@hella.db.el
ephantsql.com/vxvtteck
JWTSECRET=khaki

SMTPUSERNAME=ioulios.tsiko@gmail.com
SMTPPASSWORD=zanswxcpjdeddfhd
SMTPHOST=smtp.gmail.com
SMTPPORT=587
```

Κώδικας 8.4 Υλοποίηση του docker-compose file για την δημιουργία της υποδομής της εφαρμογής

```
version: "3"

services:
  nginx:
    image: registry.gitlab.com/khakibee/khakibee/nginx:${ENVIRONMENT}
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    environment:
      - HOST=${HOST}
```

```
    depends_on:
      - ui-admin
      - api-admin
      - ui-widget
      - api-widget
      - api-public

  ui-admin:
    image: registry.gitlab.com/khakibee/khakibee/ui-admin:${ENVIRONMENT}
    container_name: ui-admin

  api-admin:
    image: registry.gitlab.com/khakibee/khakibee/api-admin:${ENVIRONMENT}
    container_name: api-admin
    environment:
      - JWTSECRET=${JWTSECRET}
      - SQLSTRING=${SQLSTRING}
      - SMTPUSERNAME=${SMTPUSERNAME}
      - SMTPPASSWORD=${SMTPPASSWORD}
      - SMTPHOST=${SMTPHOST}
      - SMTPPORT=${SMTPPORT}

  ui-widget:
 image: registry.gitlab.com/khakibee/khakibee/ui-widget:${ENVIRONMENT}
    container_name: ui-widget

  api-widget:
image: registry.gitlab.com/khakibee/khakibee/api-widget:${ENVIRONMENT}
    container_name: api-widget
    environment:
      - SQLSTRING=${SQLSTRING}

  api-public:
    image: registry.gitlab.com/khakibee/khakibee/api-public:${ENVIRONMENT}
    container_name: api-public
    environment:
      - SQLSTRING=${SQLSTRING}
```

Κώδικας 9.1 Υλοποίηση σεναρίου δοκιμής για κώδικα JS

```
 it('should not save invalid form', async () => {
    renderNode(<BookingForm />);

    expect(getName()).toBeInvalid();
```

```
    expect(getEmail()).toBeInvalid();
    expect(getMobile()).toBeInvalid();

    expect(screen.getByTestId('bookingform')).toBeInvalid();
  });
```

Κώδικας 9.2 Υλοποίηση δεύτερου σεναρίου δοκιμής για κώδικα JS

```
it('should create a new booking', async () => {
    const { user } = renderNode(<BookingForm dte={bkng1.date}
onSave={trackOnSave} />);

    await user.type(getName(), bkng1.name);
    await user.type(getEmail(), bkng1.email);
    await user.type(getMobile(), bkng1.mob_num);

    expect(screen.getByTestId('bookingform')).toBeValid();
    await user.click(screen.getByText(/save/i));
    expect(trackOnSave).toBeCalledWith(bkng1);
  });
```

Κώδικας 9.3 Υλοποίηση βοηθητικών συναρτήσεων για την εκτέλεση δοκιμών σε κώδικα JS

```
function getDate() {
  return screen.getByLabelText(/date/i);
}
function getName() {
  return screen.getByLabelText(/full name/i);
}
function getEmail() {
  return screen.getByLabelText(/email/i);
}
function getMobile() {
  return screen.getByLabelText(/mobile number/i);
}
```

Κώδικας 9.4 Υλοποίηση του package abs_test σε Go

```
package abs_test

import (
      "testing"
      "path_to_pkg/abs"
)

func TestAbs(t *testing.T) {
    got := abs.Abs(-1)
    if got != 1 {
        t.Errorf("Abs(-1) = %d; want 1", got)
    }
}
```

Κώδικας 9.5 Υλοποίηση της συνάρτησης AssertHTTPStatus σε Go

```
func AssertHTTPStatus(t testing.TB, got, want int) {
  t.Helper()
  if got != want {
    t.Errorf("wanted http status %d but got http status %d", want, got)
  }
}
```

Κώδικας 9.6 Υλοποίηση της συνάρτησης NewGetRequest σε Go

```
func NewGetRequest() (c echo.Context, request *http.Request, response
*httptest.ResponseRecorder) {
  e := echo.New()
  e.Validator = &CustomValidator{validator: validator.New()}

  request, _ = http.NewRequest(http.MethodGet, "/", nil)
  request.Header.Set(HeaderTimezone, Timezone)
  response = httptest.NewRecorder()
  c = e.NewContext(request, response)

  return
}
```

Κώδικας 9.7 Ορισμός του struct DB και αρχικοποίηση των τιμων του πεδίου Schedule σε Go

```
type DB struct {
      Games     []model.Game
      Bookings  []model.Booking
      Users     []model.User
      Schedule  map[int][]*model.Schedule
      Calendar  map[int][]model.Event
      EmailTmpl map[string]model.EmailTmpl
      EmailDtls model.EmailDtls
}


Schedule: map[int][]*model.Schedule{
 1: [
      {
        SchdlID: 1,
        WkSchdl: model.WeekSchedule{
          Sunday:    []string{"14:00", "19:00"},
          Monday:    []string{"14:00", "19:00"},
          Tuesday:   []string{"14:00", "19:00"},
          Wednesday: []string{"14:00", "19:00"},
          Thursday:  []string{"14:00", "19:00"},
          Friday:    []string{"14:00", "19:00"},
          Saturday:  []string{"14:00", "19:00"},
        },
        Dur:      160,
        ActiveBy: "2022-09-20",
      }
   ]
}
```

Κώδικας 9.8 Ορισμός του struct MockGameStore και των μεθόδων του σε Go

```
type MockGameStore struct {
    Games        []model.Game
    Calendar     map[int][]model.Event
    InsertCalled int
    UpdateCalled int
    ReturnError  error
}
func (*MockGameStore).InsertGame(ctx context.Context, game model.Game) (err
```

```
error)
func (*MockGameStore).SelectGameByID(ctx context.Context, gameID int) (game
*model.Game, err error)
func (*MockGameStore).SelectGames(ctx context.Context, status string)
(games []model.Game, err error)
func (*MockGameStore).UpdateGame(ctx context.Context, game model.Game) (ok
bool, err error)
```

Κώδικας 9.9 Υλοποίηση της εικονικής μεθόδου InsertGame σε Go

```
func (s *MockGameStore) InsertGame(ctx context.Context, game model.Game)
(err error) {
  s.InsertCalled += 1
  if s.ReturnError != nil {
    return s.ReturnError
  }
  return nil
}
```

Κώδικας 9.10 Υλοποίηση της Testing function TestGetGame σε Go

```
func TestGetGame(t *testing.T) {
      gs = &store.MockGameStore{Games: testUtils.DB1.Games, Calendar:
testUtils.DB1.Calendar}
      bs = &store.MockBookingStore{Bookings: testUtils.DB1.Bookings}
      ss = &store.MockScheduleStore{Calendar: testUtils.DB1.Calendar,
Schedule: testUtils.DB1.Schedule}
      eh = email.NewMockEmailHandler()
      h = game.NewGameHandler(gs, bs, ss, eh)

      for _, tg := range testUtils.DB1.Games {
            t.Run(fmt.Sprintf("Get game %d successfully", tg.GameID),
func(t *testing.T) {
                  c, _, res := testUtils.NewGetRequest()
                  c.SetParamNames("gid")
                  c.SetParamValues(strconv.Itoa(tg.GameID))

                  h.GetGame(c)
                  testUtils.AssertHTTPStatus(t, res.Code, http.StatusOK)
                  var gotGame model.Game
                  json.NewDecoder(res.Body).Decode(&gotGame)

                  testUtils.AssertEqual(t, gotGame, tg)
```

```
		})
	}
	t.Run("Get game not found", func(t *testing.T) {
		c, _, _ := testUtils.NewGetRequest()
		c.SetParamNames("gid")
		c.SetParamValues("0")

		err := h.GetGame(c)
		he := err.(*echo.HTTPError)
		testUtils.AssertHTTPStatus(t, he.Code, http.StatusNotFound)
	})

	t.Run("Get game with failing select in store", func(t *testing.T) {
		c, _, _ := testUtils.NewGetRequest()
		c.SetParamNames("gid")
		c.SetParamValues(strconv.Itoa(testUtils.DB1.Games[0].GameID))

		gs.ReturnError = errStore
		err := h.GetGame(c)
		he := err.(*echo.HTTPError)
		testUtils.AssertHTTPStatus(t, he.Code,
http.StatusInternalServerError)
	})
}
```