Σύστημα Κρατήσεων Βραχυχρόνιας Μίσθωσης με χρήση Τεχνολογιών Μικροϋπηρεσιών και Εικονικοποίησης βασισμένης σε Περιέκτες

Νικόλαος Πατεράκης

Ιούλιος 2023

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Πανεπιστήμιο Δυτικής Αττικής

Επιβλέπων Καθηγητής: Βασίλειος Μάμαλης

Συνεπιβλέπων Καθηγητής: Απόστολος Αναγνωστόπουλος

Αθήνα, Ιούλιος 2023

Rental Booking Management System using Microservices and Container-based Virtualization

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Μάμαλης Βασίλειος	Καθηγητής	Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2023.07.27 13:36:47 +03'00'
2	Καντζάβελου Ιωάννα	Επίκουρη Καθηγήτρια	Ioanna Kantzavelou Digitally signed by Ioanna Kantzavelou Date: 2023.07.29 14:41:23 +03'00'
3	Καρκαζής Παναγιώτης	Αναπληρωτής Καθηγητής	Panagiotis Karkazis Digitally signed by Panagiotis Karkazis Date: 2023.07.31 18:01:19 +03'00'

Ημερομηνία εξέτασης 20/07/2023

Rental Booking Management System using Microservices and Container-based Virtualization

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Πατεράκης Νικόλαος του Αντωνίου, με αριθμό μητρώου 161084 φοιτητής του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστήμιο Δυτικής Αττικής δηλώνω ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολο τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για ανάκληση του πτυχίου μου»

Ο Δηλών

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία αναλύει την αρχιτεκτονική microservices και παρουσιάζει κατάλληλους τρόπους ανάπτυξης ενός συστήματος με αυτήν την αρχιτεκτονική. Η διπλωματική επικεντρώνεται τόσο στη θεωρητική όσο και στην πρακτική πλευρά της κατασκευής συστημάτων με την αρχιτεκτονική microservices.

Στο πλαίσιο αυτής της εργασίας, παρουσιάζεται μια εφαρμογή που έχει κατασκευαστεί με την αρχιτεκτονική microservices και γίνεται λεπτομερή ανάλυση στη λειτουργικότητα της, τη δομής της, των εργαλείων, και των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της.

Στο θεωρητικό μέρος επισημαίνονται βασικές πληροφορίες για το cloud computing, τα οφέλη της αρχιτεκτονικής microservices, η πρακτικές που χρησιμοποιούνται για την ανάπτυξη ενός συστήματος με αυτήν την αρχιτεκτονική, και θεωρία για τους περιέκτες.

Τέλος, η εργασία παρέχει μια ολοκληρωμένη μελέτη της αρχιτεκτονικής των microservices και των σχετικών πρακτικών, η έρευνα προσφέρει κατανόηση για την υλοποίηση και τη διαχείριση συστημάτων με αυτήν την αρχιτεκτονική.

Περιεχόμενα

1. Εισαγωγή

Η αρχιτεκτονική των microservices αποτελεί μία σύγχρονη προσέγγιση στην ανάπτυξη λογισμικού που έχει αποκτήσει μεγάλη δημοφιλία τα τελευταία χρόνια. Η αρχιτεκτονική αυτή αποτελείται από μικρά, ανεξάρτητα κομμάτια λογισμικού τα οποία ονομάζουμε services όπου επικοινωνούν μεταξύ τους. Κάθε service εκτελεί μία συγκεκριμένη λειτουργία ανεξάρτητη από τις υπόλοιπες λειτουργίες του συστήματος, αυτό παρέχει τη δυνατότητα στους οργανισμούς που αναπτύσσουν λογισμικό να χωρίζουν σε μικρές ανεξάρτητες ομάδες τους προγραμματιστές και κάθε ομάδα να δουλεύει πάνω σε μια συγκεκριμένη λειτουργικότητα του συνολικού συστήματος. Με αυτόν τον τρόπο επιτυγχάνεται μεγάλη ευελιξία στα συστήματα με microservices αρχιτεκτονική, αφού επιτρέπεται ανεξάρτητη ανάπτυξη, ανάλυση και επεκτασιμότητα των επιμέρους λειτουργιών του συστήματος. Με αυτόν τον τρόπο, τα συστήματα μπορούν να προσαρμοστούν γρήγορα σε νέες απαιτήσεις των χρηστών και να αντιμετωπίσουν προβλήματα στην λειτουργικότητα τους αποτελεσματικότερα.

Ωστόσο, η αρχιτεκτονική των microservices εισάγει και κάποιες προκλήσεις, όπως η ανάγκη για πιο περίπλοκες στρατηγικές επικοινωνίας μεταξύ των services, δυσκολότερη διερεύνηση προβλημάτων, πιο εξειδικευμένο προσωπικό για την ανάπτυξη των συστημάτων και η απαίτηση για ένα αποτελεσματικό σύστημα διαχείρισης των services οπού αποτελείται η εφαρμογή και παρακολούθησης αυτών.

Αντίθετα στη μονολιθική αρχιτεκτονική που χρησιμοποιείται ευρέος στον χώρο της ανάπτυξης λογισμικού, όλες οι λειτουργίες ενός συστήματος είναι συνδεδεμένες και αλληλεξαρτώμενες, γεγονός που μπορεί να οδηγήσει σε περιορισμένη ευελιξία και αυξημένη πολυπλοκότητα κατά την ανάπτυξη, την αναβάθμιση και την κλιμάκωση της εφαρμογής. Επιπλέον, ένα σφάλμα σε ένα σημείο ενός συστήματος με αρχιτεκτονική μονόλιθου μπορεί να έχει εκτεταμένες επιπτώσεις στη συνολική λειτουργικότητα του συστήματος.

Στόχος της παρούσας διπλωματικής είναι να παρουσιάσει την αρχιτεκτονική των microservices και τους κατάλληλους τρόπους που αναπτύσσεις ένα σύστημα με αυτήν την αρχιτεκτονική. Η διπλωματική αποτελείται από δύο μέρη: το πρακτικό και το θεωρητικό μέρος. Στο πρακτικό μέρος, έχει αναπτυχθεί μια εφαρμογή με αρχιτεκτονική microservices. Στο θεωρητικό μέρος, έχει γραφτεί τεκμηρίωση που περιγράφει τη θεωρία του cloud computing, της αρχιτεκτονικής των microservices και των σχετικών πρακτικών που χρησιμοποιούνται για τη σωστή υλοποίηση της. Στο τέλος της τεκμηρίωσης, παρουσιάζεται η εφαρμογή που κατασκευάστηκε στο πρακτικό μέρος, περιγράφοντας τον τρόπο κατασκευής, τη δομή των αντικειμένων, τις τεχνολογίες που χρησιμοποιήθηκαν, και τον τρόπο αλληλεπίδρασης του χρήστη με αυτήν. Παρέχονται επίσης φωτογραφίες από τη διεπαφή χρήστη της εφαρμογής όπου εξηγείται η λειτουργικότητα της.

1.1 Τι είναι Υπολογιστική Νέφους

Η υπολογιστική νέφους σου παρέχει πρόσβαση σε πόρους μιας εφαρμογής, servers (φυσικούς και εικονικούς servers), αποθηκευτικό χώρο, εργαλεία ανάπτυξης εφαρμογών, υπηρεσίες διαδικτύου κ.α. όποτε τα χρειάζεσαι μέσω του Internet. Όλες αυτές οι υπηρεσίες φιλοξενούνται σε απομακρυσμένα data center όπου η διαχείριση τους πραγματοποιείται από έναν cloud services provider (η CSP). Ένας CSP διαθέτει αυτούς τους πόρους με μία μηνιαία συνδρομή ή με χρεώσεις βάση της χρήσης που έχεις κάνει σε μια υπηρεσία.

Ο όρος υπολογιστική νέφους επίσης αναφέρεται στη τεχνολογία που κάνει το cloud να λειτουργεί. Σε αυτή συμπεριλαμβάνονται μερικές μορφές εικονικής IT υποδομής-servers, λογισμικό λειτουργικών συστημάτων, δίκτυα, και άλλες υποδομές όπου είναι αφηρημένες, χρησιμοποιώντας ειδικό λογισμικό ώστε να μπορεί να διανεμηθεί και να χωριστεί ανεξάρτητα από τα όρια του φυσικού υλικού. Για παράδειγμα, ένας server μπορεί να χωριστεί σε πολλαπλούς εικονικούς servers.

Με τη χρήση της εικονικοποίησης (virtualization) οι πάροχοι υπηρεσιών υπολογιστικής νέφους είναι σε θέση να κάνουν μέγιστη χρήση των πόρων που διαθέτουν τα data center τους. Γι' αυτό το λόγο πολλές εταιρείες έχουν υιοθετήσει το μοντέλο διανομής μέσω υπολογιστικής νέφους για τις on-premises υποδομές τους.

Αν χρησιμοποιείς υπολογιστή ή κινητή συσκευή στο σπίτι ή στη δουλεία, είναι σίγουρο ότι χρησιμοποιείς κάποιο τύπο υπολογιστικής νέφους κάθε μέρα. Αυτός ο τύπος μπορεί να είναι μια εφαρμογή υπολογιστικής νέφους σαν το Google Gmail, μια υπηρεσία streaming σαν το Netflix, ή μια υπηρεσία αποθήκευσης αρχείων όπως το OneDrive. Σύμφωνα με μια πρόσφατη έρευνα, το 92% των οργανισμών χρησιμοποιεί υπηρεσίες υπολογιστικής νέφους, και οι περισσότεροι από αυτούς σχεδιάζουν να χρησιμοποιήσουν υπηρεσίες υπολογιστικής νέφους περισσότερο στο μέλλον.

1.1.1 Οφέλη της Υπολογιστικής Νέφους

Η υπολογιστική νέφους είναι μια μεγάλη αλλαγή στον παραδοσιακό τρόπο σκέψεις που έχουν οι επιχειρήσεις για τους IT πόρους. Παρακάτω θα δούμε μερικού από τους πιο σημαντικούς λόγους που ένας οργανισμός κατευθύνεται προς τη χρήση της υπολογιστικής νέφους.

Κόστος

Η υπολογιστική νέφους ελαχιστοποιεί τα έξοδα μίας εταιρείας αφού δεν χρειάζεται να αγοράζει hardware, λογισμικό, να εγκαθιστά, να ρυθμίζει και να διαχειρίζεται λογισμικό.

Παγκόσμια επέκταση

Τα πλεονεκτήματα της υπολογιστικής νέφους περιλαμβάνουν την ικανότητα της ελαστικής επέκτασης. Αυτό σημαίνει ότι γίνεται σωστή διανομή του ποσοστού των πόρων. Για παράδειγμα, περισσότερη ή λιγότερη υπολογιστική δύναμη, αποθηκευτικός χώρος, εύρος ζώνης όπου χρειάζεται, και για συγκεκριμένη γεωγραφική τοποθεσία.

Απόδοση

Οι μεγαλύτερες υπηρεσίες υπολογιστικής νέφους τρέχουν σε ένα παγκόσμιο δίκτυο από ασφαλή data centers όπου αναβαθμίζονται σε τακτά χρονικά διαστήματα με της τελευταίας γενιάς hardware. Αυτό προσφέρει πολλά πλεονεκτήματα σε σχέση με ένα data center το οποίο κατέχει μια εταιρεία. Για παράδειγμα, μειώνει την καθυστέρηση δικτύου για τις εφαρμογές και εξοικονομεί χρήματα όταν χρειάζεται να γίνει κάποια επέκταση.

Security

Πολλοί πάροχοι υπηρεσιών υπολογιστικής νέφους προσφέρουν διάφορες πολιτικές , και τεχνολογίες όπου ενδυναμώνουν την ασφάλεια της εφαρμογής, βοηθούν στην προστασία των δεδομένων, και των υποδομών από ενδεχόμενες απειλές.

Ταχύτητα

Οι περισσότερες υπηρεσίες υπολογιστικής νέφους παρέχονται όποτε τις χρειάζεσαι. Συνεπώς τεράστια ποσοστά υπολογιστικών πόρων μπορούν να προσφερθούν σε λίγα λεπτά. Αυτό δίνει μεγάλη ευελιξία στις επιχειρήσεις.

Παραγωγικότητα

Data centers τα οποία κατέχει μια εταιρεία απαιτούν εγκαταστάσεις hardware, επιδιορθώσεις λογισμικού και άλλες χρονοβόρες IT διαδικασίες διαχείρισης. Η υπολογιστική νέφους αφαιρεί πολλές από αυτές τις διαδικασίες. Συνεπώς, οι ομάδες IT μπορούν να ξοδέψουν το χρόνο τους σε σημαντικότερες εργασίες για την εκάστοτε εταιρεία.

Αξιοπιστία

Η υπολογιστική νέφους κάνει ευκολότερο και λιγότερο ακριβό το backup δεδομένων, την αποκατάσταση καταστροφών, και τη συνοχή της επιχείρησης επειδή τα δεδομένα μπορούν να αντιγραφούν σε πολλαπλές τοποθεσίες στο δίκτυο του cloud παρόχου.

1.1.2 Κατηγορίες Υπολογιστικής Νέφους

Δημόσιο Cloud

Δημόσιο cloud είναι ένας τύπος υπολογιστικής νέφους όπου ένας πάροχος υπηρεσιών cloud παρέχει υπολογιστικούς πόρους. Παράδειγμα τέτοιων πόρων είναι SaaS εφαρμογές, εικονικές μηχανές, υπολογιστικό υλικό, ολοκληρωμένες επιχειρησιακές υποδομές και πλατφόρμες ανάπτυξης όπου είναι διαθέσιμες στους χρήστες μέσω του Internet. Αυτοί οι πόροι μπορεί να είναι διαθέσιμοι χωρίς κάποιο κόστος, ή μπορεί να πουλιούνται σύμφωνα με κάποια συνδρομή ή με τη χρήση που κάνει ο χρήστης σε αυτούς τους πόρους.

Ο πάροχος δημοσίου cloud κατέχει, διαχειρίζεται, και είναι υπεύθυνος για τα data centers, το υλικό, και τις υποδομές που τρέχουν οι εργασίες των πελατών του, και τυπικά προσφέρει υψηλό εύρος ζώνης σύνδεσης στο δίκτυο ώστε να εξασφαλίσει υψηλές αποδόσεις και ταχύτατη πρόσβαση στις εφαρμογές και στα δεδομένα.

Το δημόσιο cloud είναι ένα περιβάλλον με πολλαπλούς ενοικιαστές cloud υπηρεσιών. Το data center ενός παρόχου υπολογιστικής νέφους μοιράζεται από όλους τους πελάτες του δημόσιου cloud. Στα κύρια δημόσια clouds-Amazon Web Services(AWS), Google Cloud, IBM Cloud, Microsoft Azure και Oracle Cloud-αυτοί οι πελάτες μπορεί να είναι εκατομμύρια.

Πολλές εταιρίες μεταφέρουν κομμάτια της υπολογιστικής τους υποδομής στο δημόσιο cloud επειδή οι υπηρεσίες δημοσίου cloud είναι ελαστικές, επεκτάσιμες, και μπορούν εύκολα να γίνουν τροποποιήσεις ώστε να μπορούν να ανταπεξέλθουν στις απαιτήσεις για αλλαγές που θα χρειάζονται οι εργασίες ενός πελάτη. Άλλοι πελάτες προσελκύονται επειδή το δημόσιο cloud υπόσχεται μεγαλύτερη αποτελεσματικότητα και λιγότερη σπατάλη πόρων αφού οι πελάτες πληρώνουν ότι χρησιμοποιούν. Τέλος

υπάρχουν και οι πελάτες οι όποιοι ψάχνουν το πως θα μειώσουν το κόστος σε υλικό ή σε on-premises υποδομές.

Ιδιωτικό Cloud

Το ιδιωτικό cloud είναι ένα cloud περιβάλλον όπου όλη η cloud υποδομή και οι υπολογιστικοί πόροι είναι αφιερωμένοι και προσβάσιμοι μόνο σε έναν πελάτη. Το ιδιωτικό cloud προσφέρει πολλά από τα πλεονεκτήματα της υπολογιστικής νέφους όπως ελαστικότητα, επεκτασιμότητα, και ευκολία στην διανομή υπηρεσιών με τον έλεγχο πρόσβασης, ασφάλεια, και την προσαρμοστικότητα των πόρων μιας on-premises υποδομής.

Ένα ιδιωτικό cloud συνήθως φιλοξενείται on-premises στο data center του πελάτη. Ωστόσο ένα ιδιωτικό cloud μπορεί επίσης να φιλοξενηθεί σε μία ανεξάρτητη υποδομή ενός παρόχου cloud ή μπορεί να φτιαχτεί σε μια ενοικιαζόμενη υποδομή σε ένα εξωτερικό data center.

Πολλές εταιρίες επιλέγουν το ιδιωτικό cloud σε αντίθεση με το δημόσιο cloud επειδή το ιδιωτικό cloud είναι ένας ευκολότερος τρόπος ή και ο μοναδικός για να πετύχουν κανονισμούς συμμόρφωσης που απαιτούνται. Άλλοι επιλέγουν το ιδιωτικό cloud επειδή οι εργασίες τους έχουν να κάνουν με εμπιστευτικά έγγραφα, πνευματική ιδιοκτησία, ιατρικά ιστορικά, οικονομικά δεδομένα, ή άλλα ευαίσθητα δεδομένα.

Hybrid Cloud

Το hybrid cloud είναι ένας συνδυασμός δημόσιου με ιδιωτικού cloud περιβάλλοντος. Πιο συγκεκριμένα και ιδανικά ένα hybrid cloud συνδέει τις υπηρεσίες του ιδιωτικού cloud ενός οργανισμού με το δημόσιο cloud σε μια μοναδική ευέλικτη υποδομή για να τρέχει τις εφαρμογές και τις εργασίες που χρειάζεται ένας οργανισμός.

Ο στόχος του hybrid cloud είναι να εγκαθιδρύσει μια μίξη πόρων του δημόσιου και του ιδιωτικού cloud με ένα επίπεδο ενορχήστρωσης ανάμεσά τους. Αυτό θα δώσει σε έναν οργανισμό την ευελιξία να επιλέξει το βέλτιστο cloud για κάθε εφαρμογή ή εργασία που έχει και να μπορεί να μεταφέρει ελεύθερα τις εργασίες του ανάμεσα στα δύο clouds όταν ο περιστάσεις αλλάζουν. Αυτό επιτρέπει σε έναν οργανισμό να πετύχει τους τεχνικούς και επιχειρησιακούς του στόχους περισσότερο αποτελεσματικά και με λιγότερο κόστος από το να χρησιμοποιούσε μόνο το ιδιωτικό ή το δημόσιο cloud.

Multicloud και Hybrid Multicloud

Multicloud χαρακτηρίζεται η χρήση δύο ή περισσοτέρων clouds από δύο ή περισσότερους διαφορετικούς cloud παρόχους. Το να έχεις ένα multicloud περιβάλλον είναι τόσο απλό όσο να χρησιμοποιείς ένα email SaaS από έναν πωλητή και ένα πρόγραμμα επεξεργασίας εικόνας SaaS από έναν άλλο. Ωστόσο όταν οι επιχειρήσεις μιλούν για το multicloud εννοούν το να χρησιμοποιείς πολλαπλές cloud υπηρεσίες όπως SaaS, PaaS, και IaaS υπηρεσίες από δύο ή περισσότερους παρόχους δημοσίου cloud.

Hybrid multicloud είναι η χρήση δύο ή περισσοτέρων δημόσιων cloud μαζί με ένα ιδιωτικό cloud περιβάλλον.

Οι οργανισμοί επιλέγουν το mutlicloud για να αποφύγουν να εγκλωβιστούν σε έναν πωλητή. Με αυτόν τον τρόπο έχουν περισσότερες επιλογές σε υπηρεσίες και μεγαλύτερη πρόσβαση σε καινοτομία. Παρ' όλ' αυτά όσο περισσότερα cloud χρησιμοποιείς όπου καθένα από αυτά έχει το δικό του set από εργαλεία διαχείρισης, το δικό του ρυθμό μετάδοσης δεδομένων, και τα δικά του πρωτόκολλα ασφάλειας τόσο πιο δύσκολο γίνεται να διαχειριστείς το περιβάλλον σου. Οι Multicloud πλατφόρμες διαχείρισης παρέχουν

ορατότητα πολλαπλών cloud παρόχων μέσα από έναν κεντρικό πίνακα, όπου οι ομάδες ανάπτυξης μπορούν να δουν τα project τους.

1.1.3 Είδη Υπηρεσιών Υπολογιστικής Νέφους

Οι τρεις πιο γνωστοί τύποι υπηρεσιών υπολογιστικής νέφους είναι η IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service), και SaaS (Software-as-a-Service). Είναι σύνηθες ένας οργανισμός να χρησιμοποιεί και τις τρεις υπηρεσίες. Ωστόσο πολλές φορές υπάρχει σύγχυση στο τι περιλαμβάνει η κάθε μία από αυτές.

SaaS (Software-as-a-Service)

SaaS γνωστό και ως λογισμικό βασισμένο στο cloud ή cloud εφαρμογή είναι λογισμικό εφαρμογής όπου φιλοξενείται στο cloud και έχεις πρόσβαση σε αυτό μέσα από κάποιον φυλλομετρητή, έναν εξειδικευμένο desktop client, ή ένα API που είναι ενσωματωμένο στο λειτουργικό σύστημα του υπολογιστή σου ή της κινητής συσκευής σου. Στις περισσότερες περιπτώσεις η χρήστες SaaS πληρώνουν μια μηνιαία ή ετήσια συνδρομή. Μερική μπορεί να προσφέρουν πακέτα με χρεώσεις που έχουν να κάνουν με το πόσο χρησιμοποιείς τη συγκεκριμένη υπηρεσία.

Εκτός από τα χρήματα που γλυτώνεις, τα κέρδη της επεκτασιμότητας που έχεις στο cloud το SaaS προσφέρει τα ακόλουθα:

- Αυτόματες αναβαθμίσεις: Με το SaaS όταν γίνεται μια αναβάθμιση στην εφαρμογή κερδίζεις απευθείας τα οφέλη της, χωρίς να χρειάζεται να πρέπει να κάνεις εσύ κάποια διαδικασία ώστε να λάβεις αυτήν την αναβάθμιση.
- Προστασία από απώλειες δεδομένων: Λόγω του ότι η εφαρμογή βρίσκετε στο cloud δεν θα χάσεις τα δεδομένα σου αν η συσκευή σου χαλάσει.

Το SaaS είναι η πρώτη επιλογή για πολλές εμπορικές εφαρμογές σήμερα. Υπάρχουν χιλιάδες λύσεις SaaS διαθέσιμες όπου χρησιμοποιούμε καθημερινά.

PaaS (Platform-as-a-Service)

Το PaaS προσφέρει στους προγραμματιστές μια πλατφόρμα, υλικό, λογισμικό για την εκτέλεση των εφαρμογών τους, υποδομές, και προγραμματιστικά εργαλεία που τους βοηθούν να αναπτύξουν και να διαχειριστούν της εφαρμογές τους χωρίς κόστος, πολυπλοκότητα, που θα είχε μια on-premises πλατφόρμα.

Με το PaaS ο cloud πάροχος προσφέρει τα πάντα servers, δίκτυα, αποθηκευτικό χώρο, λειτουργικά συστήματα, βάσεις δεδομένων, middleware στο δικό του data center. Οι προγραμματιστές απλός επιλέγουν από ένα μενού τι ακριβώς χρειάζονται για την εφαρμογή τους.

Σήμερα, το PaaS είναι χτισμένο γύρο από τους containers όπου εικονικοποιούν το λειτουργικό σύστημα ώστε να επιτρέψουν στους προγραμματιστές να πακετάρουν την εφαρμογή τους με τις απολύτως απαραίτητες υπηρεσίες που χρειάζεται για να τρέξει σε οποιαδήποτε πλατφόρμα, χωρίς τροποποιήσεις η κάποιο middleware.

IaaS (Infrastructure-as-a-Service)

Το IaaS παρέχει θεμελιώδης υπολογιστικούς πόρους, φυσικούς και εικονικούς servers, δίκτυα, και αποθηκευτικό χώρο στο internet. Το IaaS επιτρέπει στους τελικούς χρήστες να επεκτείνουν ή να συρρικνώσουν τους πόρους που χρησιμοποιούν όποτε αυτό χρειάζεται μειώνοντας αχρείαστα κόστη η

υποδομές που χρησιμοποιούνται μόνο κάποιες συγκεκριμένες περιόδους που η εφαρμογή τους μπορεί να έχει αυξημένη κινητικότητα.

Σε αντίθεση με το SaaS και PaaS το IaaS παρέχει στο χρήστη το κατώτατο επίπεδο χειρισμού υπολογιστικών πόρων στο cloud.

Το IaaS είναι το πιο δημοφιλή μοντέλο υπολογιστικής νέφους από όταν αναδύθηκε το 2010. Ενώ παραμένει ένα μοντέλο υπολογιστικής νέφους για πολλά είδη εργασιών, η χρήση των SaaS και PaaS αυξάνεται πολύ πιο γρήγορα.

Serverless computing

Serverless computing είναι ένα μοντέλο υπολογιστικής νέφους το οποίο ξεφορτώνεται όλη τη διαχείριση της υποδομής, την επεκτασιμότητα, τις επιδιορθώσεις στον cloud πάροχο, ελευθερώνοντας τους προγραμματιστές ώστε να δίνουν όλο τους το χρόνο και την προσπάθεια τους στον κώδικα και στο τι χρειάζεται η επιχείρηση για την εφαρμογή που θέλουν να φτιάξουν.

Στην υπηρεσία Serverless ο κώδικας της εφαρμογής εκτελείτε μόνο όταν λάβει κάποιο αίτημα και μπορεί να επεκτείνει ή συρρικνώσει την υποστηριζόμενη υποδομή αυτόματα ανάλογα με τον αριθμό τον αιτημάτων που λαμβάνονται. Σε αυτήν την υπηρεσία η πελάτες πληρώνουν μόνο για τους πόρους που χρησιμοποιούνται όταν η εφαρμογή τρέχει.

1.1.4 Εφαρμογές της Υπολογιστικής Νέφους

Καθημερινά όταν χρησιμοποιούμε υπηρεσίες στο διαδίκτυο χωρίς να το καταλάβουμε κάνουμε χρήση της υπολογιστικής νέφους. Αν χρησιμοποιούμε διαδικτυακές υπηρεσίες για να στέλνουμε email, να επεξεργαζόμαστε έγγραφα, να δούμε ταινίες, να ακούσουμε μουσική, να παίξουμε παιχνίδια, ή να αποθηκεύσουμε φωτογραφίες ή άλλα αρχεία, είναι πολύ πιθανό ότι κάνουμε χρήση της υπολογιστικής νέφους. Οι πρώτες υπηρεσίες υπολογιστικής νέφους δημιουργήθηκαν το πολύ πριν μια δεκαετία, παρόλα αυτά πληθώρα οργανισμών από μικρές εταιρίες έως πολυεθνικές χρησιμοποιούν αυτήν την τεχνολογία για διαφόρους λόγους.

Μερικά παραδείγματα χρήσης της υπολογιστικής νέφους:

Δημιουργία cloud-native εφαρμογών

Γρήγορη κατασκευή και επεκτασιμότητα εφαρμογών διαδικτύου, κινητών, και API. Εκμετάλλευση των cloud-native τεχνολογιών και προσεγγίσεων, όπως οι containers, το Kubernetes, την αρχιτεκτονική microservices, την API-driven επικοινωνία, και το DevOps.

Αποθήκευση, αντίγραφα ασφαλείας, και ανάκτηση δεδομένων

Προστασία δεδομένων μεταφέροντας τα στο διαδίκτυο.

Παραγωγή ήχου και video

Σύνδεση με το κοινό σου από οπουδήποτε, και από οποιαδήποτε συσκευή.

Ανάλυση δεδομένων

Αποθήκευση δεδομένων στο νέφος και έπειτα χρησιμοποίηση υπηρεσιών όπως μηχανική μάθηση και τεχνητή νοημοσύνη για την αποκάλυψη γνώσης για καλύτερες αποφάσεις πάνω στα δεδομένα.

Τεστ και κατασκευή εφαρμογών

Μείωση του κόστους και του χρόνου ανάπτυξης της εφαρμογής χρησιμοποιώντας την υποδομή της υπολογιστικής νέφους.

Παράδοση λογισμικού όποτε αυτό απαιτείται

Γνωστό και ως software as a service (SaaS), η παράδοση λογισμικού όποτε αυτό απαιτείται προσφέρει στους πελάτες σου τις τελευταίες εκδώσεις και ενημερώσεις λογισμικού οποιαδήποτε ώρα τις χρειάζονται.

1.2 Τι είναι DevOps

Η λέξη DevOps είναι ένας συνδυασμός των όρων development και operations, αυτό σημαίνει ότι εκπροσωπεί μια προσέγγιση συνεργασίας στις εργασίες που εκτελούνται από τις ομάδες ανάπτυξης εφαρμογών και IT λειτουργιών μιας εταιρίας.

Η Ευρύτερη σημασία του όρου DevOps είναι η φιλοσοφία όπου προωθεί την καλύτερη επικοινωνία και συνεργασία αυτών των ομάδων σε έναν οργανισμό. Σε μια πιο στενή ερμηνεία ο όρος DevOps περιγράφει την υιοθέτηση της επαναλαμβανόμενης ανάπτυξης κώδικα, την αυτοματοποίηση, και την συντήρηση.

Ενώ το DevOps δεν είναι τεχνολογία, ένα περιβάλλον DevOps εφαρμόζει κάποιες κοινές μεθοδολογίες. Μερικές από αυτές είναι

- Continuous integration και continuous delivery ή continuous deployment (CI/CD) εργαλεία, με έμφαση στο αυτοματοποίηση των εργασιών.
- Εργαλεία και συστήματα όπου υποστηρίζουν την υιοθέτηση του DevOps, όπως πραγματικού χρόνου παρακολούθηση, διαχείριση ρυθμίσεων και πλατφόρμες συνεργασίας.
- Υπολογιστική νέφους, microservices και containers

Η DevOps προσέγγιση είναι μια από τις πολλές τεχνικές που το IT προσωπικό χρησιμοποιεί για να επιτευχθούν οι ανάγκες μιας επιχείρησης. Το DevOps μπορεί να συνυπάρξει με τις Agile μεθοδολογίες.

1.3 Τι είναι μια Cloud Native Εφαρμογή

Το cloud native αναφέρεται λιγότερο στο πού κατοικεί ή εφαρμογή και περισσότερο στο πως έχει κατασκευαστεί και διανεμηθεί.

- Μια cloud native εφαρμογή αποτελείται από διακριτά, επαναχρησιμοποιήσιμα στοιχεία γνωστά και ως microservices όπου είναι σχεδιασμένα για να ενσωματώνονται σε οποιοδήποτε cloud περιβάλλον.
- Τα microservices ενεργούν σαν building blocks και συνήθως πακετάρονται σε containers.
- Τα microservices συνεργάζονται μεταξύ τους και όλα μαζί αποτελούν την εφαρμογή. Το καθένα μπορεί να επεκτείνεται ανεξάρτητα, και να βελτιώνεται συνεχώς.
- Η ευελιξία των microservices προσθέτουν ευκινησία και συνεχόμενη βελτίωση σε μια cloud native εφαρμογή.

2. Θεωρητικό Υπόβαθρο

2.1 Microservices

2.1.1 Αρχιτεκτονική Microservices

Η Microservices αρχιτεκτονική είναι μια cloud native αρχιτεκτονική όπου μια εφαρμογή αποτελείται από πολλά ανεξάρτητα χαλαρά συνδεδεμένα μεταξύ τους services. Κάθε service είναι αυτοδύναμο και πρέπει να υλοποιεί μια συγκεκριμένη δουλεία. Παράδειγμα σε μια επιχείρηση που πουλάει παπούτσια, ένα service είναι υπεύθυνο για τις παραγγελίες των πελατών.

- Κάθε service μπορεί να υλοποιηθεί με διαφορετικές τεχνολογίες και από διαφορετικές ομάδες υλοποίησης.
- Μία ομάδα υλοποίησης μπορεί να κάνει ενημερώσεις σε ένα υπάρχον service χωρίς να κάνει ανακατασκευή ολόκληρης της εφαρμογής.
- Η επικοινωνία μεταξύ των services επιτυγχάνεται με τεχνολογίες όπως REST API, event streaming, και message brokers.
- Το κάθε service είναι υπεύθυνο για τη διατήρηση των δικών του δεδομένων.
- Τα services είναι οργανωμένα με βάση τις απαιτήσεις της εκάστοτε επιχείρησης.

Η διαφορά μεταξύ microservices αρχιτεκτονικής με μια αρχιτεκτονική μονόλιθου είναι ότι τα microservices συνθέτουν μια εφαρμογή από πολλά χαλαρά συνδεδεμένα μεταξύ τους services σε αντίθεση με την αρχιτεκτονική μονόλιθου όπου αποτελείται από μια στενά συνδεδεμένη εφαρμογή.

2.1.2 Οφέλη

Ευκινησία

Επειδή τα microservices αναπτύσσονται ανεξάρτητα το ένα από το άλλο, είναι πιο εύκολη η διαχείριση των σφαλμάτων και η κυκλοφορία νέων χαρακτηριστικών της εφαρμογής. Μπορείς να ενημερώσεις ένα service χωρίς να ξανά διανέμεις ολόκληρη την εφαρμογή. Σε πολλές παραδοσιακές εφαρμογές αν ένα σφάλμα προκύψει σε ένα σημείο της εφαρμογής μπορεί να μπλοκάρει ολόκληρη την διαδικασία διανομής της εφαρμογής.

Μικρές ομάδες επικεντρωμένες σε μια διαδικασία

Ένα service πρέπει να είναι τόσο μικρό ώστε να μπορεί να κατασκευαστεί, να τεσταριστεί και διανεμηθεί από μια μόνο ομάδα. Ολιγομελής ομάδες επιτυγχάνουν μεγαλύτερη ευκινησία. Μεγάλες ομάδες τείνουν να είναι λιγότερο παραγωγικές επειδή η επικοινωνία είναι πιο αργή.

Μικρή βάση κώδικα

Σε μια αρχιτεκτονική μονόλιθου υπάρχει η τάση οι εξαρτήσεις στον κώδικα να είναι πιο μπλεγμένες. Όταν θέλεις να προσθέσεις ένα καινούριο χαρακτηριστικό στην εφαρμογή χρειάζεται να πειράξεις τον κώδικα σε πολλά σημεία. Όταν δεν μοιράζεσαι κώδικα ή δεδομένα οιπκ εξαρτήσεις μειώνονται και αυτό φέρνει σαν αποτέλεσμα ευκολότερη προσθήκη νέων χαρακτηριστικών.

Υλοποίηση πολλών services με διαφορετικές τεχνολογίες το κάθε ένα

Κάθε ομάδα μπορεί να διαλέξει την τεχνολογία που ταιριάζει καλύτερα στο service που τους έχει ανατεθεί να υλοποιήσουν.

Απομόνωση σφαλμάτων

Αν ένα service γίνει μη προσβάσιμο, δεν θα διακόψει τη λειτουργία ολόκληρης της εφαρμογής καθώς όλα τα υπόλοιπα services έχουν σχεδιαστεί να χειρίζονται κατάλληλα τα σφάλματα.

Επεκτασιμότητα

Τα services μπορούν να επεκταθούν ανεξάρτητα το ένα από το άλλο. Αυτό σου δίνει τη δυνατότητα να μπορείς να επεκτείνεις συγκεκριμένα υποσυστήματα που απαιτούν περισσότερους πόρους.

Απομόνωση δεδομένων

Είναι πολύ πιο εύκολο να εκτελείς ενημερώσεις στον σχεδιασμό μιας βάσης δεδομένων, επειδή μόνο ένα service επηρεάζεται. Σε μία εφαρμογή μονόλιθου οι ενημερώσεις στον σχεδιασμό μιας βάσης δεδομένων μπορεί να γίνουν πολύ δύσκολες, ο λόγος είναι γιατί διαφορετικά μέρη μιας εφαρμογής μπορούν να αγγίζουν τα ίδια δεδομένα.

2.1.3 Δυσκολίες

Η υλοποίηση της αρχιτεκτονικής microservices φέρνει και κάποια αρνητικά μαζί της. Στο παρόν κεφάλαιο θα δούμε ποιες δυσκολίες είναι αυτές.

Πολυπλοκότητα

Μια εφαρμογή με microservices αρχιτεκτονική αποτελείται από περισσότερα μέρη σε σύγκριση με μια εφαρμογή με αρχιτεκτονική μονόλιθου. Κάθε service είναι απλούστερο, αλλά ολόκληρο το σύστημα είναι πιο πολύπλοκο.

Ανάπτυξη και έλεγχος

Γράφοντας ένα μικρό service το οποίο βασίζεται σε άλλα services απαιτεί μια διαφορετική προσέγγιση σε σχέση με το να γράφεις μια εφαρμογή μονόλιθου. Τα υπάρχοντα εργαλεία δεν είναι πάντα σχεδιασμένα να δουλεύουν με τις εξαρτήσεις των services. Ο έλεγχος για την εύρεση σφαλμάτων όταν έχουμε να κάνουμε με πολλά services γίνεται δυσκολότερος.

Έλλειψη διοίκησης

Η αποκεντρωμένη προσέγγιση των microservices έχει πολλά πλεονεκτήματα, παρ’ όλα αυτά μπορεί να οδηγήσει σε προβλήματα κατά την ανάπτυξη μιας εφαρμογής. Μπορεί να καταλήξεις σε ένα σύστημα με πολλές διαφορετικές γλώσσες προγραμματισμού και frameworks οπού αυτό να καταστήσει δύσκολη τη συντήρηση του. Ίσως είναι χρήσιμο να θεσπίσεις κάποιους κανόνες χωρίς βέβαια να περιορίσεις την ευελιξία των ομάδων ανάπτυξης.

Συμφόρηση δικτύου

Η χρήση πολλών μικρών services έχει ως αποτέλεσμα περισσότερη εσωτερική επικοινωνία μεταξύ των services. Επιπλέον, αν η αλυσίδα των εξαρτήσεων των services γίνει πολύ μεγάλη (το service A καλεί το B, το οποίο καλεί το C…) η πρόσθετη καθυστέρηση θα δημιουργήσει νέο πρόβλημα. Για αυτούς τους λόγους ο σχεδιασμός των APIs πρέπει να γίνει πολύ προσεκτικά.

Ακεραιότητα δεδομένων

Λόγω του ότι κάθε service είναι υπεύθυνο για την διατήρηση των δικών του δεδομένων, αυτό μπορεί να προκαλέσει προβλήματα συνοχής στα δεδομένα.

Διαχείριση

Για μία πετυχημένη ανάπτυξη μια εφαρμογής με αρχιτεκτονική microservices απαιτείται μια ώριμη DevOps κουλτούρα.

Εκδόσεις

Όταν ενημερώνεις ένα service δεν πρέπει να καταστρέφονται οι εξαρτήσεις που έχει με τα υπόλοιπα services. Πολλαπλά services μπορούν να ενημερωθούν οποιαδήποτε στιγμή. Χωρίς προσεκτική σχεδίαση μπορεί να αντιμετωπίσεις προβλήματα συμβατότητας.

Ικανότητες ανθρωπίνου δυναμικού

Τα συστήματα με microservices αρχιτεκτονική είναι υψηλού επιπέδου κατανεμημένα συστήματα. Πρέπει προσεκτικά να αξιολογήσεις αν μια ομάδα έχει τις ικανότητες και την εμπειρία να πραγματοποιήσει με σωστό τρόπο μια υλοποίηση ενός τέτοιου συστήματος.

2.1.4 Επικοινωνία σε μια Αρχιτεκτονική Microservices

Σε μια εφαρμογή μονόλιθου τα διάφορα στοιχεία της εφαρμογής καλούν το ένα το άλλο χρησιμοποιώντας συναρτήσεις. Η μεγαλύτερη δυσκολία όταν αλλάζεις από μια αρχιτεκτονική μονόλιθου σε μια αρχιτεκτονική microservices είναι στην αλλαγή του μηχανισμού επικοινωνίας.

Μια εφαρμογή βασισμένη στην αρχιτεκτονική microservices είναι ένα κατανεμημένο σύστημα όπου τρέχει πολλαπλά services, συνήθως κατά μήκος πολλαπλών εξυπηρετητών. Κάθε service τυπικά είναι μια διεργασία. Συνεπώς, τα services πρέπει να επικοινωνούν μεταξύ τους με κάποιο πρωτόκολλο εσωτερικής επικοινωνίας όπως είναι τα HTTP, AMQP ή ένα δυαδικό πρωτόκολλο όπως είναι το TCP, αυτό εξαρτάται από τη φύση του κάθε service.

Ο πελάτης και τα services μπορούν να επικοινωνήσουν με πολλούς διαφορετικούς τρόπους. Κάθε τρόπος στοχεύει σε ένα διαφορετικό σενάριο και στόχο. Αρχικά αυτοί οι τύποι επικοινωνίας μπορούν να ταξινομηθούν σε δύο άξονες.

Ο πρώτος άξονας καθορίζει αν το πρωτόκολλο είναι σύγχρονο ή ασύγχρονο.

- **Σύγχρονο πρωτόκολλο:** Το HTTP είναι ένα σύγχρονο πρωτόκολλο. Οι πελάτες στέλνουν ένα ερώτημα και περιμένουν μια απάντηση από το service. Αυτό είναι ανεξάρτητο από την εκτέλεση του κώδικα του πελάτη που μπορεί να είναι σύγχρονη (το νήμα είναι μπλοκαρισμένο) ή ασύγχρονη (το νήμα δεν είναι μπλοκαρισμένο). Το σημαντικό σημείο εδώ είναι ότι το πρωτόκολλο (HTTP/HTTPS) είναι σύγχρονο και ο πελάτης μπορεί να συνεχίσει τις εργασίες του μόλις λάβει την απάντηση από τον HTTP εξυπηρετητή.
- **Ασύγχρονο πρωτόκολλο**: Αλλά πρωτόκολλα σαν το AMQP (ένα πρωτόκολλο όπου υποστηρίζεται από πολλά λειτουργικά συστήματα και περιβάλλοντα cloud) όπου χρησιμοποιεί ασύγχρονα μηνύματα. Ο κώδικας του πελάτη ή αυτός που στέλνει το μήνυμα δεν περιμένει για την απάντηση. Απλά στέλνει το μήνυμα όπως στέλνει ένα μήνυμα σε μια RabbitMQ ουρά ή οποιονδήποτε άλλο message broker.

Ο δεύτερος άξονας καθορίζει αν η επικοινωνία έχει έναν η πολλαπλούς δέκτες.

- **Ένας δέκτης**: Κάθε ερώτημα πρέπει να επεξεργάζεται από ακριβώς έναν δέκτη ή ένα service.
- **Πολλαπλοί δέκτες**: Κάθε ερώτημα πρέπει να επεξεργάζεται από κανέναν ή πολλούς δέκτες. Αυτός ο τύπος επικοινωνίας πρέπει να είναι ασύγχρονος. Ένα παράδειγμα είναι ο publish/subscribe μηχανισμός που χρησιμοποιείται από patterns σαν το Event-driven architecture.

Μια εφαρμογή με microservice αρχιτεκτονική συχνά χρησιμοποιεί ένα συνδυασμό αυτών των τύπων επικοινωνιών. Ο περισσότερο γνωστός τύπος είναι ένας δέκτης να επικοινωνεί με ένα σύγχρονο πρωτόκολλο σαν το HTTP/HTTPS όταν καλεί ένα Web API HTTP service. Η αρχιτεκτονική microservices χρησιμοποιεί πολύ συχνά messaging πρωτόκολλα για ασύγχρονη επικοινωνία μεταξύ των microservices.

2.1.5 Διαχείριση Δεδομένων

Η διαχείριση των δεδομένων είναι το δυσκολότερο μέρος στην υλοποίηση της αρχιτεκτονικής microservices. Σε μία εφαρμογή μονόλιθου μια σχεσιακή βάση δεδομένων διαχειρίζεται τη συνέπεια των δεδομένων. Από την άλλη πλευρά, σε μια αρχιτεκτονική microservices κάθε service έχει τη δική του βάση δεδομένων αν χρησιμοποιείς το pattern database per service. Με αυτό το pattern οι βάσεις δεδομένων είναι κατανεμημένες μέσα στην εφαρμογή. Κάθε service μπορεί να χρησιμοποιεί διαφορετική τεχνολογία βάσης δεδομένων π.χ. non-sql βάση δεδομένων. Αν και αυτό το είδος κατανεμημένης αρχιτεκτονικής έχει πολλά θετικά, όπως η επεκτασιμότητα, υψηλή διαθεσιμότητα, και η ευκινησία, στο θέμα της διαχείρισης των δεδομένων έχει κάποιες δυσκολίες στην υλοποίηση όπως η συλλογή των δεδομένων ανάμεσα στα services, και η συνέπεια των δεδομένων.

Για να επιλύσουμε το πρόβλημα της συνέπειας των δεδομένων μπορούμε να χρησιμοποιήσουμε το μοντέλο του Eventual consistency που χρησιμοποιείται στα κατανεμημένα συστήματα για να πετύχει υψηλή διαθεσιμότητα. Σε ένα eventual consistent σύστημα, οι ασυνέπειες επιτρέπονται για ένα μικρό χρονικό διάστημα μέχρι να λυθεί το πρόβλημα των κατανεμημένων δεδομένων.

Ένα γνωστό pattern που βοηθάει στην υλοποίηση του eventual consistency μοντέλου είναι το SAGA pattern. Το SAGA pattern είναι ένα ασύγχρονο pattern. Στο SAGA pattern οι κατανεμημένη συναλλαγή εκτελείται από ασύγχρονες τοπικές συναλλαγές σε όλα τα σχετιζόμενα services που χρειάζονται για να επιτευχθεί η συγκεκριμένη συναλλαγή. Κάθε service ενημερώνει τα δικά του δεδομένα σε μια τοπική συναλλαγή. Υπάρχουν δύο υλοποιήσεις του SAGA pattern.

Choreography-Based SAGA: Σε αυτή την υλοποίηση δεν υπάρχει κεντρικός συντονιστής. Κάθε service παράγει ένα γεγονός (event) και μετά την ολοκλήρωση μιας εργασίας κάθε service ακούει αυτό το γεγονός και με βάση αυτό κάνει μια ενέργεια. Αυτό το pattern απαιτεί μια event-driven αρχιτεκτονική. Αυτή η υλοποίηση του SAGA pattern δουλεύει καλά για μικρό αριθμό βημάτων συναλλαγών (2 με 4 βήματα). Όταν ο αριθμός των βημάτων μια συναλλαγής αυξηθεί, είναι δύσκολο να παρακολουθηθεί ποιο service ακούει σε ποιο γεγονός.

Orchestration-based SAGA: Ένα service συντονιστής είναι υπεύθυνο για την σειρά με την οποία θα πραγματοποιηθούν οι συναλλαγές σύμφωνα με την λογική της επιχείρησης. Αν μια λειτουργία αποτύχει, το service συντονιστής είναι υπεύθυνο για να ακυρώσει όλες τις προηγούμενες ενέργειες που έχουν πραγματοποιηθεί.

Γενικά πρέπει να αποφεύγεται όσο είναι εφικτό η κατανεμημένες συναλλαγές. Η εργασία με συναλλαγές που απαρτίζονται από πολλά services φέρνει σαν αποτέλεσμα πιο πολύπλοκα προβλήματα.

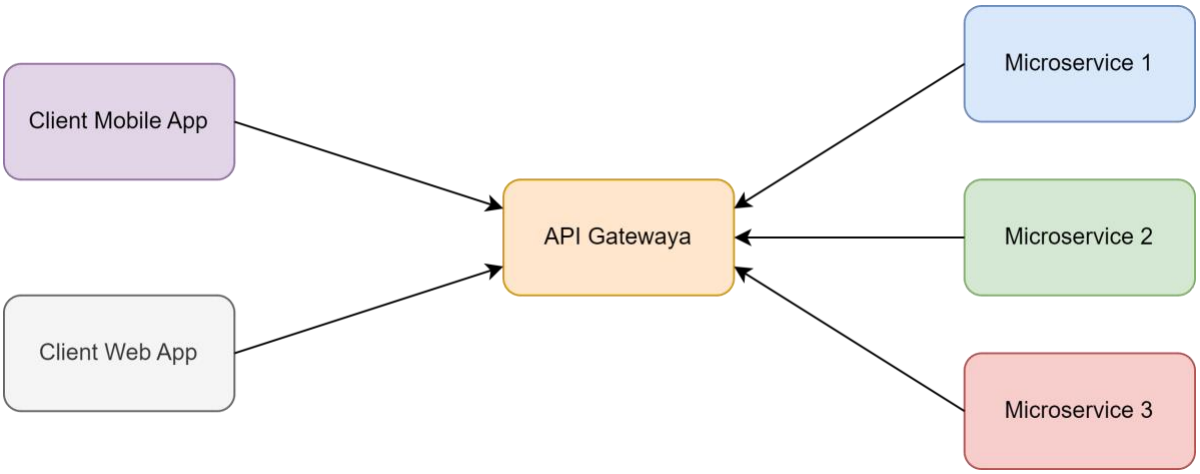
2.1.6 API Gateway

Σε μια αρχιτεκτονική microservices η εφαρμογή πελάτη συνήθως χρειάζεται να καταναλώσει λειτουργίες από περισσότερα από ένα microservice. Αν αυτή η κατανάλωση εκτελούταν απευθείας, οι πελάτες θα χρειαζόντουσαν να διαχειριστούν πολλαπλές κλήσεις στα endpoints των mircoservices. Τι θα συμβεί αν η εφαρμογή εξελισσόταν και αναπτύσσονταν νέα microservices ή ήδη υπάρχων microservices ενημερώνονταν. Αν μια εφαρμογή έχει πολλά microservices και διαχειρίζεται πολλά endopoints από την μεριά της εφαρμογής πελάτη αυτό είναι ένας εφιάλτης.
Συνεπώς αν έχεις ένα ενδιάμεσο επίπεδο (API Gateway) μπορεί να είναι χρήσιμο για εφαρμογές που χρησιμοποιούν την microservices αρχιτεκτονική. Αν δεν έχεις ένα API Gateway οι εφαρμογές πελάτη πρέπει να μιλούν απευθείας με τα microservices και αυτό δημιουργεί πολλά προβλήματα. Μερικά από αυτά είναι:

Coupling: Χωρίς ένα API Gateway οι εφαρμογές πελάτη είναι σφιχτά συνδεδεμένες με τα εσωτερικά microservices.

Πολλά round trips: Μια single page/screen εφαρμογή πελάτη μπορεί να χρειάζεται να κάνει αρκετές κλήσεις σε πολλαπλά services. Αυτή η προσέγγιση έχει σαν αποτέλεσμα πολλά round trips στο δίκτυο μεταξύ πελάτη και server και αυτό προσθέτει σημαντική καθυστέρηση στην εφαρμογή.

Cross-cutting προβληματισμοί: Microservices που είναι δημόσια πρέπει να διαχειρίζονται προβλήματα όπως εξουσιοδότηση και SSL. Αυτά τα προβλήματα μπορούν να λυθούν στο επίπεδο του API Gateway και έτσι τα microservices να είναι απλούστερα.



2.1.7 Service Discovery

Ας φανταστούμε ότι έχουμε μια εφαρμογή η οποία αποτελείται από πολλά microservices. Αυτά επικοινωνούν με κάποιο τρόπο μεταξύ τους (παράδειγμα REST API, gRPC).

Μια εφαρμογή με αρχιτεκτονική microservices τυπικά τρέχει σε ένα εικονικό ή containerized περιβάλλον. Ο αριθμός των services και οι τοποθεσίες τους αλλάζουν δυναμικά. Εμείς χρειαζόμαστε να ξέρουμε που είναι αυτά τα services και τα ονόματα τους ώστε να επιτρέψουμε στα αιτήματα να φτάσουν

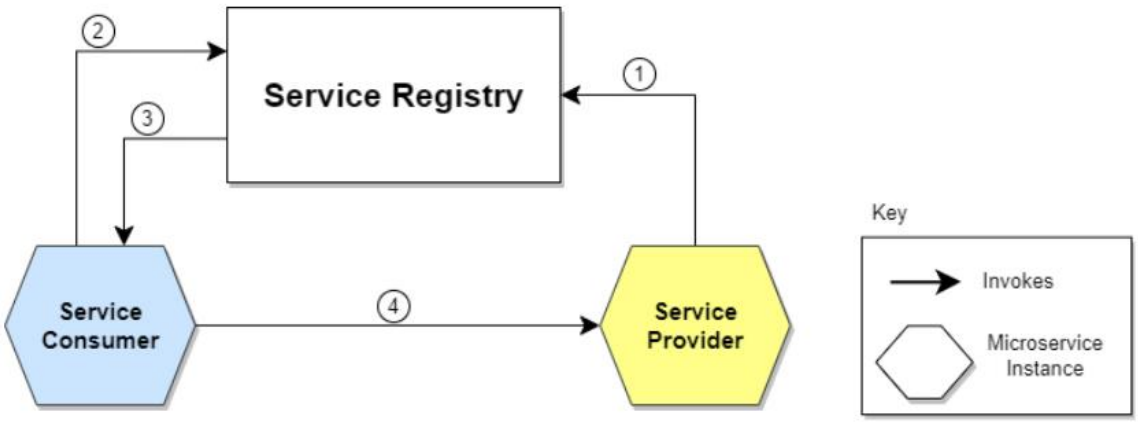
στο microservice το οποίο θέλουν να επικοινωνήσουν. Για να το πετύχουμε αυτό χρησιμοποιούμε τακτικές όπως το Service Discovery.

Το Service Discovery μας βοηθάει να ξέρουμε την τοποθεσία του κάθε service. Άρα το Service Discovery λειτουργεί σαν ένα μητρώο όπου παρακολουθεί όλες τις διευθύνσεις των services. Σε κάθε service έχει ανατεθεί δυναμικά μια διαδικτυακή διαδρομή. Συνεπώς, αν ένας πελάτης θέλει να στείλει ένα αίτημα σε ένα service χρειάζεται έναν μηχανισμό Service Discovery για να το πετύχει.

Το Service Discovery ελέγχει τα πράγματα σε δύο μέρη. Πρώτα, παρέχει έναν μηχανισμό όπου ένα service εγγράφεται και λέει “Είμαι εδώ”. Δεύτερον, παρέχει έναν τρόπο να μπορείς να βρεις το service μόλις αυτό εγγράφεται.

Παράδειγμα έχουμε ένα service καταναλωτής και ένα service πάροχος (ένα service με REST API). Το service καταναλωτής χρειάζεται το service πάροχο ώστε να γράψει και να διαβάσει δεδομένα.

Το ακόλουθο διάγραμμα αναπαριστά τη ροή επικοινωνίας:



Ας περιγράψουμε τα βήματα τα οποία απεικονίζονται στο διάγραμμα:

- Η τοποθεσία του service παρόχου στέλνεται στο service registry (μια βάση δεδομένων όπου εμπεριέχει τις τοποθεσίες όλων των διαθέσιμων services).
- Το service καταναλωτή ρωτάει τον Service Discovery Server για την τοποθεσία του service παρόχου.
- Η τοποθεσία του service παρόχου αναζητείται από το service registry στις εσωτερικές του βάσεις δεδομένων και στέλνεται πίσω στο service καταναλωτή.
- Το service καταναλωτή μπορεί τώρα να δημιουργήσει ένα απευθείας αίτημα στο service πάροχο

2.2 Ανάπτυξη Microservices Χρησιμοποιώντας το Μοτίβο “Container as a Service”

2.2.1 Τι είναι Containerization

Containerization είναι το πακετάρισμα του κώδικα με τα πλήρως απαραίτητα συστατικά για να μπορεί να πραγματοποιηθεί η εκτέλεση του. Τα συστατικά αυτά είναι το λειτουργικό σύστημα, οι βιβλιοθήκες που χρησιμοποιεί ο κώδικας και οι εξαρτήσεις που έχει. Το πακέτο αυτό το ονομάζουμε container και μπορεί να εκτελεστεί σε οποιοδήποτε υπολογιστικό περιβάλλον. Οι containers είναι πιο φορητοί και αποτελεσματικοί σε πόρους απο τις εικονικές μηχανές, γι’ αυτό είναι η κατάλληλη επιλογή για μια μοντέρνα cloud native εφαρμογή.

Η τεχνική του containerization επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές πιο γρήγορα και με περισσότερη αφάλεια. Με τις κλασσικές μεθόδους ανάπτυξης ο κώδικας αναπτύσσεται σε ένα συγκεκριμένο υπολογιστικό περιβάλλον. Αυτό έχει ως αποτέλεσμα όταν μεταφερθεί σε ένα διαφορετικό περιβάλλον να εμφανίζει προβλήματα. Παράδειγμα τέτοιον μεταφορών είναι όταν οι προγραμματιστές μεταφέρουν τον κώδικα μιας εφαρμογής από ένα desktop υπολογιστή σε μια εικονική μηχανή ή από ένα λειτουργικό σύστημα linux σε ένα λειτουργικό σύστημα Windows.

Με αυτά που προσφέρει το containerization έχει εξαλειφθεί αυτό το πρόβλημα κασκευάζοντας τον κώδικα της εφαρμογής μαζί με τα σχετικά αρχεία ρυθμίσεων, τις βιβλιοθήκες, και τις εξαρτήσεις που είναι απαραίτητα για την εκτέλεση του.

Η ιδέα του conatinerization δεν είναι καινούρια στο χώρο τις τεχνολογίας. Παρ’ όλα αυτά αναδείχθηκε το 2013 με την τεχνολογία Docker Engine, ένα πρότυπο της βιομηχανίας για τους containers με απλά προγραμματιστικά εργαλεία και μια καθολική προσέγγιση πακεταρίσματος που βοήθησε στην επιτάχυνση της υιοθέτησης αυτής της τεχνολογίας. Σήμερα ολοένα και περισσότερο οι οργανισμοί χρησιμοποιούν την τεχνολογία του containerization για να δημιουργήσουν καινούριες εφαρμογές και να εξελίξουν τις ήδη υπάρχουσες μεταφέροντας τες στο cloud.

2.2.2 Διαφορές Μεταξύ Containerization και Virtualization

Οι containers και οι εικονικές μηχανές είναι τεχνολογίες πακεταρίσματος ενός υπολογιστικού περιβάλλοντος όπου συνδυάζουν πληθώρα IT συστατικών και τα απομονώνουν από το υπόλοιπο σύστημα. Οι βασικές διαφορές τους είναι στην επέκταση και στη φορητότητα.

Το μέγεθος των containers μετριέται σε megabyte. Δεν πακετάρουν τίποτα μεγαλύτερο από μία εφαρμογή και όλα τα αρχεία που είναι απαραίτητα για την εκτέλεση της. Επιπλέον, οι containers συχνά χρησιμοποιούνται για να πακετάρουν μια μόνο συνάρτηση όπου εκτελεί μια συγκεκριμένη εργασία.

Οι εικονικές μηχανές μετριούνται σε gigabyte. Συνήθως εμπεριέχουν το δικό τους λειτουργικό σύστημα όπου τους επιτρέπει να εκτελούν πολλαπλές λειτουργίες σε μία. Η αυξημένοι πόροι που είναι διαθέσιμοι σε μια εικονική μηχανή της επιτρέπουν να προσομοιώνει ολόκληρους servers, λειτουργικά συστήματα, βάσεις δεδομένων, και δίκτυα.

Η μικρή και ελαφριά φύση των containers τους επιτρέπει να μπορούν να μετακινηθούν εύκολα σε διαφορετικά συστήματα καθώς και σε δημοσία, ιδιωτικά, υβριδικά, και multicloud περιβάλλοντα. Επίσης, είναι ιδανικά περιβάλλοντα για να διανείμουν μια cloud native εφαρμογή όπου αποτελείται από μια συλλογή microservices.

Μια εικονική μηχανή είναι ικανή να τρέξει πολύ περισσότερες λειτουργίες από έναν container. Γι’ αυτό το λόγο είναι ο παραδοσιακός τρόπος για να πακετάρεις μια εφαρμογή μονόλιθου. Αυτή η διευρυμένη

λειτουργικότητα των εικονικών μηχανών έχει σαν αποτέλεσμα λιγότερη φορητότητα λόγω των εξαρτήσεων από το λειτουργικό σύστημα, τις εφαρμογές, και τις βιβλιοθήκες.

2.2.3 Microservices και Containerization

Οι ιδέες πίσω από τα microservices και containerization είναι όμοιες αφού και οι δύο είναι software development πρακτικές όπου μετατρέπουν τις εφαρμογές σε συλλογές από μικρότερα services όπου είναι φορητά, επεκτάσιμα, αποδοτικά, και πιο εύκολα στη διαχείριση τους.

Επιπλέον, τα microservices και το containerization δουλεύουν πολύ καλά όταν συνεργάζονται μεταξύ τους. Οι containers παρέχουν μια ελαφριά ενθυλάκωση οποιασδήποτε εφαρμογής είτε είναι μια εφαρμογή μονόλιθος είτε είναι μια εφαρμογή microservices. Όταν αναπτύσσεται ένα microservice μέσα σε έναν container, τότε κερδίζει όλα τα πλεονεκτήματα που προσφέρει το containerization, όπως φορητότητα, ευελιξία στους προγραμματιστές, fault isolation, καλύτερες αποδόσεις στους servers, αυτοματοποίηση στην εγκατάσταση, επεκτασιμότητα και διαχείριση, και τέλος επίπεδα ασφαλείας.

Οι containers, τα microservices, και το cloud computing συνεργάζονται μεταξύ τους για να φέρουν την ανάπτυξη των εφαρμογών και διανομή τους σε νέα επίπεδα όπου δεν θα ήταν εφικτό να φτάσουμε με τις παραδοσιακές μεθοδολογίες και περιβάλλοντα. Αυτές η επόμενης γενιάς προσεγγίσεις προσθέτουν ευελιξία, αποδοτικότητα, αξιοπιστία, και ασφάλεια στον κύκλο ζωής της ανάπτυξης λογισμικού.

2.2.4 Ασφάλεια

Οι containerized εφαρμογές έχουν από μόνες τους ένα επίπεδο ασφάλειας. Ο λόγος είναι γιατί μπορούν να τρέξουν σαν απομονωμένες διαδικασίες και να λειτουργήσουν ανεξάρτητα από άλλους containers. Η πραγματική απομόνωση μπορεί να αποτρέψει οποιονδήποτε κακόβουλο κώδικα από το να επηρεάσει άλλους containers η να εισχωρήσει στο σύστημα. Παρόλα αυτά, επίπεδα εφαρμογής στους containers συχνά μοιράζονται μεταξύ των containers. Αυτό είναι πολύ θετικό για την αποτελεσματικότητα των πόρων, αλλά ανοίγει την πόρτα στην παραβίαση μεταξύ των containers. Το ίδιο συμβαίνει και με τον διαμοιρασμό στο λειτουργικό σύστημα όπου πολύ containers μπορεί να σχετίζονται με το ίδιο λειτουργικό σύστημα. Απειλές ασφάλειας σε ένα κοινό λειτουργικό σύστημα μπορεί να επηρεάσουν όλους τους containers που σχετίζονται με αυτό.

Πως μπορεί μια εφαρμογή να πακεταριστεί σε έναν container και αυτό να βελτιώσει την ασφάλεια; Οι πάροχοι container τεχνολογιών όπως το docker συνεχίζουν να αντιμετωπίζουν προκλήσεις ασφάλειας στους containers. Το containerization έχει μια προσέγγιση “ασφάλειας-εξ’ ορισμού” πιστεύοντας ότι η ασφάλεια πρέπει να είναι έμφυτη στην πλατφόρμα και όχι μια ξεχωριστεί λύση. Τέλος, η container μηχανή υποστηρίζει όλες τις εξ’ ορισμού ιδιότητες απομόνωσης έμφυτα στο υποκείμενο λειτουργικό σύστημα. Μπορούν να οριστούν security δικαιώματα ώστε να γίνεται αυτόματα μπλοκάρισμα ανεπιθύμητων τμημάτων που προσπαθούν να εισέλθουν στους containers η να μειωθεί η επικοινωνία με αχρείαστους πόρους.

Ερευνητές δουλεύουν για να δυναμώσουν περεταίρω την ασφάλεια σε linux containers, και ένα ευρύ φάσμα από λύσεις ασφαλείας είναι διαθέσιμο για να αυτοματοποιήσει τον εντοπισμό και την απάντηση σε απειλές κατά μήκος μιας εταιρίας, να παρακολουθήσει και να εφαρμόσει τα πρότυπα της βιομηχανίας και τις πολιτικές ασφάλειας, να διασφαλίσει την ασφαλή ροή των δεδομένων μέσα από τις εφαρμογές και τα endpoints, και πολλά περισσότερα.

2.2.5 Οφέλη

Το containerization προσφέρει σημαντικά οφέλη στους προγραμματιστές και στις ομάδες ανάπτυξης. Αυτά είναι:

Φορητότητα: Ένας container δημιουργεί ένα εκτελέσιμο πακέτο λογισμικού όπου δεν εξαρτάται από το λειτουργικό σύστημα. Συνεπώς είναι φορητό και μπορεί να τρέξει ομοιόμορφα και σταθερά σε οποιαδήποτε πλατφόρμα ή cloud.

Ευκινησία: Η ανοιχτογύ κώδικα Docker Engine όπου χρησιμοποιείται για να τρέχουν οι containers είναι ένα πρότυπο βιομηχανίας για την τεχνολογία των containers με απλά προγραμματιστικά εργαλεία και μια καθολική προσέγγιση πακεταρίσματος όπου δουλεύει στο λειτουργικό σύστημα Linux και στο λειτουργικό σύστημα Windows.

Ταχύτητα: Οι containers συνήθως αναφέρονται ως “lightweight”, αυτό σημαίνει ότι μοιράζονται τον πυρήνα του λειτουργικού συστήματος της μηχανής και δεν είναι φορτωμένοι με οτιδήποτε άλλο περιττό για τη λειτουργία τους. Αυτό έχει σαν αποτέλεσμα ο server να έχει υψηλότερη αποδοτικότητα και επιπλέον μειώνει τα κόστη των αδειών ενώ κάνει ταχύτερους τους χρόνους εκκίνησης της εφαρμογής αφού δεν χρειάζεται να τρέξει κάποιο λειτουργικό σύστημα για να ξεκινήσει η εφαρμογή τη λειτουργία της.

Σφάλμα Απομόνωσης: Κάθε containerized εφαρμογή είναι απομονωμένη και λειτουργεί ανεξάρτητα από τις υπόλοιπες. Η αποτυχία ενός container δεν επιδρά στη λειτουργία των υπόλοιπων containers. Μια ομάδα ανάπτυξης μπορεί να εντοπίσει και να διορθώσει ένα τεχνικό πρόβλημα σε έναν container χωρίς να θέση εκτός λειτουργίας τους υπόλοιπους.

Αποτελεσματικότητα: Το λογισμικό όπου τρέχει σε ένα containerized περιβάλλον μοιράζεται τον πυρήνα του λειτουργικού συστήματος της μηχανής και τα επίπεδα της εφαρμογής εντός ενός container μπορούν να μοιραστούν σε όλους τους containers. Συνεπώς οι containers είναι μικρότεροι σε μέγεθος από τις εικονικές μηχανές και χρειάζονται λιγότερο χρόνο εκκίνησης. Αυτό έχει ως αποτέλεσμα να μπορούν πολύ περισσότεροι containers να τρέξουν σε ένα υπολογιστικό σύστημα αντί για εικονικές μηχανές.

Εύκολο στη διαχείριση: Μια container orchestration πλατφόρμα μπορεί να αυτοματοποιήσει την εγκατάσταση, την επεκτασιμότητα και να διαχειριστεί το φόρτο και τις υπηρεσίες των containers. Οι πλατφόρμες container orchestration μπορούν εύκολα να διαχειριστούν εργασίες όπως επεκτασιμότητα μιας containerized εφαρμογής, λανσάρισμα νέων εκδόσεων, παροχή παρακολούθησης, logging, debugging, και άλλα. Το πιο γνωστό σύστημα container orchestration που είναι διαθέσιμο είναι το Kubernetes, μια ανοιχτού κώδικα τεχνολογία η οποία αυτοματοποιεί Linux container λειτουργίες.

Security: Η απομόνωση των εφαρμογών σε containers έμφυτα αποτρέπει την εισβολή κακόβουλου κώδικα από το να επηρεάσει άλλους containers η συστήματα φιλοξενίας. Επιπρόσθετα, δικαιώματα ασφαλείας μπορούν να ορισθούν για να μπλοκάρουν αυτόματα μη επιθυμητά στοιχεία να εισαχθούν σε containers ή για να βάλουν όρια σε επικοινωνίες με περιττές πηγές.

2.2.6 Container Orchestration

Το container orchestration αυτοματοποιεί την ανάπτυξη, την διαχείριση, την επεκτασιμότητα, και τη δικτύωση στους containers. Εταιρίες που χρειάζονται να κάνουν deploy και να διαχειριστούν εκατοντάδες ή χιλιάδες linux containers μπορούν να έχουν κέρδος από το container orchestration.

Το container orchestrations μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον που μπορείς να χρησιμοποιείς containers. Μπορεί να σε βοηθήσει να κάνεις deploy την εφαρμογή σου σε διαφορετικά περιβάλλοντα χωρίς να χρειάζεται να την τροποποιήσεις. Και τα microservices μέσα σε containers κάνουν πιο εύκολη την ενορχήστρωση των services, περιλαμβανομένης της αποθήκευσης, της δικτύωσης και της ασφάλειας.

Οι containers έδωσαν μια ιδανική deployment μονάδα σε εφαρμογές που είναι βασισμένες σε αρχιτεκτονική microservices. Έκανα εφικτό να μπορείς να τρέχεις πολλαπλά κομμάτια μιας εφαρμογής ανεξάρτητα σε μια αρχιτεκτονική microservices, στο ίδιο υλικό, με πολύ καλύτερο έλεγχο σε ατομικά κομμάτια και κύκλους ζωής.

Επίσης, η διαχείριση του κύκλου ζωής των containers με το container orchestration υποστηρίζει DevOps ομάδες που το ενσωματώνουν μέσα στην CI/CD ροή εργασιών. Μαζί με το application programming interfaces (APIs) και τις DevOps ομάδες, τα containerized microservices είναι τα θεμέλια των cloud-native-applications

3. Σχεδιασμός Εφαρμογής

3.1 Περιγραφή Εφαρμογής

Με την ανάπτυξη του Internet ο κόσμος πλέον έχει μάθει να το χρησιμοποιεί για να διεκπεραιώνει διάφορες δουλείες και συναλλαγές που παλιότερα χρειαζόταν η φυσική του παρουσία για να τις πραγματοποιήσει. Μερικά παραδείγματα είναι τα ψώνια μέσω Internet, η πληρωμή λογαριασμών, η εκμάθηση νέων δεξιοτήτων και πολλά αλλά. Η εφαρμογή που σχεδιάζουμε θα βοηθάει τον κόσμο μέσω του Internet να βρίσκει κατάλυμα προς ενοικίαση για να διαμείνει για ένα συγκεκριμένο χρονικό διάστημα.

Οι χρήστες της εφαρμογής θα έχουν τη δυνατότητα να αναζητήσουν μια κατοικία για να μείνουν με βάσει την τοποθεσία στην οποία θέλουν να μείνουν, τις ημερομηνίες άφιξης και αναχώρησης τους, και το πλήθος των ατόμων το οποίο θα φιλοξενηθεί στην κατοικία. Μόλις ο χρήστης δώσει αυτές τις πληροφορίες η εφαρμογή θα είναι σε θέση να αναζητήσει κατοικίες που πληρούν αυτά τα κριτήρια και να τις παρουσιάσει στο χρήστη με κατάλληλο τρόπο ώστε ο χρήστης να μπορεί να βρει με ευκολία την κατοικία που του ταιριάζει καλύτερα. Μόλις ο χρήστης βρει μια κατοικία που τον ενδιαφέρει θα μπορεί να δει αναλυτικά πληροφορίες για αυτή και να κάνει μια κράτηση αν επιθυμεί.

Επιπλέον οι χρήστες θα είναι σε θέσει να δουν τις κρατήσεις που έχουν κάνει. Τα στοιχεία του ιδιοκτήτη της κατοικίας στην οποία έχουν κάνει την κράτηση, την περιγραφή της κατοικίας και τι παρέχει στους φιλοξενούμενους της, καθώς και τις μέρες τις οποίες θέλουν να διαμείνουν σε αυτή. Τέλος αν μετανιώσει ένας χρήστης για την κράτηση που έχει κάνει είναι σε θέση να την ακυρώσει. Όταν ακυρωθεί μια κράτηση αποστέλνεται email στον ιδιοκτήτη της κατοικίας με τις πληροφορίες της ακύρωσης.

Πέρα από τους χρήστες που θέλουν να βρουν μια κατοικία για να διαμείνουν ένα συγκεκριμένο χρονικό διάστημα, η εφαρμογή απευθύνεται και σε χρήστες που θέλουν να δημοσιεύσουν μια αγγελία μιας κατοικίας που διαθέτουν με σκοπό το χρηματικό κέρδος. Για να επιτευχθεί αυτό, η εφαρμογή διαθέτει ειδική φόρμα για τη συλλογή των χαρακτηριστικών της κατοικίας του χρήστη που θέλει να τη διαθέσει προς ενοικίαση. Έπειτα όταν κάποιος χρήστης κάνει μια κράτηση σε αυτήν την κατοικία αποστέλνεται ένα email στον ιδιοκτήτη της κατοικίας αυτής με τις πληροφορίες της κράτησης και του ανθρώπου όπου πραγματοποίησε την κράτηση.

Η εφαρμογή παρέχει τις απαραίτητες λειτουργίες για να βοηθήσει ένα χρήστη να αναζητήσει και να βρει ένα κατάλυμα για να μείνει. Επιπρόσθετα, παρέχει δυνατότητες διαχείρισης των διαφόρων κρατήσεων που έχει κάνει. Επιπλέον, η εφαρμογή δίνει τη δυνατότητα σε ανθρώπους που έχουν κατοικίες στην κατοχή τους να δημοσιεύσουν αγγελίες παρουσιάζοντας τες με σκοπό την εύρεση νέων ενοικιαστών.

3.2 Περιγραφή Μοντέλου Εφαρμογής

Τα αντικείμενα στην εφαρμογή είναι τρία, η κατοικία, ο χρήστης της εφαρμογής, και η κράτηση η οποία πραγματοποιείται από τους χρήστες. Παρακάτω θα δούμε τα δεδομένα από τα οποία αποτελείται το κάθε αντικείμενο.

Κατοικία (Property)

id

Τύπος: Long

Περιγραφή:

Το id είναι ένας αριθμός ο οποίος είναι μοναδικός για κάθε κατοικία.

PropertyType

Τύπος: object

Δεδομένα:

Όνομα	Τύπος
Id	Long
Name	String

Περιγραφή:

Το αντικείμενο propertyType χρησιμοποιείται για την καταγραφή του τύπου μιας κατοικίας. Ο τύπος που μπορεί να έχει μια κατοικία είναι ένας από τους ακόλουθους house, apartment, secondary unit, unique space, bed and breakfast.

GuestSpace

Τύπος: object

Δεδομένα:

Όνομα	Τύπος
Id	Long
Name	String

Περιγραφή:

Το αντικείμενο guestSpace χρησιμοποιείται για την καταγραφή του χώρου που θα μπορούν να μείνουν στην κατοικία οι άνθρωποι οι οποίοι έχουν κάνει κράτηση σε αυτή. Οι επιλογές του χώρου είναι οι εξής, entire place, private room, shared room.

maxGuestNumber

Τύπος: Integer

Περιγραφή:

Ο αριθμός maxGuestNumber αντιπροσωπεύει το όριο καλεσμένων το οποίο μπορεί να δεχθεί μια κατοικία.

bedroomNumber

Τυπος: Integer

Περιγραφή:

Ο αριθμός bedroomNumber αντιπροσωπεύει τον αριθμό των υπνοδωματίων της εκάστοτε κατοικίας.

bathNumber

Τύπος: Integer

Περιγραφή:

Ο αριθμός bathNumber αντιπροσωπεύει τον αριθμό των μπάνιων της εκάστοτε κατοικίας.

title

Τύπος: String

Περιγραφή:

Το αλφαριθμητικό title αντιπροσωπεύει τον τίτλο της αγγελίας της εκάστοτε κατοικίας.

description

Τύπος: String

Περιγραφή:

Το αλφαριθμητικό description αντιπροσωπεύει την περιγραφή της αγγελίας της εκάστοτε κατοικίας.

pricePerNight

Τύπος: Float

Περιγραφή:

Ο πραγματικός αριθμός pricePerNight αντιπροσωπεύει το κόστος της κατοικίας ανά βραδιά.

owner

Τύπος: string

Περιγραφή:

Το αλφαριθμητικό owner είναι ένα αναγνωριστικό που αντιστοιχίζεται μοναδικά στον κάθε χρήστη της εφαρμογής.

amenities

Τύπος: List of objects

Όνομα	**Τύπος**
Id	Long
Name	String

Περιγραφή:

Η λίστα αντικειμένων amenities είναι μια λίστα με τις ανέσεις τις οποίες έχει μια κατοικία. Μερικές από τις ανέσεις που μπορεί να έχει μια κατοικία και ο χρήστης είναι σε θέση να της δηλώσει σε αυτήν τη λίστα είναι το wifi, breakfast, indoor fireplace και άλλα.

Address

Τύπος: Object

Δεδομένα:

Όνομα	**Τύπος**
Id	Long
City	String
Country	Object
Postcode	String
streetName	String
streetNumber	int

Περιγραφή:

Στο αντικείμενο address καταγράφεται η διεύθυνση της κατοικίας.

Images

Τύπος: List of objects

Δεδομένα:

Όνομα	Τύπος
Id	Long
Name	String

Περιγραφή:

Η λίστα αντικειμένων images είναι μια λίστα η οποία εμπεριέχει τις διαδρομές των αρχείων όλων των φωτογραφιών οι οποίες απεικονίζουν την εκάστοτε κατοικία.

Κράτηση (Reservation)

id

Τύπος: String

Περιγραφή:

Το id είναι ένα αλφαριθμητικό το οποίο αναγνωρίζει μοναδικά την κάθε κράτηση.

checkIn

Τύπος: LocalDate

Περιγραφή:

Το αντικείμενο checkIn καταγράφει τη μέρα που ο χρήστης θέλει να ξεκινήσει να φιλοξενείται σε μια κατοικία.

checkOut

Τύπος: LocalDate

Περιγραφή:

Το αντικείμενο checkOut καταγράφει την τελευταία μέρα που ο χρήστης θέλει να φιλοξενηθεί σε μια κατοικία.

Location

Τύπος: String

Περιγραφή:

Στο αλφαριθμητικό location καταγράφεται η τοποθεσία στην οποία βρίσκεται η κατοικία στην οποία έχει γίνει η κράτηση.

Price

Τύπος: BigDecimal

Περιγραφή:

Στο αντικείμενο price καταγράφεται η συνολική τιμή της κράτησης.

userId

Τύπος: UUID

Περιγραφή:

Στο αντικείμενο userId καταγράφεται ένα αναγνωριστικό το οποίο είναι μοναδικό για κάθε χρήστη και αντιστοιχεί στον χρήστη που θέλει να νοικιάσει την κατοικία.

ownerId

Τύπος: UUID

Περιγραφή:

Στο αντικείμενο ownerId καταγράφεται ένα αναγνωριστικό το οποίο είναι μοναδικό για κάθε χρήστη και αντιστοιχεί στον χρήστη όπου του ανήκει ή κατοικία.

Χρήστης (User)

id

Τύπος: UUID

Περιγραφή:

Στο αντικείμενο id καταγράφεται ένα αναγνωριστικό το οποίο είναι μοναδικό για κάθε χρήστη.

email

Τύπος: String

Περιγραφή:

Στο αλφαριθμητικό email καταγράφεται το email του χρήστη.

firstName

Τύπος: String

Περιγραφή:

Στο αλφαριθμητικό firstName καταγράφεται το όνομα του χρήστη.

lastName

Τύπος: String

Περιγραφή:

Στο αλφαριθμητικό lastName καταγράφεται το επώνυμο του χρήστη.

phone

Τύπος: String

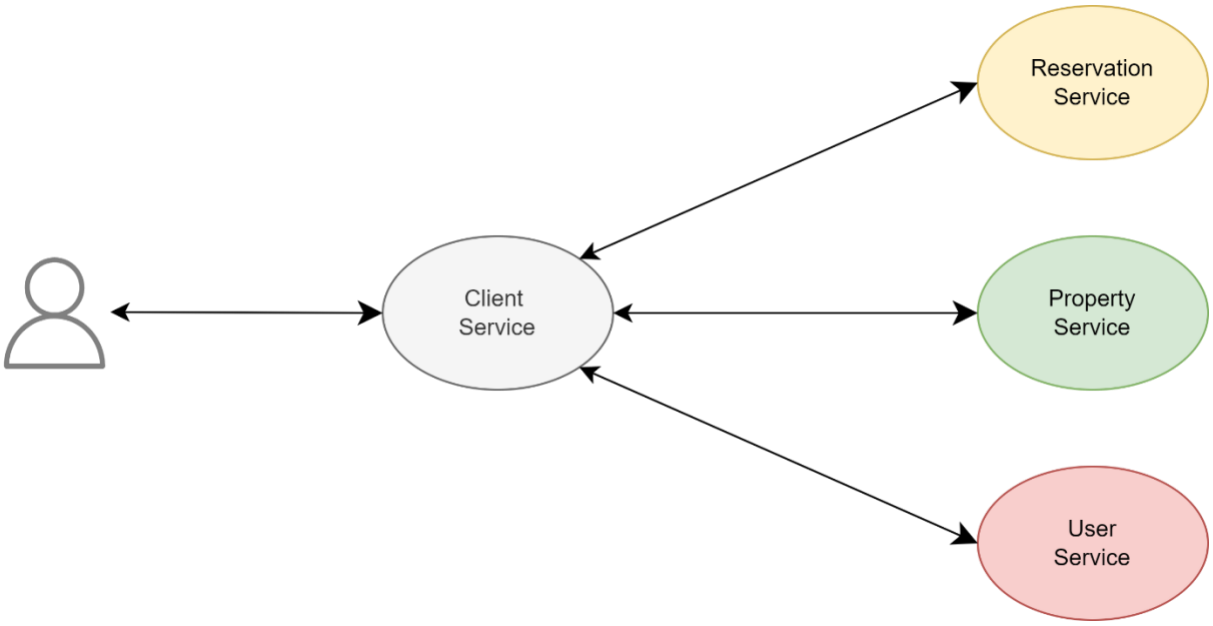
Περιγραφή:

Στο αλφαριθμητικό phone καταγράφεται το τηλέφωνο του χρήστη.

3.3 Καθορισμός Υπηρεσιών

Όταν θέλουμε να δημιουργήσουμε μια εφαρμογή με αρχιτεκτονική microservices μια από τις πρώτες εργασίες που πρέπει να γίνουν είναι να καθορίσω τα services από τα οποία θα αποτελείται η εφαρμογή. Καθώς η εφαρμογή αναπτύσσεται μπορεί να χρειαστεί να γίνουν αλλαγές στον καθορισμό των services. Τέτοιες αλλαγές μπορεί να είναι η διάσπαση ενός service σε μικρότερα ή η ένωση δύο ή περισσοτέρων services.

Η εφαρμογή που αναπτύσσουμε αποτελείται από τέσσερα services. Το client service το οποίο εμπεριέχει τη διεπαφή χρήστη και επιπλέον κάνει τη δουλεία ενός load balancer ο οποίος επικοινωνεί με τον έξω κόσμο και προωθεί τα αιτήματα του κατάλληλα στα υπόλοιπα services. Το Reservation Service το οποίο είναι υπεύθυνο για τη διαχείριση των κρατήσεων που πραγματοποιούνται στην εφαρμογή. Το Property Service το οποίο διαχειρίζεται τις αγγελίες και τις κατοικίες όπου υπάρχουν στην εφαρμογή και τέλος το User Service το οποίο είναι υπεύθυνο για τη διαχείριση των χρηστών.



Γραφική απεικόνιση των services της εφαρμογής

3.4 Λειτουργίες

Κάθε service της εφαρμογής εκτελεί συγκεκριμένες λειτουργίες που έχουν να κάνουν με το αντικείμενο το οποίο ασχολείται. Για παράδειγμα το Property Service είναι υπεύθυνο για λειτουργίες που έχουν να κάνουν με τις κατοικίες και τις αγγελίες των κατοικιών. Παρακάτω θα δούμε τί λειτουργίες έχει το κάθε service ξεχωριστά.

PropertyService:

Αναζήτηση Κατοικιών

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/properties/search	GET	Αναζήτηση κατοικιών με βάσει συγκεκριμένα χαρακτηριστικά τα οποία θέτει ο χρήστης.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
location	Query Parameter	String	Απαιτείται	Η τοποθεσία στην οποία βρίσκεται η κατοικία.
checkIn	Query Parameter	Date	Απαιτείται	Η ημέρα που ο χρήστης θέλει να ξεκινήσει να νοικιάζει την κατοικία.
checkOut	Query Parameter	Date	Απαιτείται	Η ήμερα που ο χρήστης θέλει σταματήσει να νοικιάζει την κατοικία.
guestNumber	Query Parameter	Number	Απαιτείται	Ο αριθμός των ανθρώπων οι οποίοι θα φιλοξενηθούν στην κατοικία
currentPage	Query Parameter	Number	Απαιτείται	Αριθμός σελίδας

Δημιουργία Αγγελίας

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/properties/create-property	POST	Δημιουργία αγγελίας.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.
FormData	Form Parameter	multipart/form-data	Απαιτείται	Περιέχει τις πληροφορίες που απαιτούνται για την δημιουργία μιας αγγελίας.

Εύρεση Κατοικίας

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/properties/property/{propertyId}	GET	Εύρεση αναλυτικών στοιχείων συγκεκριμένης κατοικίας.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
propertyId	Path Parameter	Number	Απαιτείται	Μοναδικό αναγνωριστικό για κάθε κατοικία.

Εύρεση Κατοικιών Συγκεκριμένου Χρήστη

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/properties/my-properties/{currentPage}	GET	Εύρεση κατοικιών που αντιστοιχούν σε συγκεκριμένο χρήστη.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
currentPage	Path Parameter	Number	Απαιτείται	Αριθμός Σελίδας.
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

Διαγραφή Κατοικίας

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/properties/delete/{propertyId}	DELETE	Διαγραφή κατοικίας.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
currentPage	Path Parameter	Number	Απαιτείται	Αριθμός Σελίδας.
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

Εύρεση Όλων των Χωρών

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/countries/all	GET	Εύρεση όλων των χωρών που υπάρχουν κατοικίες στην εφαρμογή.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή

Reservation Service:

Εύρεση IDs κατοικιών

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/reservations/	GET	Εύρεση Ids των κατοικιών που έχουν κάνει κράτηση χρήστες της εφαρμογής όπου μέρος της κράτησης τους βρίσκεται στο διάστημα μεταξύ των δύο ημερομηνιών. Επιπλέον η εύρεση γίνεται για κατοικίες που βρίσκονται σε συγκεκριμένη χώρα

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
Location	Query Parameter	String	Απαιτείται	Χωρά στην οποία θα γίνει η αναζήτηση των κατοικιών.
checkIn	Query Parameter	Date	Απαιτείται	Η πρώτη μέρα της περιόδου όπου θα γίνει η αναζήτηση.
checkout	Query Parameter	Date	Απαιτείται	Η τελευταία μέρα της περιόδου όπου θα γίνει η αναζήτηση.

Εύρεση Κρατήσεων Συγκεκριμένου Χρήστη

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/reservations/	GET	Εύρεση των κρατήσεων συγκεκριμένου χρήστη.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα

				στοιχεία του χρήστη.

Δημιουργία Κράτησης

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/reservations/create	POST	Δημιουργία Κράτησης.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
reservation	Body Parameter	Object	Απαιτείται	Αντικείμενο το οποίο περιέχει τα δεδομένα της κράτησης
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

Διαγραφή Κράτησης

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/reservations/delete/{reservationId}	DELETE	Διαγραφή κράτησης.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
reservationId	Path Parameter	String	Απαιτείται	Μοναδικό αναγνωριστικό για κάθε κράτησης
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

User Service:

Εύρεση Στοιχείων Χρήστη

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/users/me	GET	Εύρεση στοιχείων χρήστη.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

Εύρεση Χρήστη με Βάση το ID

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/users/user-id/{userId}	GET	Εύρεση στοιχείων χρήστη με βάση το id του.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
id	Path Parameter	UUID	Απαιτείται	Id χρήστη

Εύρεση Χρήστη με Βάση το email

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/users/{email}	GET	Εύρεση στοιχείων χρήστη με βάση το email του.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
Email	Path Parameter	String	Απαιτείται	Email χρήστη.

Ενημέρωση Στοιχείων Χρήστη

Αίτημα

Όνομα	Μέθοδος	Περιγραφή
/users/update	PUT	Ενημέρωση στοιχείων χρήστη.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
user	Body Parameter	Object	Απαιτείται	Στοιχεία χρήστη.
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

Ενημέρωση Φωτογραφίας Χρήστη

Αίτημα

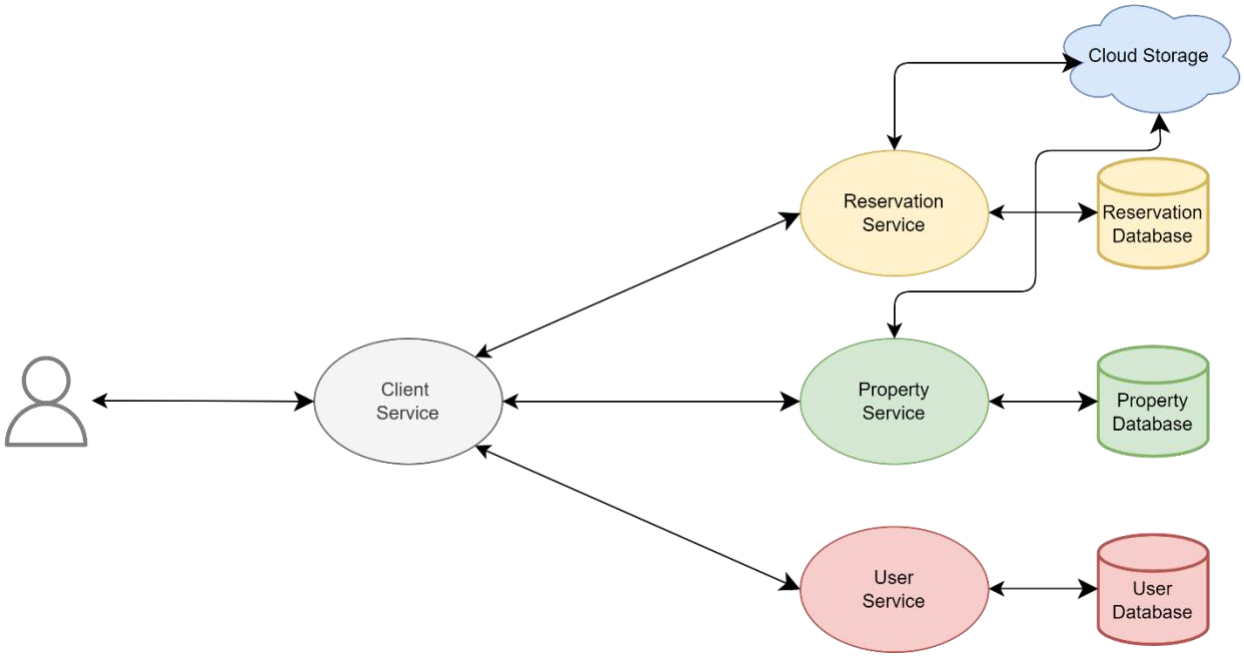
Όνομα	Μέθοδος	Περιγραφή
/users/imageUpload	POST	Ενημέρωση φωτογραφίας χρήστη.

Παράμετροι

Όνομα	Τύπος Παραμέτρου	Τύπος Δεδομένου	Απαιτείται/Προαιρετική	Περιγραφή
formData	Form Parameter	multipart/form-data	Απαιτείται	Περιέχει τη φωτογραφία
Authorization	Header Parameter	Token Authentication	Απαιτείται	Token το οποίο εμπεριέχει τα στοιχεία του χρήστη.

3.5 Διαχείριση Δεδομένων

Για τη διαχείριση των δεδομένων της εφαρμογής θα χρησιμοποιηθεί το Database-per-Service pattern. Με τη χρήση αυτού του pattern καταφέρνουμε να διατηρήσουμε το κάθε service ανεξάρτητο από τα υπόλοιπα. Κάθε service έχει ξεχωριστή βάση δεδομένων όπου μπορεί να επεκτείνεται ανεξάρτητα από τις υπόλοιπες. Επιπλέον με τη χρήση αυτού του pattern αν ένας database server πέσει, αυτό δεν θα επηρεάσει τα υπόλοιπα services. Άλλα πλεονεκτήματα που κερδίζουμε είναι ότι κάθε service μπορεί να έχει διαφορετικές τεχνολογίες βάσεων δεδομένων, αυτό είναι πολύ χρήσιμο διότι ανάλογα με το σχεδιασμό ενός service και των δεδομένων που θες να αποθηκεύσεις υπάρχει και η κατάλληλη τεχνολογία βάσεων δεδομένων που σε βοηθάει να επιτύχεις το επιθυμητό αποτέλεσμα. Τέλος οι φωτογραφίες της εφαρμογής θα αποθηκεύονται σε cloud storage.



3.6 Επικοινωνία Μεταξύ των Services

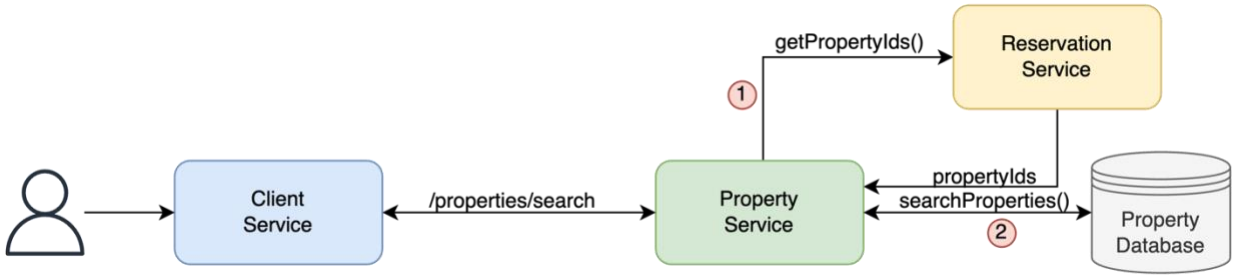
Στην αρχιτεκτονική microservices όσο περισσότερη σύγχρονη επικοινωνία μεταξύ των services υπάρχει τόσο χειρότερος γίνεται ο χρόνος απόκρισης της εφαρμογής προς τα ερωτήματα του χρήστη. Επιπλέον όταν η επικοινωνία μεταξύ των services είναι σύγχρονη τα services που παίρνουν μέρος σε αυτή την επικοινωνία χάνουν την αυτονομία τους. Γι' αυτό το λόγο στην εφαρμογή που σχεδιάζουμε χρησιμοποιούμε όπου είναι δυνατό να επιτευχθεί ασύγχρονη επικοινωνία μεταξύ των services και σύγχρονη επικοινωνία μόνο στα ερωτήματα του πελάτη. Με αυτό τον τρόπο επιτυγχάνεται μεγαλύτερη αυτονομία και ανθεκτικότητα σε σφάλματα στα services της εφαρμογής.

Παρακάτω θα δούμε σε διαγράμματα τα ερωτήματα που μπορεί να κάνει ένας χρήστης στην εφαρμογή τα οποία για να επιτευχθούν χρειάζεται η επικοινωνία πολλαπλών services. Επιπλέον θα εξηγηθεί γιατί χρειάζεται να πραγματοποιηθεί επικοινωνία με άλλα services για να εκτελεστεί το ερώτημα.

Παρακάτω θα δούμε πως το Property Service επικοινωνεί με τα υπόλοιπα services για να εξυπηρετήσει τις ανάγκες των χρηστών του.

Αναζήτηση Κατοικιών

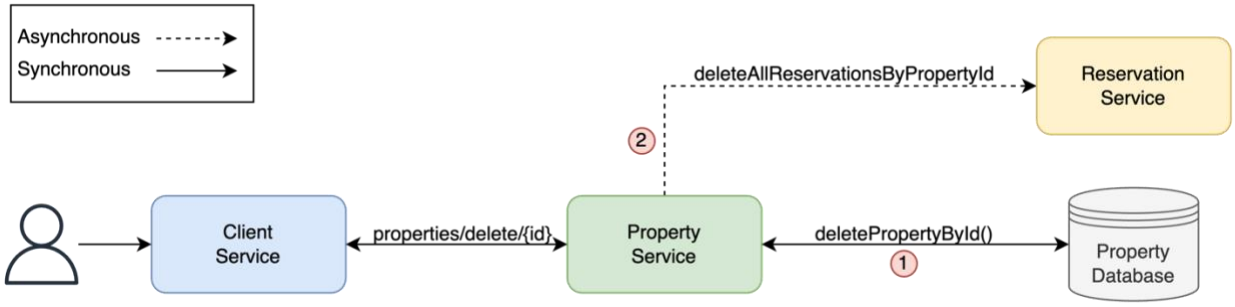
Για να επιτευχθεί το ερώτημα της αναζήτησης κατοικιών το client service επικοινωνεί με το property service το οποίο είναι υπεύθυνο για τις κατοικίες. Η αναζήτηση γίνεται με βάση την ημερομηνία όπου ο χρήστης θέλει να διαμείνει σε μία κατοικία. Συνεπώς η αναζήτηση πρέπει να επιστρέψει στο χρήστη κατοικίες οι οποίες δεν έχουν κάποια κράτηση τις συγκεκριμένες ημερομηνίες. Γι' αυτό το λόγο πραγματοποιείται επικοινωνία με το reservation service όπου με βάση τις ημερομηνίες που θέλει ο χρήστης και την τοποθεσία που θέλει να αναζητήσει για κατοικία του επιστρέφει τα IDs των κατοικιών των οποίων έχουν κλείσει άλλοι χρήστες να πάνε εκείνη την περίοδο. Έτσι το property service αποκλείει εκείνες τις κατοικίες και επιστρέφει στο χρήστη τις κατοικίες εκείνες που πληρούν τις απαιτήσεις του.



Διαγραφή Κατοικίας

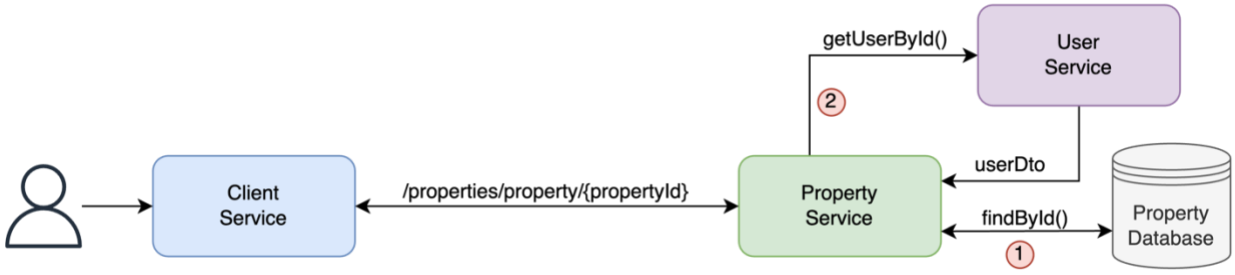
Για να επιτευχθεί το ερώτημα της διαγραφής μιας κατοικίας το client service πρέπει να επικοινωνήσει με το property service το οποίο είναι υπεύθυνο για τις κατοικίες. Όταν μια κατοικία διαγράφεται πρέπει να

διαγράφονται και οι κρατήσεις που έχουν πραγματοποιηθεί στην εκάστοτε κατοικία. Γι' αυτό το λόγο το property service επικοινωνεί με το reservation service στέλνοντας του το ID της κατοικίας που πρέπει να διαγραφτεί ώστε το reservation service να διαγράψει όλες τις κρατήσεις που έχουν γίνει στη συγκεκριμένη κατοικία.



Εύρεση Κατοικίας

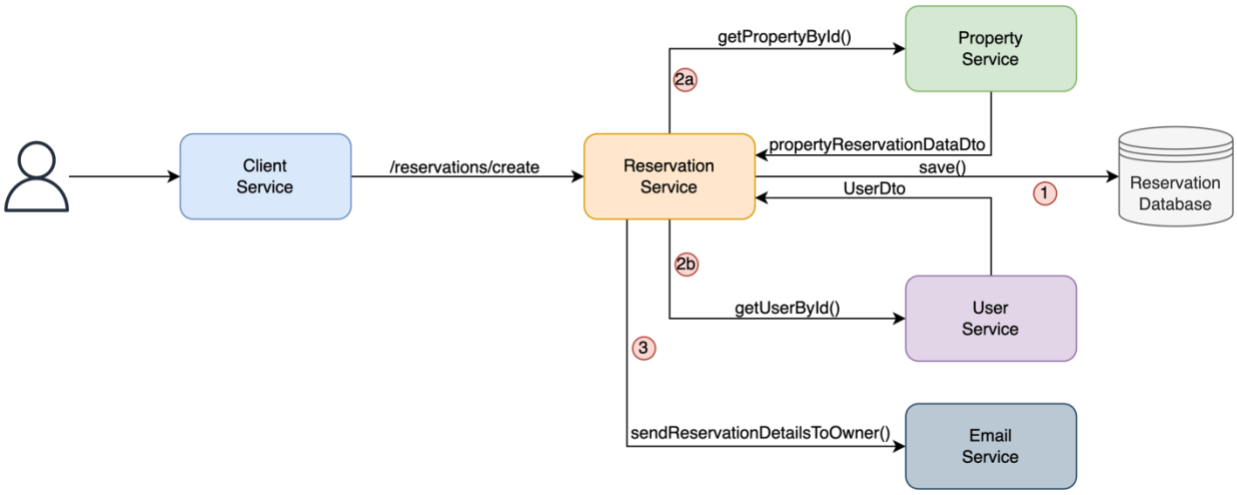
Για να επιτευχθεί το ερώτημα της εύρεσης μιας κατοικίας το client service πρέπει να επικοινωνήσει με το property service το οποίο είναι υπεύθυνο για τις κατοικίες. Η εφαρμογή όταν παρουσιάζει μια κατοικία πέρα από τις πληροφορίες που παρέχει για την κατοικία αυτή καθαυτή παρέχει και πληροφορίες για τον ιδιοκτήτη της συγκεκριμένης κατοικίας. Γι' αυτό το λόγο το property service επικοινωνεί με το user service ώστε να ανακτήσει τις απαραίτητες πληροφορίες για τον ιδιοκτήτη.



Παρακάτω θα δούμε πως το Reservation Service επικοινωνεί με τα υπόλοιπα services για να εξυπηρετήσει τις ανάγκες των χρηστών του.

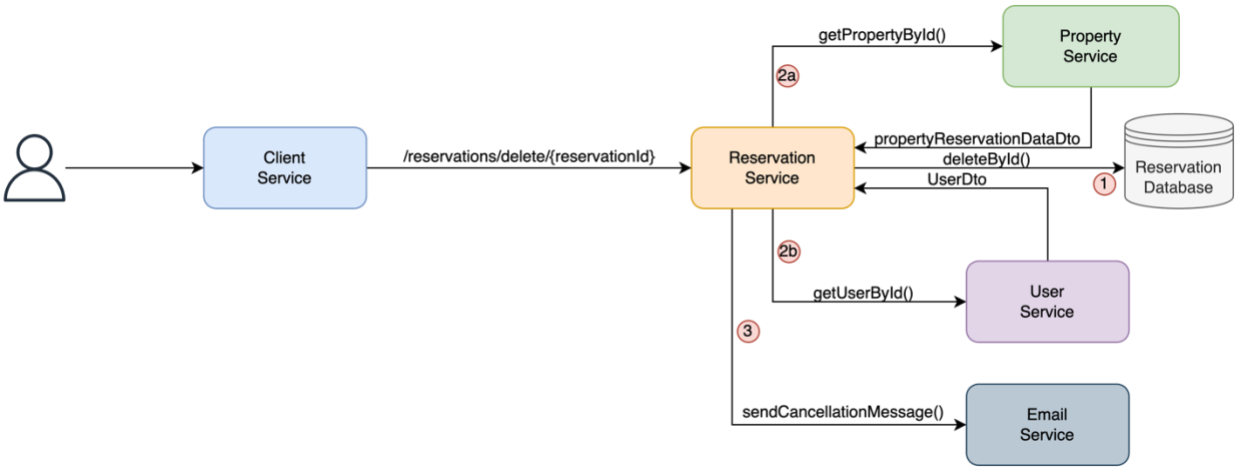
Δημιουργία Κράτησης

Για να επιτευχθεί το ερώτημα της δημιουργίας μιας κράτησης το client service πρέπει να επικοινωνήσει με το reservation service το οποίο είναι υπεύθυνο για τις κρατήσεις. Μόλις ένας χρήστης κάνει μια κράτηση τότε η εφαρμογή στέλνει αυτόματα στον ιδιοκτήτη της κατοικίας στην οποία έγινε η κράτηση mail με τις πληροφορίες του χρήστη που έκανε την κράτηση και της κατοικίας στην οποία την έκανε. Για να μπορεί να συνταχθεί αυτό το mail το reservation service πρέπει να επικοινωνήσει με το property και το user service ώστε να συλλέξει όλες τις πληροφορίες τις οποίες χρειάζεται.



Διαγραφή Κράτησης

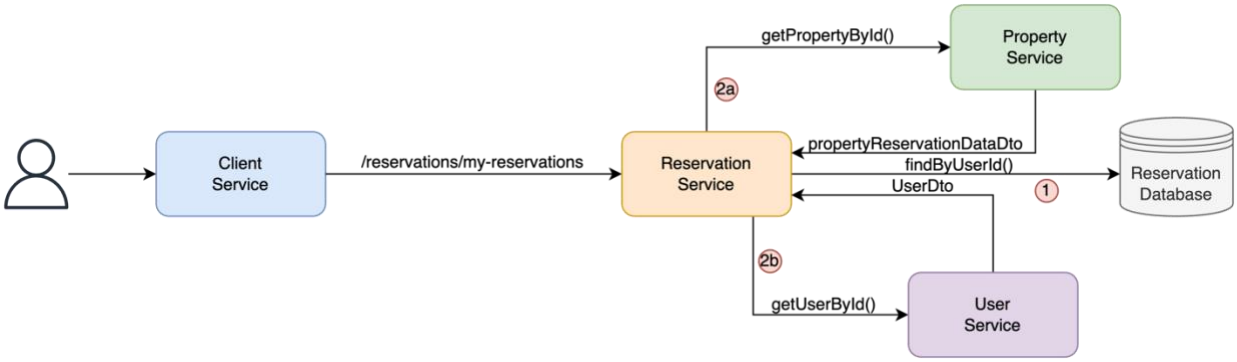
Για να επιτευχθεί το ερώτημα της διαγραφής μιας κράτησης το client service πρέπει να επικοινωνήσει με το reservation service το οποίο είναι υπεύθυνο για τις κρατήσεις. Μόλις ένας χρήστης ακυρώσει μια κράτηση τότε η εφαρμογή στέλνει αυτόματα στον ιδιοκτήτη της κατοικίας στην οποία έγινε η ακύρωση της κράτησης mail με τις πληροφορίες του χρήστη που έκανε την ακύρωση και σε ποια κατοικία έγινε η ακύρωση. Για να μπορεί να συνταχθεί αυτό το mail το reservation service πρέπει να επικοινωνήσει με το property και το user service ώστε να συλλέξει όλες τις πληροφορίες τις οποίες χρειάζεται.



Εύρεση Κρατήσεων Συγκεκριμένου Χρήστη

Για να επιτευχθεί το ερώτημα της εύρεσης των κρατήσεων ενός συγκεκριμένου χρήστη το client service πρέπει να επικοινωνήσει με το reservation service το οποίο είναι υπεύθυνο για τις κρατήσεις. Η παρουσίαση μιας κράτησης στο χρήστη της εφαρμογής πέρα από τις πληροφορίες της κράτησης παρέχει και πληροφορίες που έχουν να κάνουν με τον ιδιοκτήτη και την κατοικία στην οποία έχει γίνει η

κράτηση. Για να παρέχει το reservation service αυτές τις πληροφορίες στο χρήστη πρέπει να επικοινωνήσει με το user και το property service ώστε να συλλέξει τις απαραίτητες πληροφορίες.



3.7 Ασφάλεια

Η ασφάλεια του συστήματος επιτυγχάνεται με τα πρωτόκολλα OAuth 2.0 και OpenID. Για authorization server χρησιμοποιείται ο Keycloak

3.7.1 OAuth 2.0

Το OAuth 2.0 το οποίο προέρχεται από το “Open Authorization” είναι ένα πρότυπο σχεδιασμένο να επιτρέπει σε ένα website ή σε μία εφαρμογή να έχει πρόσβαση σε δεδομένα τα οποία φιλοξενούνται σε μια άλλη web εφαρμογή εκ μέρους ενός χρήστη. Αυτός ο μηχανισμός χρησιμοποιείται από εταιρείες σαν την Amazon, Google, Facebook, Microsoft και Twitter ώστε να επιτρέπουν στους χρήστες τους να μοιράζονται πληροφορίες από τους λογαριασμούς τους με third-party εφαρμογές ή websites. Με αυτόν τον μηχανισμό οι εφαρμογές έχουν πρόσβαση σε δεδομένα χρηστών χωρίς οι χρήστες να έχουν δώσει κάποιο κωδικό.

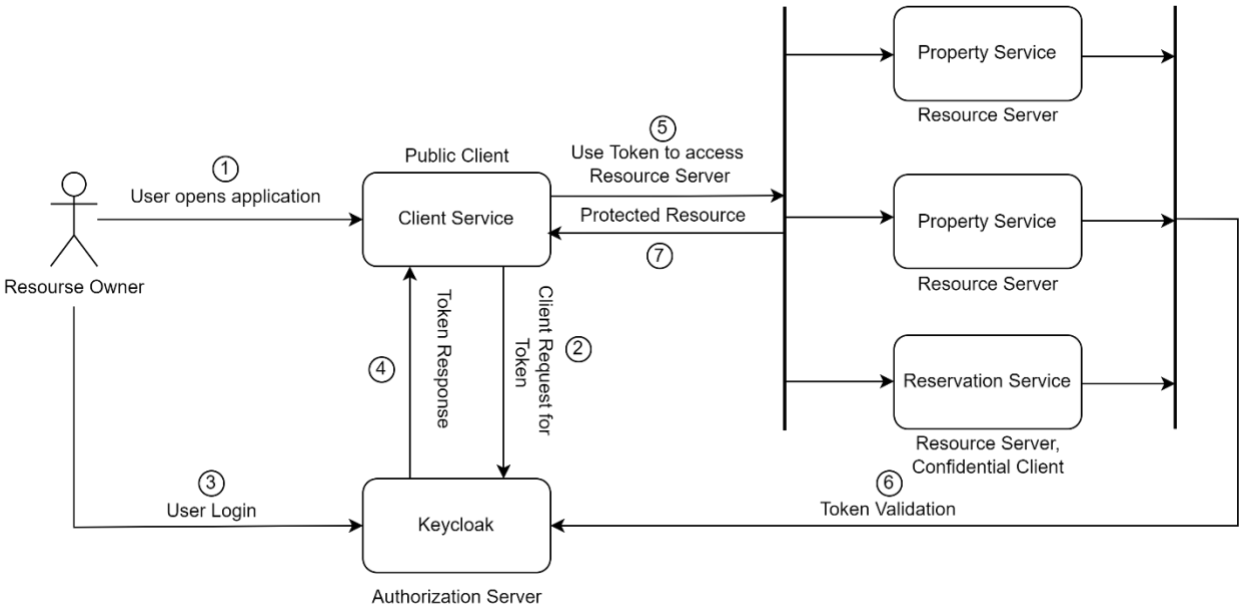
3.7.2 OAuth 2.0 Ορολογία

Για να καταλάβεις πως δουλεύει αυτό το πρωτόκολλο χρειάζεται να ξέρεις την ορολογία του.

- **Resource Owner**: Ο χρήστης ο οποίος κατέχει τα δεδομένα όπου η εφαρμογή πελάτης θέλει να έχει πρόσβαση
- **Client**: Η εφαρμογή που θέλει να έχει πρόσβαση στα δεδομένα του χρήστη
- **Authorization Server**: Ο authorization server δίνει άδεια στον client να έχει πρόσβαση στα δεδομένα του χρήστη στα οποία ο χρήστης παραχώρησε δικαιώματα.
- **Resource Server**: Το σύστημα το οποίο κατέχει τα δεδομένα τα οποία ο client θέλει να έχει πρόσβαση. Σε μερικές περιπτώσεις ο resource server και ο authorization server είναι το ίδιο.
- **Access Token**: Το access token είναι το μόνο κλειδί όπου ο client χρησιμοποιεί για να έχει πρόσβαση στα δεδομένα του χρήστη τα οποία βρίσκονται στον resource server.

3.7.3 OAuth 2.0 Flow

1. Ο χρήστης ξεκινάει το authorization flow συνήθως πατώντας ένα κουμπί για login το οποίο τον μεταφέρει στον authorization server.
2. Ο client τότε ανακατευθύνει τον χρήστη στον authorization server. Καθώς ο client ανακατευθύνει τον χρήστη στέλνει πληροφορίες στον authorization server όπως το client id και το redirect URI.
3. Ο authorization server ελέγχει την αυθεντικοποίηση του χρήστη και του παρουσιάζει μια οθόνη συγκατάθεσης ώστε ο χρήστης να παραχωρήσει δικαιώματα.
4. Αν ο χρήστης παραχωρήσει δικαιώματα ο authorization server ανακατευθύνει τον χρήστη στον client με ένα authorization key (code/token)
5. Ο client τότε είναι σε θέση να ζητήσει από τον resource server τα δεδομένα του χρήστη χρησιμοποιώντας το authorization key.
6. Ο resource server τότε επιβεβαιώνει το authorization key και απαντάει με τα δεδομένα του χρήστη που ζήτησε ο client.



Το OAuth flow της εφαρμογής

3.7.4 OpenID Connect

Το OpenID Connect είναι ένα στρώμα ταυτοποίησης πάνω από το OAuth 2.0 πρωτόκολλο. Επεκτείνει το OAuth 2.0 ώστε να τυποποιήσει έναν τρόπο για την αυθεντικοποίηση.

Το OAuth δεν παρέχει ταυτοποίηση χρήστη απευθείας αλλά παρέχει ένα access token για να αποκτήσει κάποια εξουσιοδότηση. Το OpenID Connect επιτρέπει στον client να ταυτοποιεί τον χρήστη με βάση την αυθεντικοποίηση που πραγματοποιήθηκε από τον authorization server. Αυτό επιτυγχάνεται ορίζοντας εάν scope με το όνομα openid όταν αιτείσαι από τον authorization server να γίνει σύνδεση του χρήστη. Το openid scope είναι απαραίτητο για να πεις στον authorization server ότι απαιτείται το OpenID Connect.

Το αποτέλεσμα από την αίτηση είναι ένα application code το όποιο ανταλλάζει ο client για ένα access token και ένα ID token. Το ID token είναι ένα JWT ή JSON Web Token. Ένα JWT είναι ένα κωδικοποιημένο token το οποίο αποτελείται από τρία μέρη: την κεφαλίδα, την υπογραφή, και το payload. Μετά την απόκτηση του ID token ο client μπορεί να το αποκωδικοποιήσει και να πάρει τις πληροφορίες του χρήστη οι οποίες βρίσκονται στο payload μέρος του.

4. Τεχνολογίες που Χρησιμοποιήθηκαν

4.1 Τεχνολογίες Διαχείρισης Δεδομένων στην Πλευρά του Εξυπηρετητή

4.1.1 Java

Η Java είναι μια υψηλού επιπέδου, βασισμένη σε κλάσεις, αντικειμενοστραφής γλώσσα προγραμματισμού. Είναι γενικού σκοπού γλώσσα προγραμματισμού με στόχο να επιτρέπει στους προγραμματιστές να γράφουν τον κώδικα τους μία φορά και αυτός να τρέχει παντού. Αυτό σημαίνει ότι ο μεταγλωττισμένος Java κώδικας μπορεί να τρέξει σε όλες τις πλατφόρμες που υποστηρίζουν την Java χωρίς να χρειάζεται να ξανά γίνει μεταγλώττιση του κώδικα. Η εφαρμογές σε Java είναι μεταγλωττισμένες σε bytecode όπου μπορεί να τρέξει σε οποιοδήποτε Java virtual machine ανεξάρτητα την αρχιτεκτονική του υπολογιστή. Η σύνταξη της Java είναι παρόμοια με αυτή της C και C++ αλλά έχει λιγότερες χαμηλού επιπέδου λειτουργίες από αυτές.

4.1.2 Spring Framework

Το Spring Framework είναι ένα application framework και inversion of control container για την πλατφόρμα της Java. Τα κύρια χαρακτηριστικά του μπορούν να χρησιμοποιηθούν από οποιοδήποτε Java application. Επιπλέον υπάρχουν επεκτάσεις για να χτίζεις web εφαρμογές πάνω στην πλατφόρμα Java EE (Enterprise Edition). Το Spring framework είναι ανοιχτού κώδικα.

4.1.3 Maven

Το Maven είναι ένα build automation εργαλείο το οποίο χρησιμοποιείται κυρίως σε Java projects. Το Maven φιλοξενείται από το Apache Software Foundation όπου ήταν προηγουμένως μέρος του Jakarta Project.

Το Maven αντιμετωπίζει δύο πλευρές για την κατασκευή λογισμικού: πως αυτό θα κατασκευαστεί, και τις εξαρτήσεις του.

4.2 Ασφάλεια Δεδομένων

4.2.1 Keycloak

Το keycloak είναι ένα ανοικτού κώδικα προϊών το οποίο επιτρέπει single sign-on με διαχείριση πρόσβασης και ταυτοποίησης και στοχεύει μοντέρνες εφαρμογές και υπηρεσίες. Από τον Μάρτιο του 2018 αυτό το project της κοινότητας WildFly είναι υπό τη διαχείριση της Red Hat όπου το χρησιμοποιεί σαν upstream project για το RH-SOO προϊόν τους.

4.2.2 Τεχνολογίες Παρουσίασης και Διαχείρισης Δεδομένων στην Πλευρά του Χρήστη

4.2.3 HTML

Η HTML (HyperText Markup Language) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία της αποτελούν τα βασικά δομικά στοιχεία των ιστοσελίδων. Τα στοιχεία της HTML αποτελούνται από ετικέτες (tags) οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <body>). Οι ετικέτες στην HTML τις περισσότερες φορές λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>) με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης. Ανάμεσα στις ετικέτες οι σχεδιαστές μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

4.2.4 CSS

Η CSS (Cascading Style Sheets - διαδοχικά φύλλα ύφους ή επάλληλα φύλλα ύφους) είναι μία γλώσσα που ανήκει στην κατηγορία των γλωσσών φύλλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στη γλώσσα HTML. Η CSS είναι μια γλώσσα προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα.

4.2.5 Sass

Το Sass (Syntactically awesome style sheets) είναι ένας προεπεξεργαστής γλώσσας σεναρίων όπου μεταφράζεται σε CSS (Cascading Style Sheets). Το SassScript είναι μια γλώσσα σεναρίων από μόνη της.

Το Sass αποτελείται από δύο συντακτικά. Το πρωτότυπο συντακτικό το οποίο ονομάζεται “the indented syntax” όπου χρησιμοποιεί ένα συντακτικό όμοι με το Haml. Χρησιμοποιεί εσοχές για να ξεχωρίσει τα μπλοκ κώδικα και newline χαρακτήρες για να ξεχωρίσει τους κανόνες. Το καινούριο συντακτικό, “SCSS (Sassy CSS) χρησιμοποιεί μορφοποίηση σε μπλοκ σαν το CSS. Χρησιμοποιεί άγκιστρα για να δηλώσει μπλοκ κώδικα και ερωτηματικά για να ξεχωρίσει τους κανόνες μέσα στα μπλοκ.

4.2.6 JavaScript

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φύλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme.

4.2.7 React

Η React (γνωστή και ως React.js ή ReactJS) είναι μια δωρεάν και ανοικτού κώδικα front-end βιβλιοθήκη της JavaScript για την κατασκευή διεπαφών χρήστη βασισμένη σε UI συνιστώσες. Συντηρείται από την Meta (πρώην Facebook) και από μια κοινότητα ανεξάρτητων προγραμματιστών και εταιριών. Η React μπορεί να χρησιμοποιηθεί σαν βάση στην ανάπτυξη μιας single-page εφαρμογής.

4.3 Βάσεις Δεδομένων

4.3.1 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα (RDBMS). Το όνομα της είναι συνδυασμός του “My”, από το όνομα του συνιδρυτή της κόρης του Michael Widenius και το SQL η σύντομη εκδοχή του Structured Query Language. Μια σχεσιακή βάση δεδομένων οργανώνει τα δεδομένα της σε έναν η περισσότερους πίνακες όπου οι τύποι δεδομένων τους μπορεί να σχετίζονται ο ένας με τον άλλο. Αυτές οι συσχετίσεις βοηθούν στη δομή των δεδομένων. Η SQL είναι μια γλώσσα που χρησιμοποιούν οι προγραμματιστές για να δημιουργήσουν, τροποποιήσουν, και εξάγουν δεδομένα από μία σχεσιακή βάση δεδομένων. Επίσης χρησιμοποιείται για τον έλεγχο της πρόσβασης του χρήστη στη βάση δεδομένων.

4.3.2 MongoDB

Η MongoDB είναι μια ανοιχτού κώδικα NoSQL βάση δεδομένων. Η NoSQL χρησιμοποιείται σαν μια εναλλακτική τη παραδοσιακής σχεσιακής βάσης δεδομένων. Οι NoSQL βάσεις δεδομένων είναι πολύ χρήσιμες όταν δουλεύεις με πολύ μεγάλα κατανεμημένα δεδομένα. Η MongoDB είναι ένα εργαλείο το οποίο μπορεί να διαχειριστεί πληροφορίες προσανατολισμένες σε έγγραφα.

4.4 Τεχνολογίες Εικονικοποίησης σε Επίπεδο Λειτουργικού Συστήματος

4.4.1 Docker

Το Docker είναι μια containerization πλατφόρμα ανοιχτού κώδικα. Επιτρέπει στους προγραμματιστές να πακετάρουν τις εφαρμογές τους σε containers (standardized executable components) που συνδυάζουν τον κώδικα της εφαρμογής με το λειτουργικό σύστημα και τις εξαρτήσεις που είναι απαραίτητες για να τρέξει η εφαρμογή σε οποιοδήποτε περιβάλλον. Οι containers απλοποιούν τη διανομή κατανεμημένων εφαρμογών, και η δημοφιλία τους αυξάνεται όσο οι οργανισμοί τείνουν προς την ανάπτυξη cloud native εφαρμογών.

Οι προγραμματιστές μπορούν να δημιουργήσουν containers χωρίς το Docker, παρόλα αυτά η πλατφόρμα κάνει τη δημιουργία τους ευκολότερη, πιο απλή, και ασφαλέστερη στο να κατασκευαστή, διανεμηθεί και διαχειριστεί ένας container. Το docker είναι βασικά μια εργαλειοθήκη που επιτρέπει στους προγραμματιστές να χτίζουν, διανέμουν, εκτελούν, ενημερώνουν, και σταματούν containers χρησιμοποιώντας απλές εντολές.

4.4.2 Docker Desktop

Το Docker Desktop είναι ένα εύκολο στην εγκατάσταση πρόγραμμα το οποίο είναι για Mac και Windows περιβάλλοντα όπου σου επιτρέπει να χτίσεις και να μοιραστείς containerized εφαρμογές και microservices. Το Docker Desktop περιέχει την Docker Engine, το Docker CLI client, Docker Compose, Docker Content Trust, Kubernetes και Credential Helper.

Το Docker Desktop δουλεύει με τα εργαλεία και τις γλώσσες που εσύ έχεις επιλέξει και σου δίνει πρόσβαση σε μια τεράστια βιβλιοθήκη από πιστοποιημένα images και templates στο Docker Hub. Αυτό επιτρέπει στις ομάδες ανάπτυξης να επεκτείνουν το περιβάλλον τους ταχύτατα σε αυτόματη κατασκευή και continuously integrate και να υπάρχει συνεργασία χρησιμοποιώντας έναν ασφαλή αποθηκευτικό χώρο.

4.5 Container Orchestration

4.5.1 Kubernetes

Το Kubernetes είναι ένα ανοιχτού κώδικα container orchestration σύστημα για την αυτοματοποίηση της ανάπτυξης λογισμικού, της επέκτασης και της διαχείρισης. Ο σχεδιασμός του Kubernetes πραγματοποιήθηκε από την Google, αλλά τώρα συντηρείται από το Cloud Native Computing Foundation.

Το Kubernetes δουλεύει με το Docker, το Containerd, και το CRI-O. Αρχικά η διεπαφή του δούλευε αποκλειστικά με το Docker μέσα από το “Dockershim” παρόλα αυτά από το 2016 το Kubernetes έχει δυσφημίση το shim για χάρη της απευθείας διεπαφής του με το container μέσα από το Containerd,

4.6 Τεχνολογίες Message Broker

4.6.1 RabbitMQ

Το RabbitMQ είναι ένας από τους πιο δημοφιλής ανοιχτού κώδικα message broker με δεκάδες χιλιάδες χρήστες. Από την T-Mobile έως την Runtastic το RabbitMQ χρησιμοποιείται παγκοσμίως από μικρές εταιρίες έως τεράστιες πολυεθνικές.

Το RabbitMQ είναι ελαφρύ και εύκολο να το χρησιμοποιήσεις στις εγκαταστάσεις σου ή στο νέφος. Επιπλέον υποστηρίζει πολλαπλά messaging πρωτόκολλα. Το RabbitMQ μπορεί να χρησιμοποιηθεί με κατανεμημένες ή ενοποιημένες ρυθμίσεις ώστε να πετύχει high-scale, και high-availability απαιτήσεις.

4.7 Εργαλεία Ανάπτυξης

4.7.1 Visual Studio Code

Το Visual Studio Code είναι ένα πρόγραμμα επεξεργασίας κώδικα το οποίο το δημιούργησε η Microsoft για τα Windows, linux και macOS. Τα χαρακτηριστικά που υποστηρίζει είναι debugging, syntax highlighting, intelligent code completion, snippets, code regactoring, και έχει ενσωματωμένο το Git. Οι χρήστες του μπορούν εγκαταστήσουν επεκτάσεις οι οποίες προσφέρουν επιπλέον λειτουργίες.

Για την ανάπτυξη της εφαρμογής το Visual Studio Code χρησιμοποιήθηκε για την δημιουργία της διεπαφής χρήστη (front-end).

4.7.2 IntelliJ IDEA

Το IntelliJ IDEA είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) το οποίο έχει γραφτεί σε Java και χρησιμοποιείται για την ανάπτυξη λογισμικού. Αναπτύχθηκε από την JetBrains (πρώην γνωστή και ως IntelliJ), και οι διαθέσιμες εκδώσεις του είναι η community έκδοση και η commercial έκδοση. Και οι δύο εκδώσεις χρησιμοποιούνται για εμπορική ανάπτυξη.

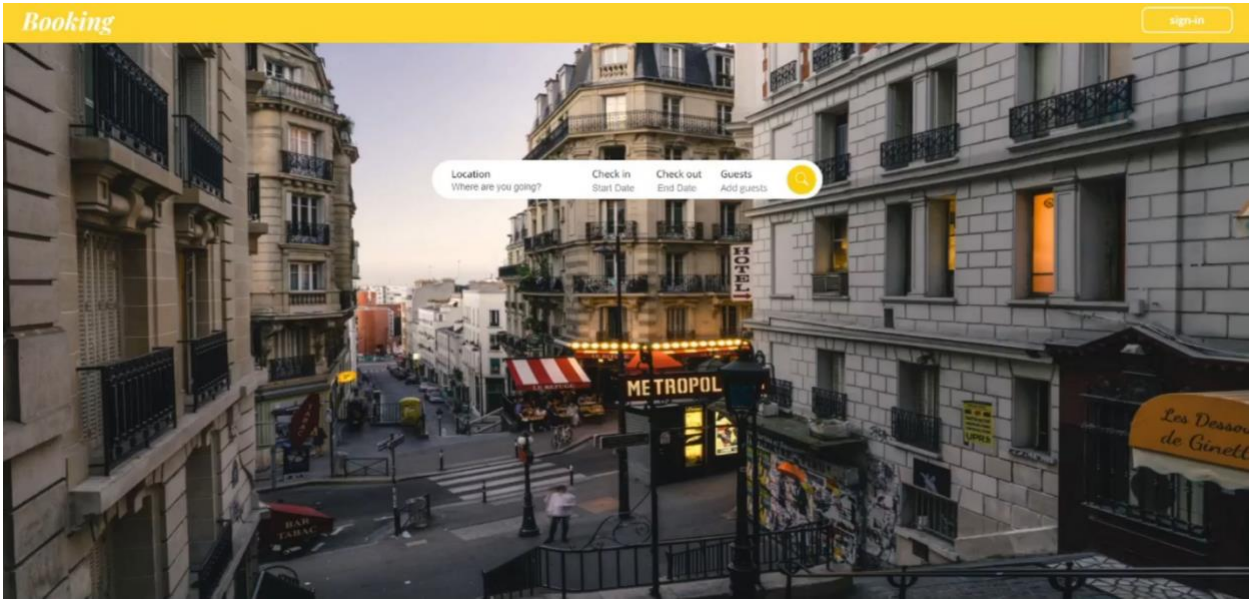
Για την ανάπτυξη της εφαρμογής το IntelliJ IDEA χρησιμοποιήθηκε για την σύνταξη κώδικα για τη διαχείριση των δεδομένων.

5. Υλοποίηση

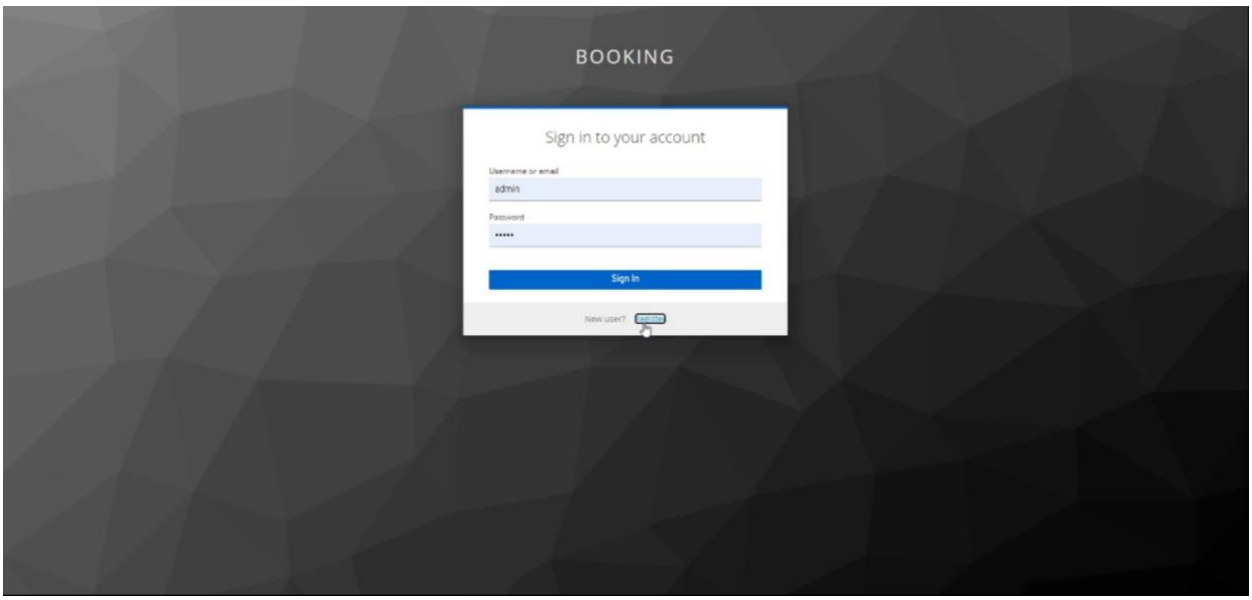
5.1 Αναπαράσταση Λειτουργικότητας Εφαρμογής

5.1.1 Δημιουργία Χρήστη

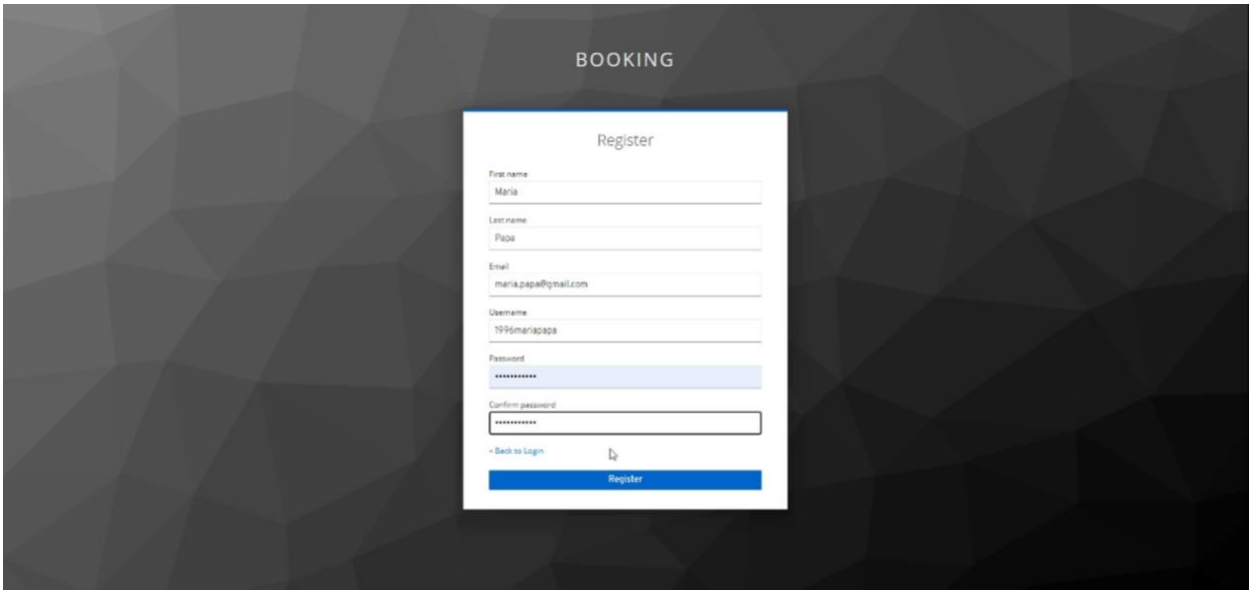
Για να δημιουργήσουμε ένα χρήστη το πρώτο βήμα που κάνουμε είναι να πατήσουμε στο κουμπί sign-in πάνω δεξιά στην οθόνη.



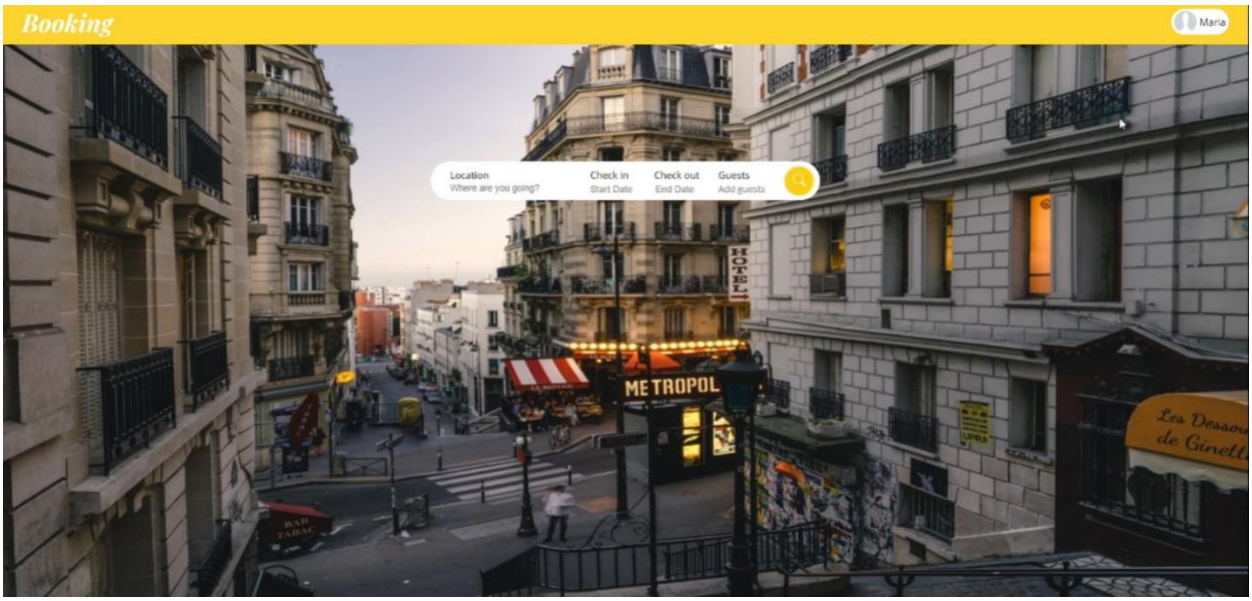
Το κουμπί αυτό μας μεταφέρει σε μια σελίδα στην οποία υπάρχει ένα κουμπί register το όποιο πρέπει να πατήσουμε για να δημιουργήσουμε έναν νέο χρήστη.



Το κουμπί register μας μεταφέρει σε μια φόρμα στην οποία θα εισάγουμε τα στοιχεία του χρήστη τα οποία είναι απαραίτητα για τη δημιουργία του. Μόλις συμπληρώσουμε κατάλληλα όλα τα παιδία πατάμε το κουμπί Register για τη δημιουργία ενός νέου χρήστη.

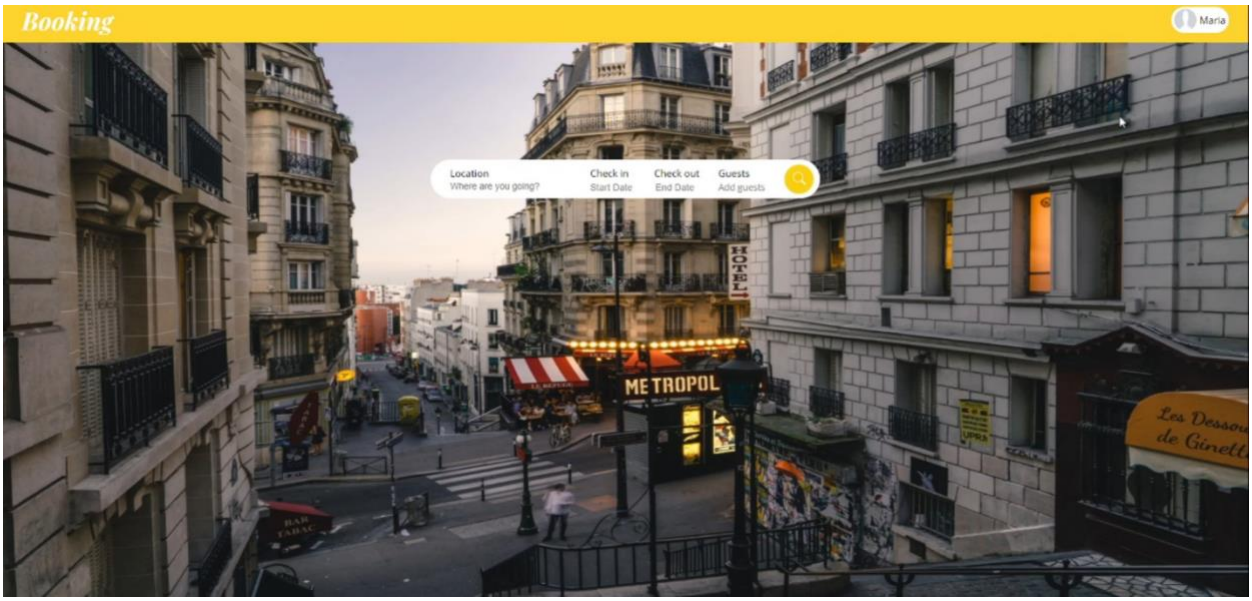


Έπειτα το κουμπί Register μας μεταφέρει στην αρχική σελίδα της εφαρμογής όπου πάνω αριστερά βλέπουμε να έχει φύγει το κουμπί sign-in και τη θέση του έχει πάρει ένα νέο κουμπί το οποί περιέχει πληροφορίες για το χρήστη που είναι συνδεδεμένος.

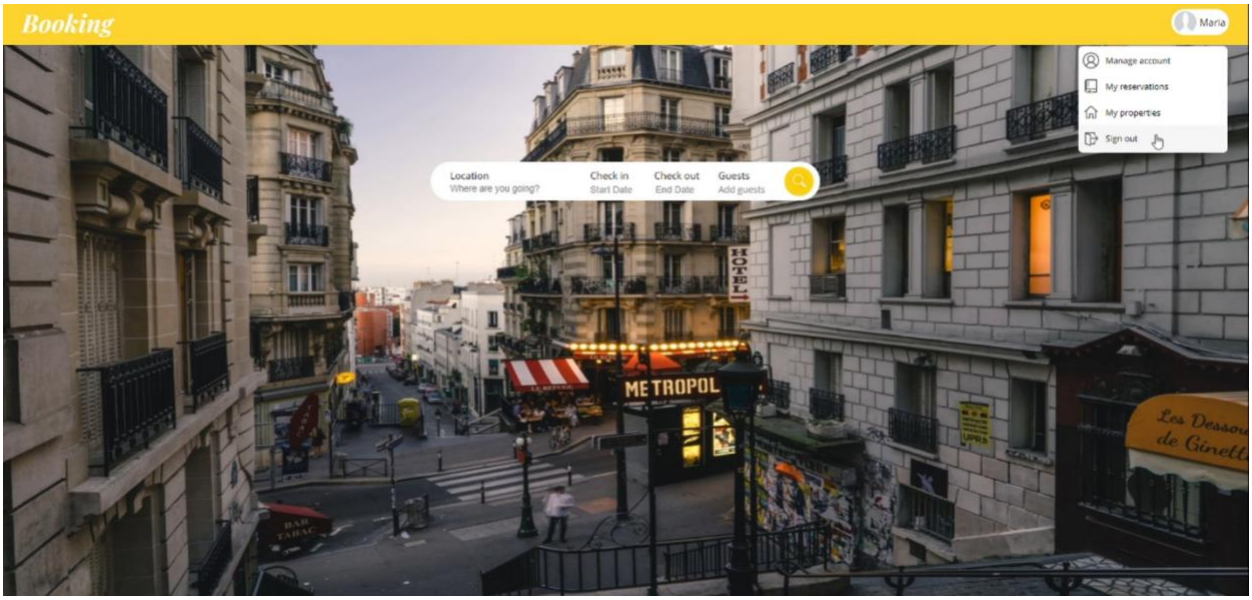


5.1.2 Είσοδος και Έξοδος χρήστη

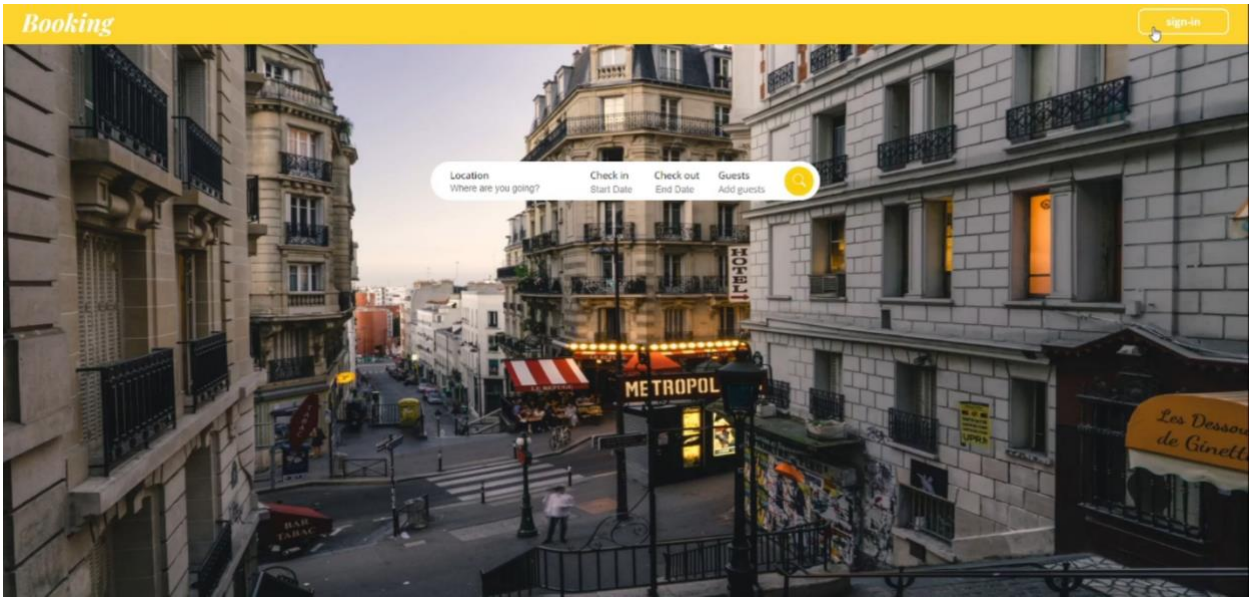
Για να κάνουμε αποσύνδεση ενός χρήστη από την εφαρμογή πατάμε το πάνω αριστερά κουμπί όπου αναγράφει το όνομα του.



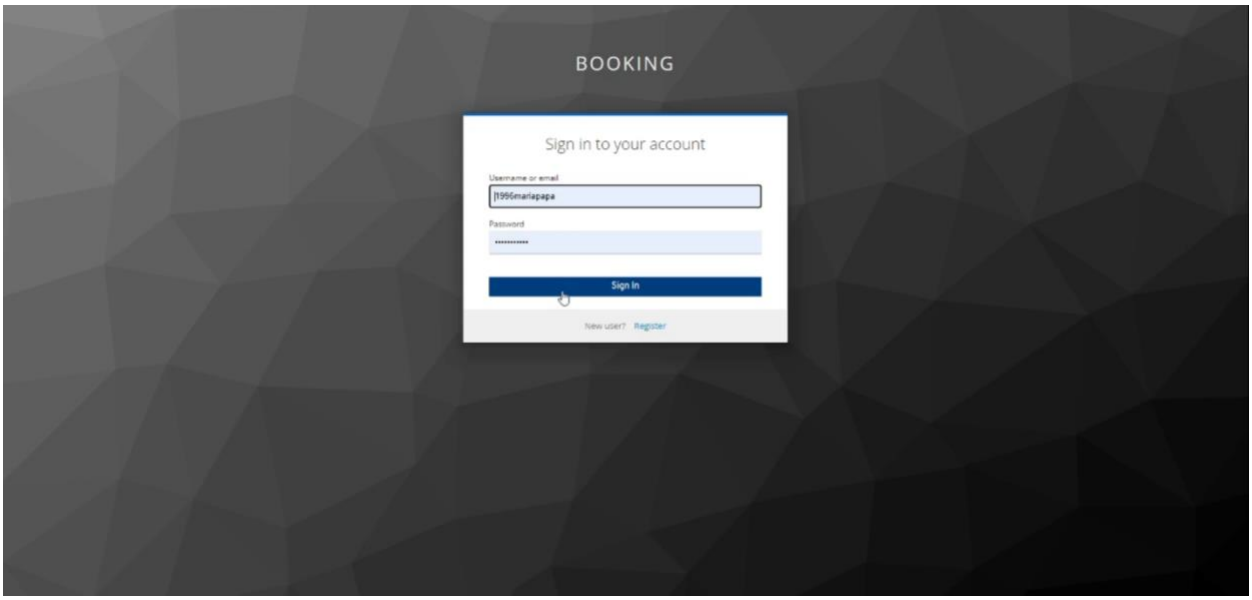
Όταν πατήσουμε το κουμπί εμφανίζεται ένα μενού, από το μενού επιλέγουμε το Sign out για την αποσύνδεση του χρήστη.



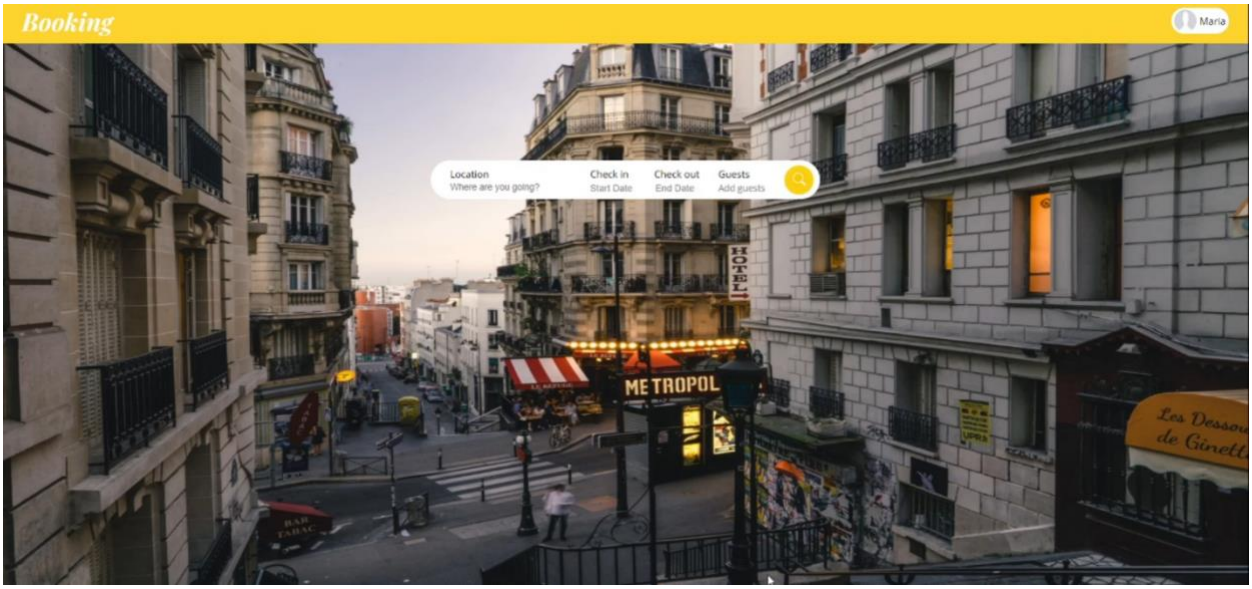
Μόλις πατήσουμε το Sign out η εφαρμογή κάνει αποσύνδεση του χρήστη και μας μεταφέρει στην αρχική σελίδα. Αν θέλουμε να πραγματοποιήσουμε σύνδεση του χρήστη τότε πατάμε το κουμπί sign in που βρίσκεται πάνω αριστερά.



Μόλις πατήσουμε το κουμπί sign-in μας μεταφέρει σε μια φόρμα η οποία μας ζητάει email είτε username και κωδικό πρόσβασης. Όταν πληκτρολογήσουμε κατάλληλα τα στοιχεία πατάμε το κουμπί Sign in για να πραγματοποιηθεί η σύνδεση του χρήστη.

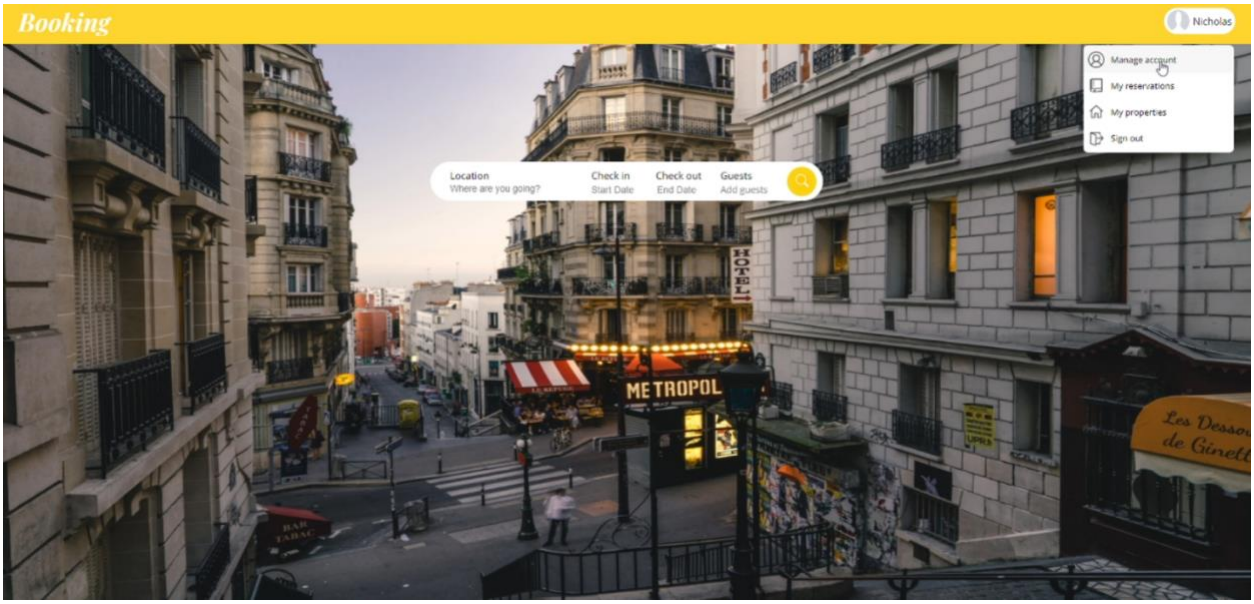


Αν τα στοιχεία που εισαγάγαμε είναι σωστά η εφαρμογή μας επιστρέφει στην αρχική σελίδα και ο χρήστης είναι πλέον συνδεδεμένος. Παρατηρούμε πάνω αριστερά ότι δεν υπάρχει το κουμπί sign-in αλλά ένα κουμπί το οποίο αναγράφει το όνομα του χρήστη που έγινε η σύνδεση.

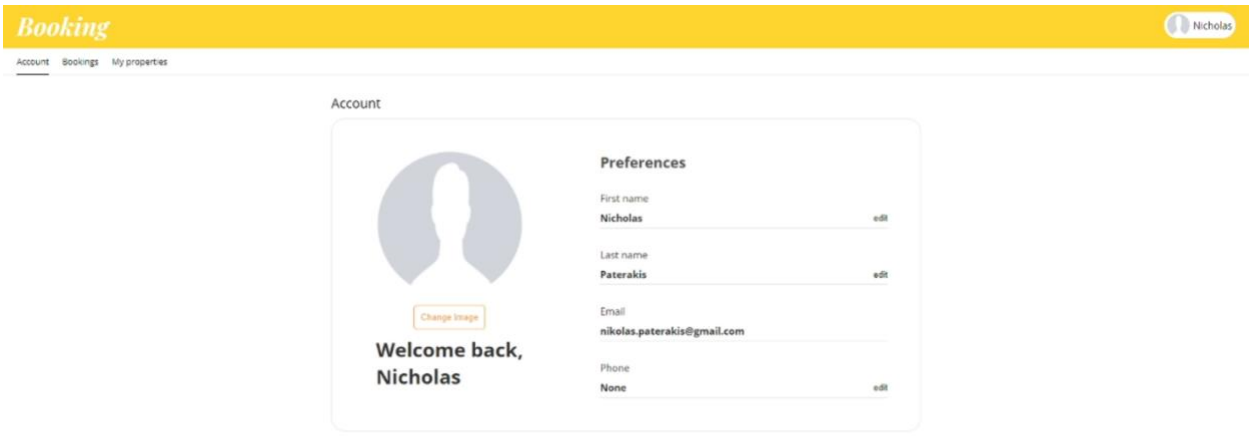


5.1.3 Τροποποίηση Χρήστη

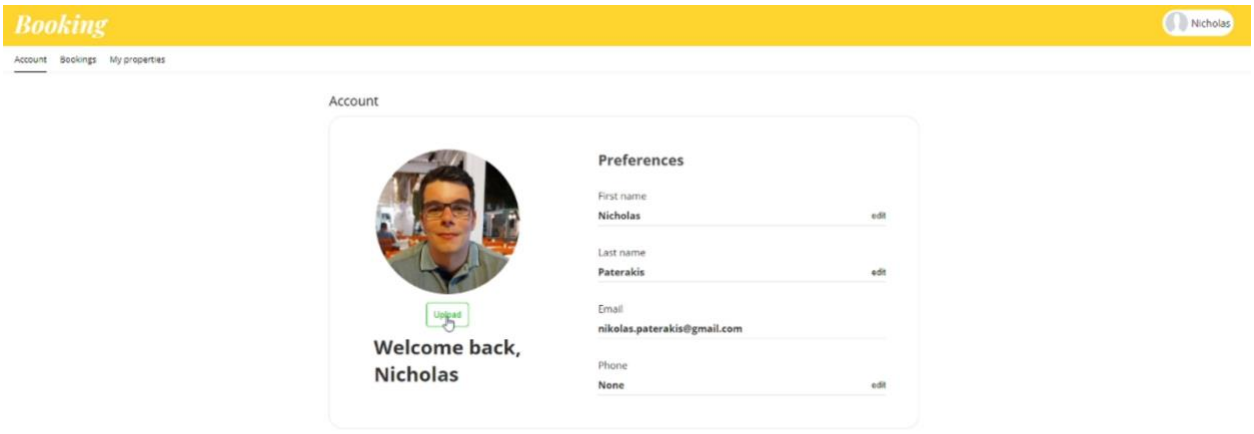
Για να τροποποιήσουμε ένα χρήστη ενώ έχουμε συνδεθεί στην εφαρμογή πατάμε το πάνω αριστερά κουμπί που αναγράφει το όνομα χρήστη και έπειτα την πρώτη επιλογή του μενού που εμφανίζεται Manage account.



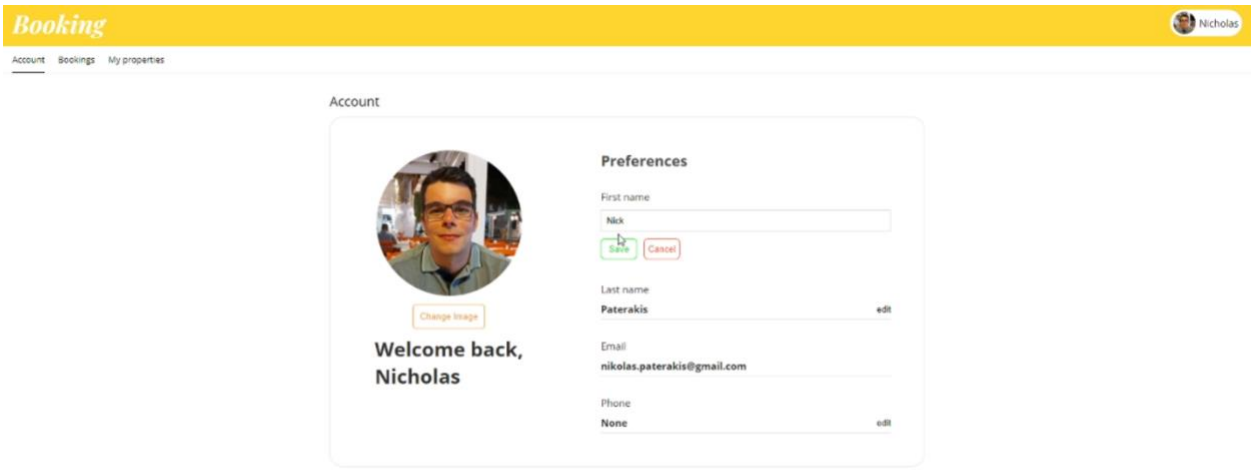
Η επιλογή Manage account μας πηγαίνει σε μια σελίδα στην οποία αναγράφονται τα στοιχεία του χρήστη. Αν θέλουμε να προσθέσουμε μια φωτογραφία στο χρήστη μας πατάμε στο κουμπί change image κάτω από τη φωτογραφία.



Μόλις πατήσουμε το κουμπί μας εμφανίζεται ο file manager του υπολογιστή μας όπου τον χρησιμοποιούμε για να βρούμε ποια φωτογραφία θέλουμε να ανεβάσουμε. Όταν βρούμε τη φωτογραφία την επιλέγουμε και αυτή εμφανίζεται στον κύκλο. Αν θέλουμε αυτή τη φωτογραφία πατάμε το κουμπί upload το οποίο αποθηκεύει τη φωτογραφία στην εφαρμογή.

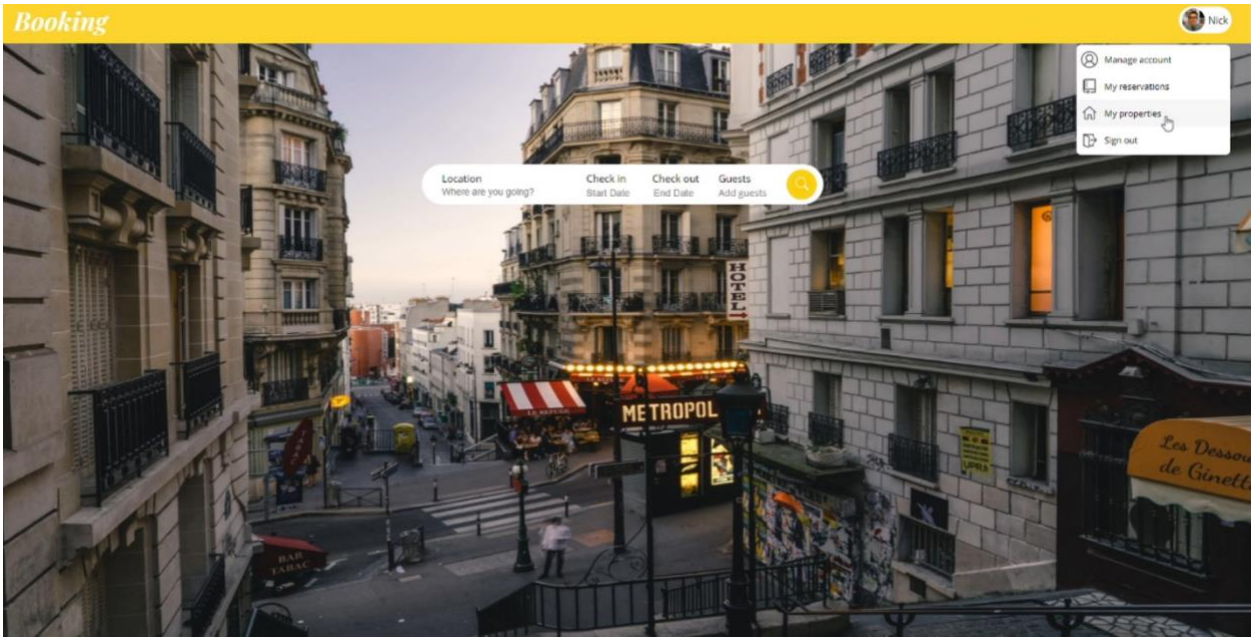


Αν θέλουμε να τροποποιήσουμε κάποιο στοιχείο του χρήστη πατάμε στο κουμπί edit το οποίο βρίσκεται στα αριστερά του κάθε στοιχείου πλην του στοιχείου email το οποίο δεν μπορεί να τροποποιηθεί. Μόλις πατήσουμε το κουμπί εμφανίζεται κατάλληλη φόρμα για την τροποποίηση του στοιχείου που επιλέξαμε. Όταν εισάγουμε την τιμή που θέλουμε στο πεδίο πατάμε save για να γίνει η τροποποίηση στο στοιχείο ή αλλιώς cancel για να ακυρώσουμε την τροποποίηση.

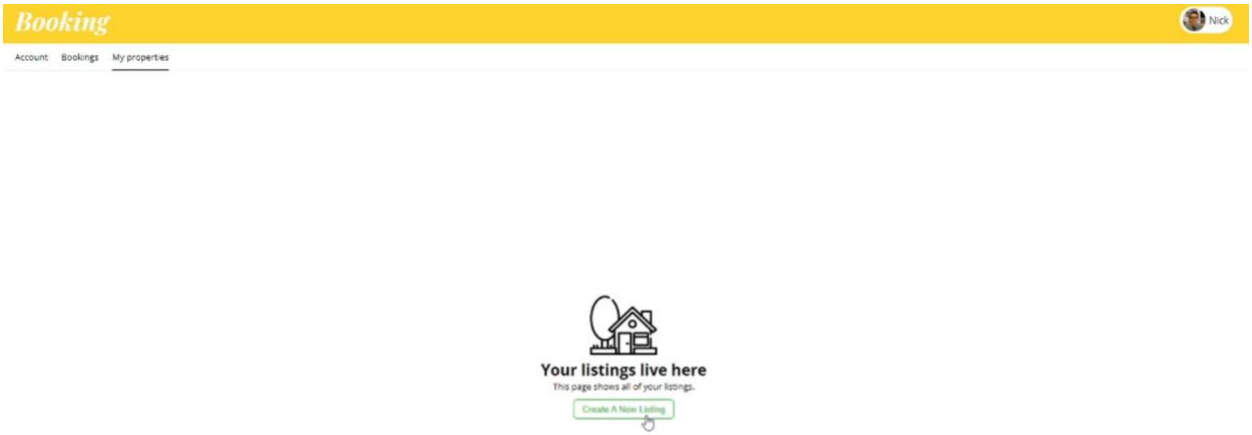


5.1.4 Δημιουργία Αγγελίας

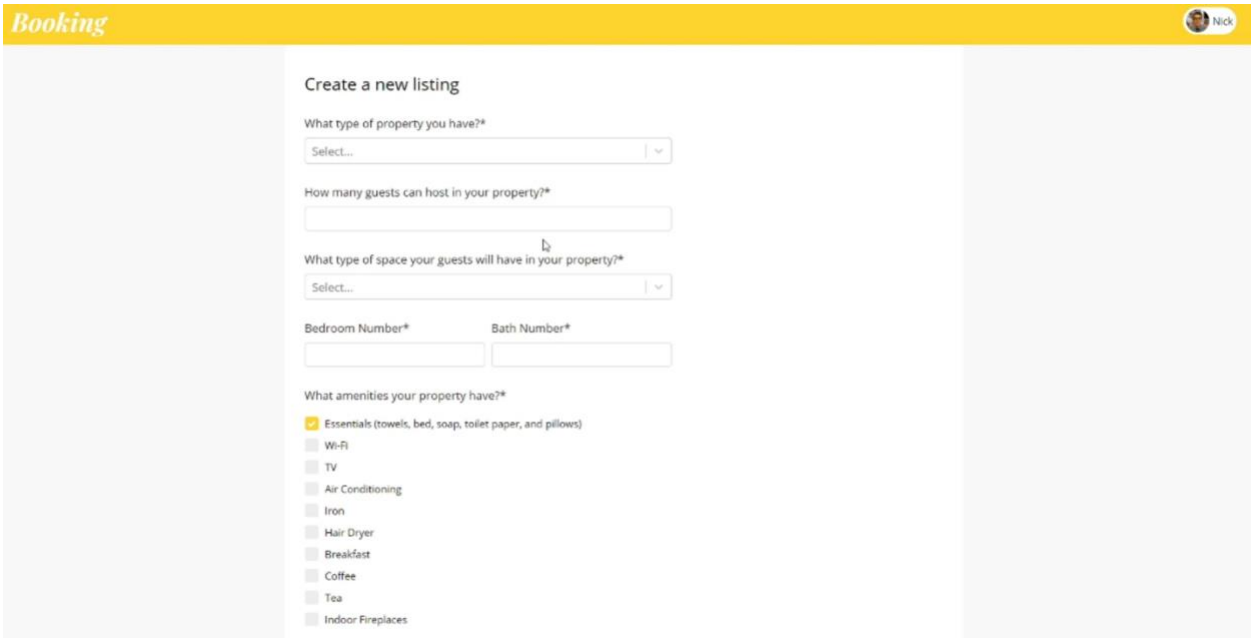
Για να δημιουργήσεις μια νέα αγγελία κλικάρεις το κουμπί που αναγράφεται το όνομα του χρήστη πάνω δεξιά και έπειτα από το μενού που εμφανίζεται επιλέγεις το My properties.

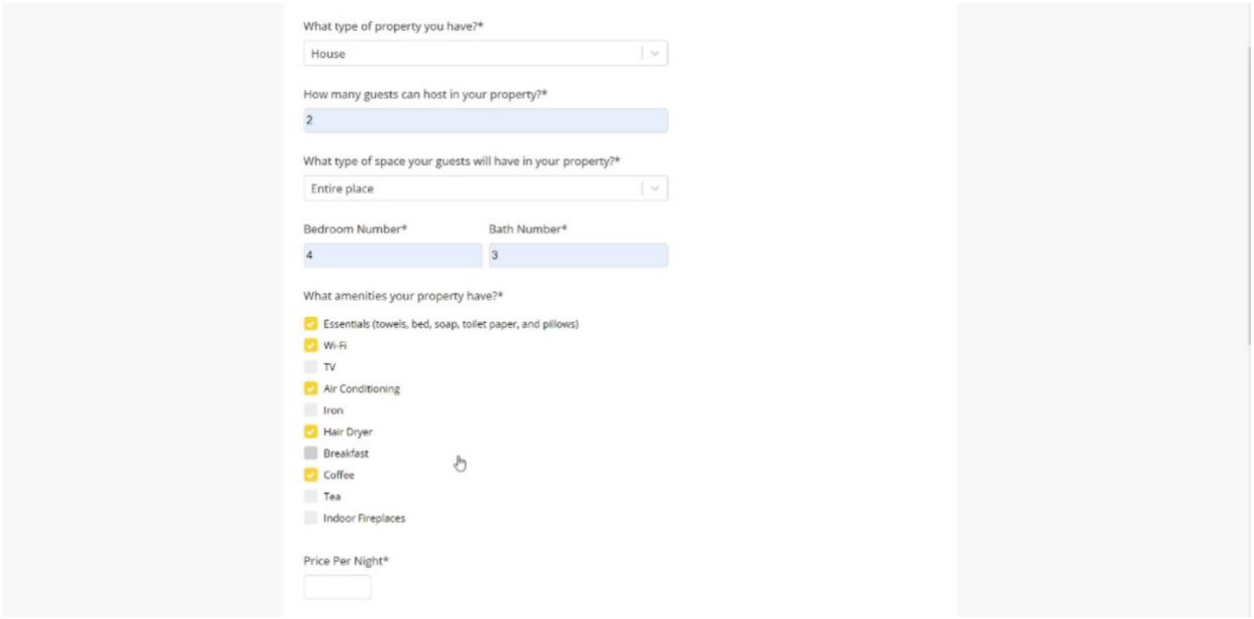


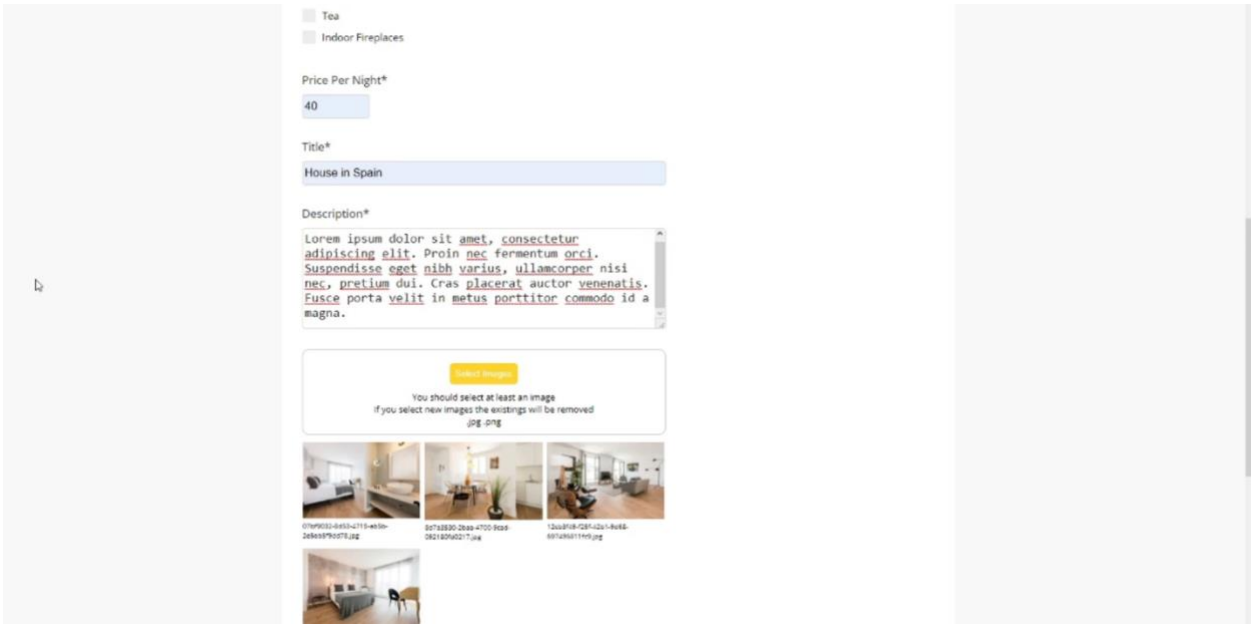
Μόλις επιλέξεις το My properties η εφαρμογή σε μεταφέρει σε μια σελίδα στην οποία μπορείς να διαχειριστείς τις κατοικίες σου. Αν θες να δημιουργήσεις μια νέα αγγελία πατάς το κουμπί Create a new listing.

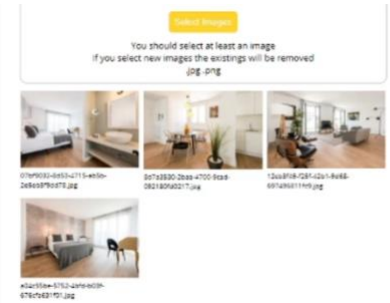


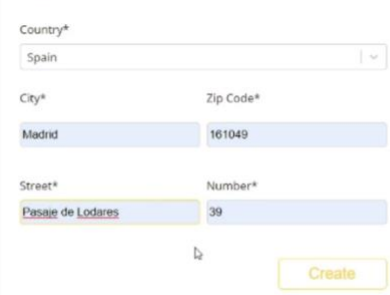
Το κουμπί αυτό σε μεταφέρει σε μια φόρμα στην οποία συμπληρώνεις τα στοιχεία που απαιτούνται για να δημιουργηθεί η αγγελία σου. Σε περίπτωση κάποιου λάθους στην εισαγωγή των στοιχείων η φόρμα σε ενημερώνει με κατάλληλο μήνυμα. Μόλις συμπληρώσεις σωστά όλη τη φόρμα η εφαρμογή σου επιτρέπει να πατήσεις το κουμπί για τη δημιουργία της.

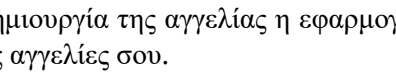



What type of property you have?*
House
How many guests can host in your property?*
2
What type of space your guests will have in your property?*
Entire place
Bedroom Number*
4
Bath Number*
3
What amenities your property have?*
Essentials (towels, bed, soap, toilet paper, and pillows)
Wi-Fi
TV
Air Conditioning
Iron
Hair Dryer
Breakfast
Coffee
Tea
Indoor Fireplaces
Price Per Night*


Tea
Indoor Fireplaces
Price Per Night*
40
Title*
House in Spain
Description*
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin nec fermentum orci. Suspendisse eget nibh varius, ullamcorper nisi nec, pretium dui. Cras placerat auctor venenatis. Fusce porta velit in metus porttitor commodo id a magna.
Select Images
You should select at least an image
If you select new images the existings will be removed
jpg ,png



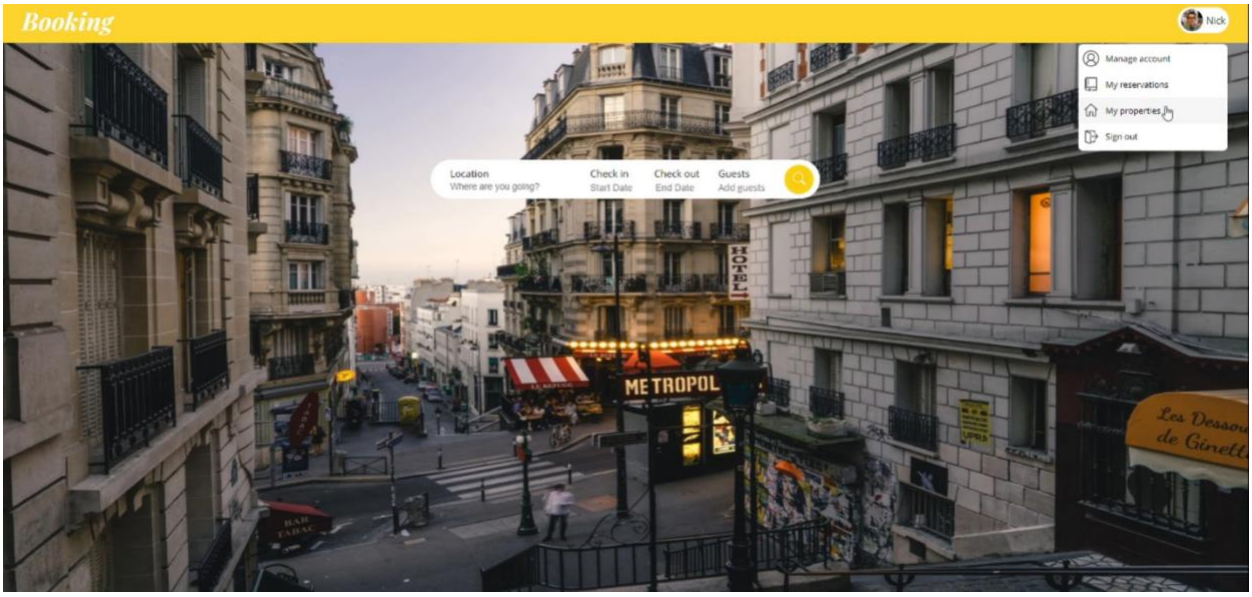




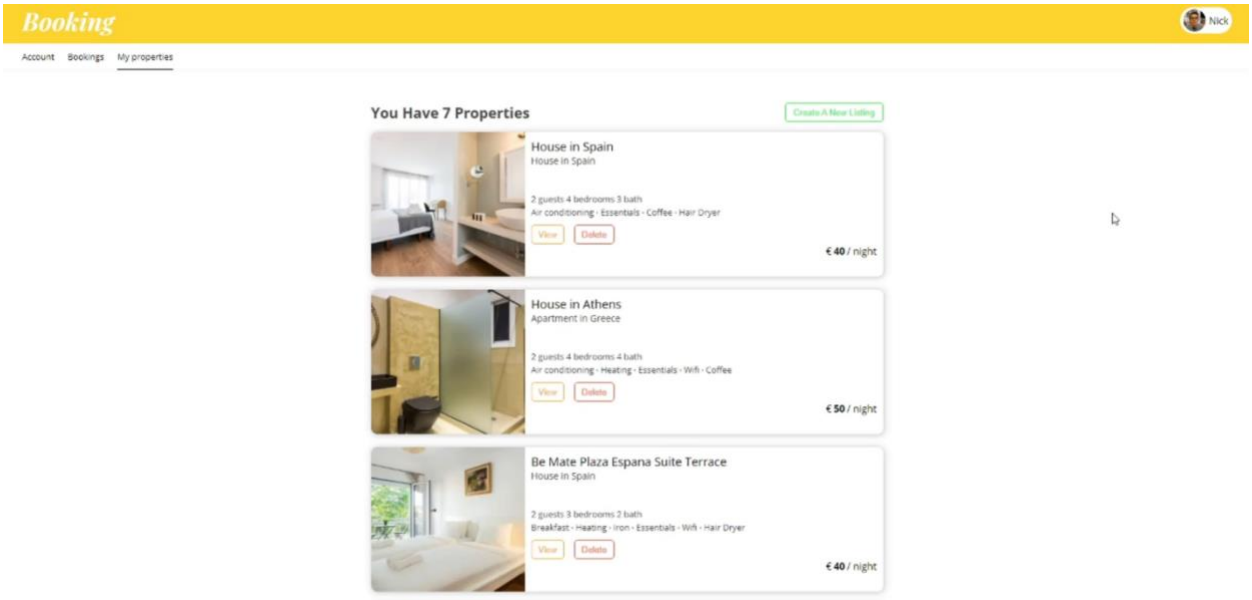
Όταν ολοκληρωθεί η δημιουργία της αγγελίας η εφαρμογή σε πηγαίνει στην σελίδα με τις κατοικίες σου όπου μπορείς να δεις τις αγγελίες σου.

5.1.5 Ανασκόπηση Αγγελίας

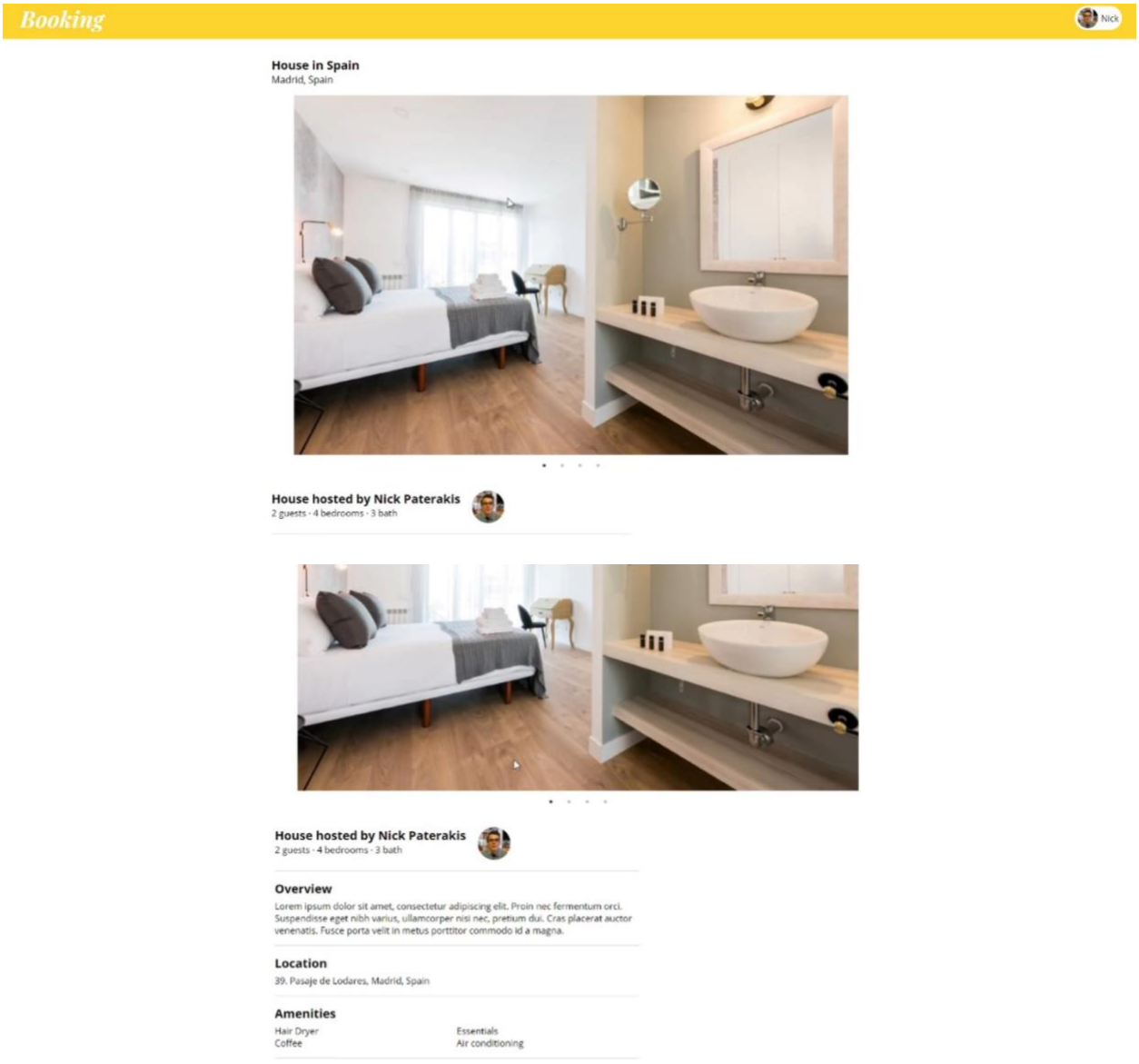
Για να δεις μια αγγελία την οποία έχεις δημοσιεύσει κλικάρεις το κουμπί που αναγράφεται το όνομα χρήστη πάνω δεξιά και επιλέγεις My properties από το menu που εμφανίζεται.



Στη σελίδα που εμφανίζεται μπροστά αναγράφονται με κατάλληλο τρόπο όλες οι αγγελίες που έχεις δημοσιεύσει. Για να δεις την αγγελία με περισσότερες λεπτομέρειες πατάς το κίτρινο κουμπί View που βρίσκεται πάνω στην καρτέλα της αγγελίας.

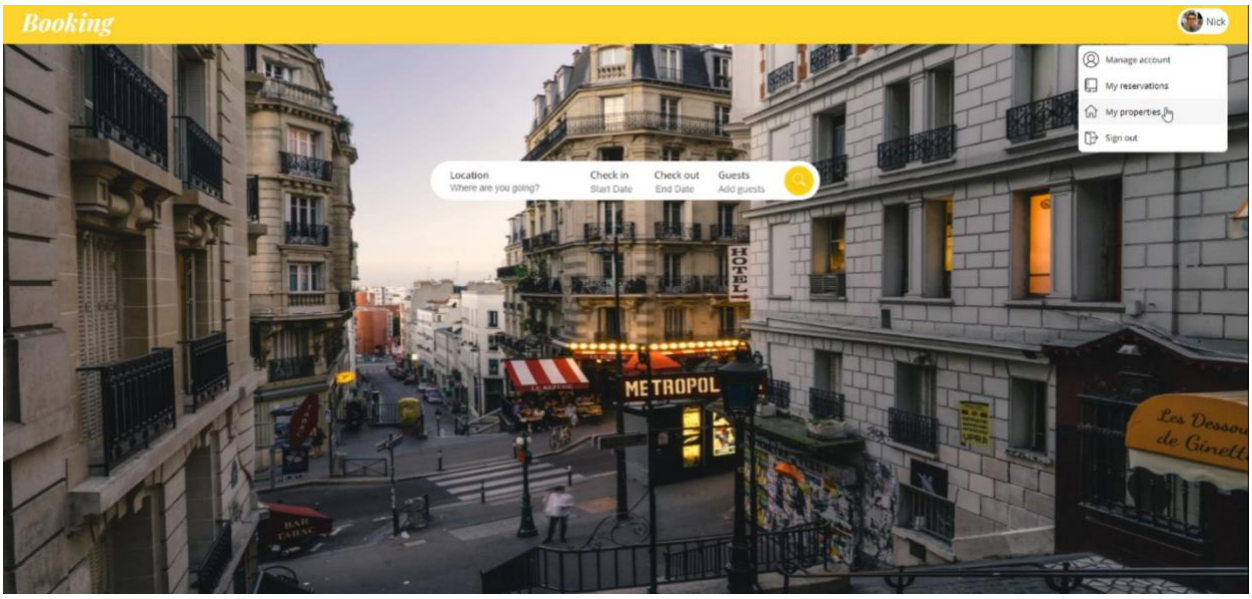


Έπειτα η εφαρμογή σε πηγαίνει σε σελίδα την οποία μπορείς να δεις πληροφορίες για την αγγελία σου με κατάλληλο τρόπο παρουσιασμένες. Υπάρχει ειδικό slider για να δεις τις φωτογραφίες τις οποίες έχεις ανεβάσει.

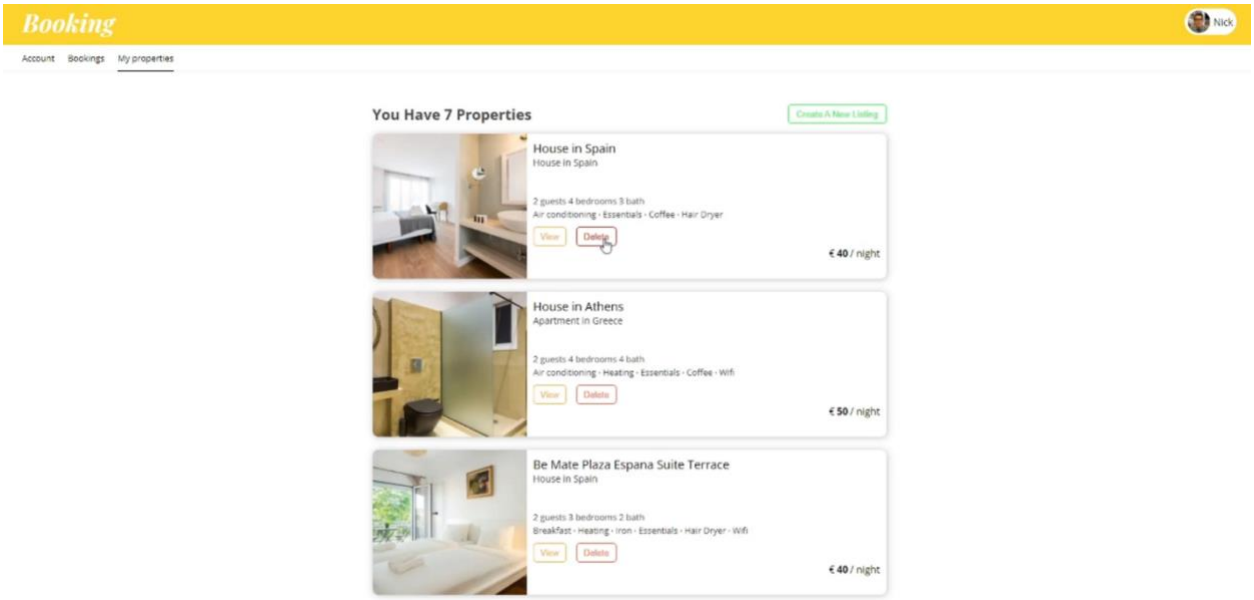


5.1.6 Διαγραφή Αγγελίας

Για να διαγράψεις μια αγγελία την οποία έχεις δημοσιεύσει πατάς στο κουμπί που αναγράφεται το όνομα χρήστη πάνω δεξιά και επιλέγεις My properties από το menu που εμφανίζεται.

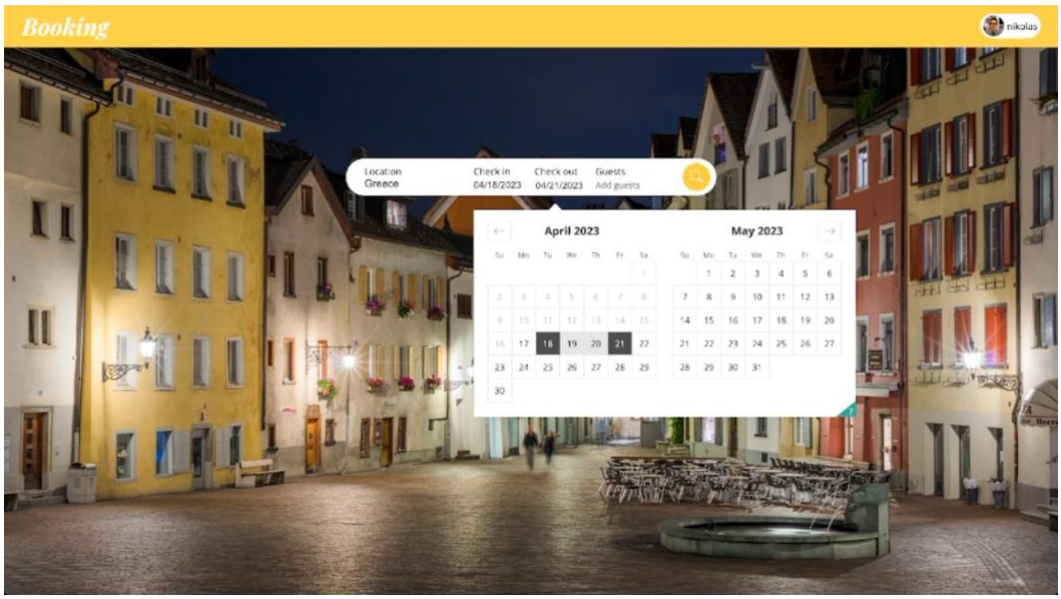


Στη σελίδα που εμφανίζεται αναγράφονται με κατάλληλο τρόπο όλες οι αγγελίες που έχεις δημοσιεύσει. Για να διαγράψεις μια αγγελία κλικάρεις το κόκκινο κουμπί Delete το οποίο βρίσκεται στην καρτέλα της αγγελίας που θες να διαγράψεις. Έπειτα η εφαρμογή διαγράφη την αγγελία και σου εμφανίζει τις υπόλοιπες αγγελίες.

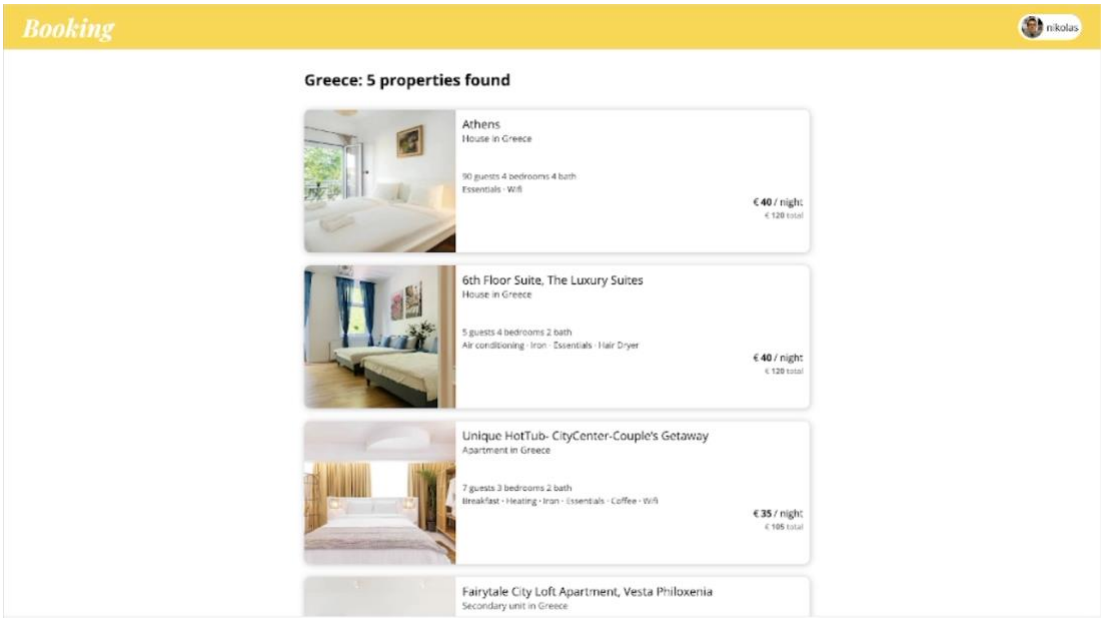


5.1.7 Εύρεση Κατοικίας και Κράτηση

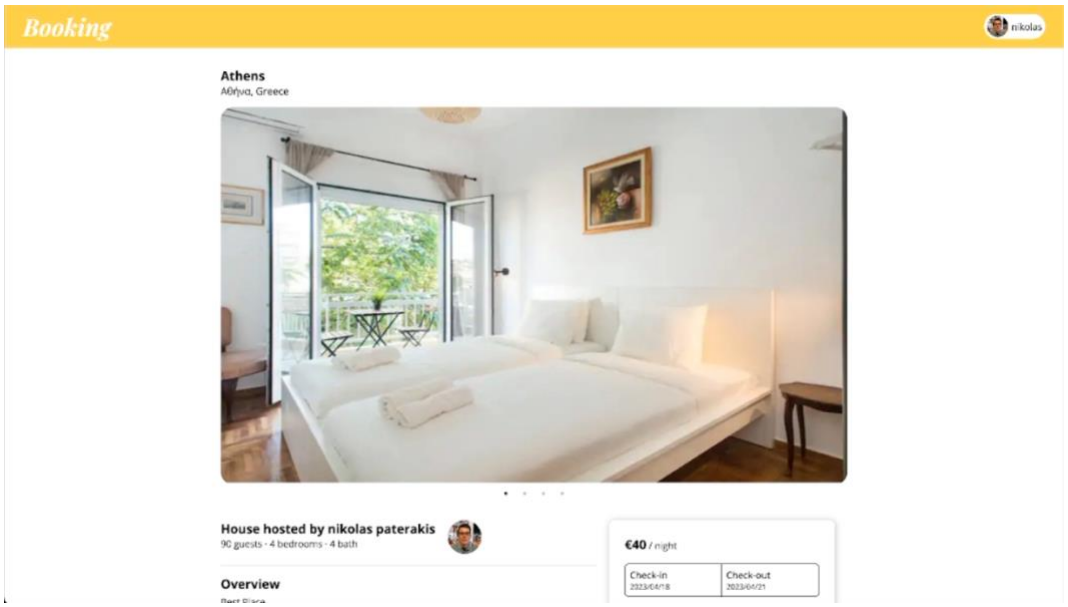
Για να βρεις την κατάλληλη κατοικία για τη διαμονή σου, πρέπει να παρέχεις στοιχεία σχετικά με την τοποθεσία που επιθυμείς, τη χρονική περίοδο που θέλεις να μείνεις και τον αριθμό των ατόμων που θα μείνουν σε αυτή.



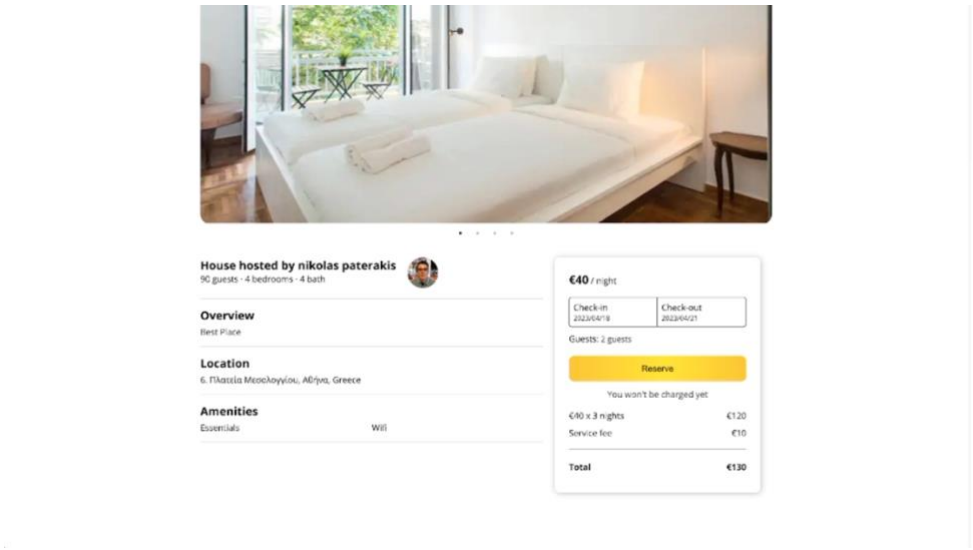
Αφού ο χρήστης συμπληρώσει τα στοιχεία για την κατοικία του, το σύστημα επεξεργάζεται αυτά τα δεδομένα και του παρουσιάζει μια λίστα με τις διαθέσιμες κατοικίες που ανταποκρίνονται στα κριτήρια του.



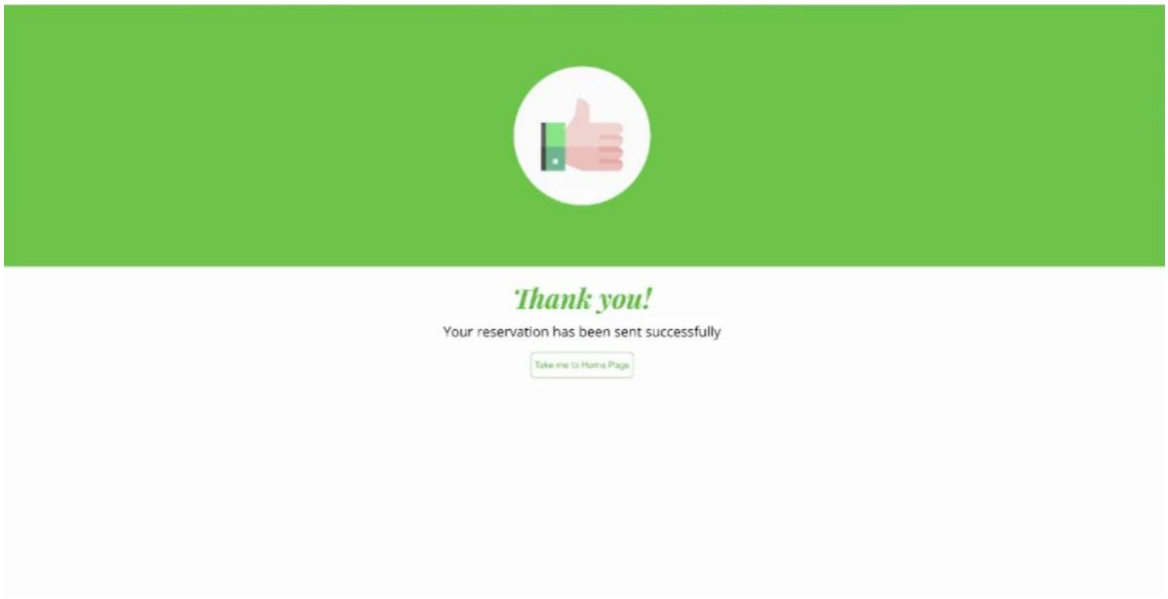
Αφού ο χρήστης επιλέξει μια κατοικία από τη λίστα των διαθέσιμων κατοικιών, το σύστημα τον μεταφέρει σε μια νέα σελίδα όπου παρουσιάζεται αναλυτικά η επιλεγμένη κατοικία. Σε αυτήν τη σελίδα μπορούν να παρουσιαστούν λεπτομέρειες όπως ο αριθμός των δωματίων, οι παροχές και οι ανέσεις που προσφέρονται, οι τιμές, οι εικόνες της κατοικίας και οι πληροφορίες για την περιοχή.



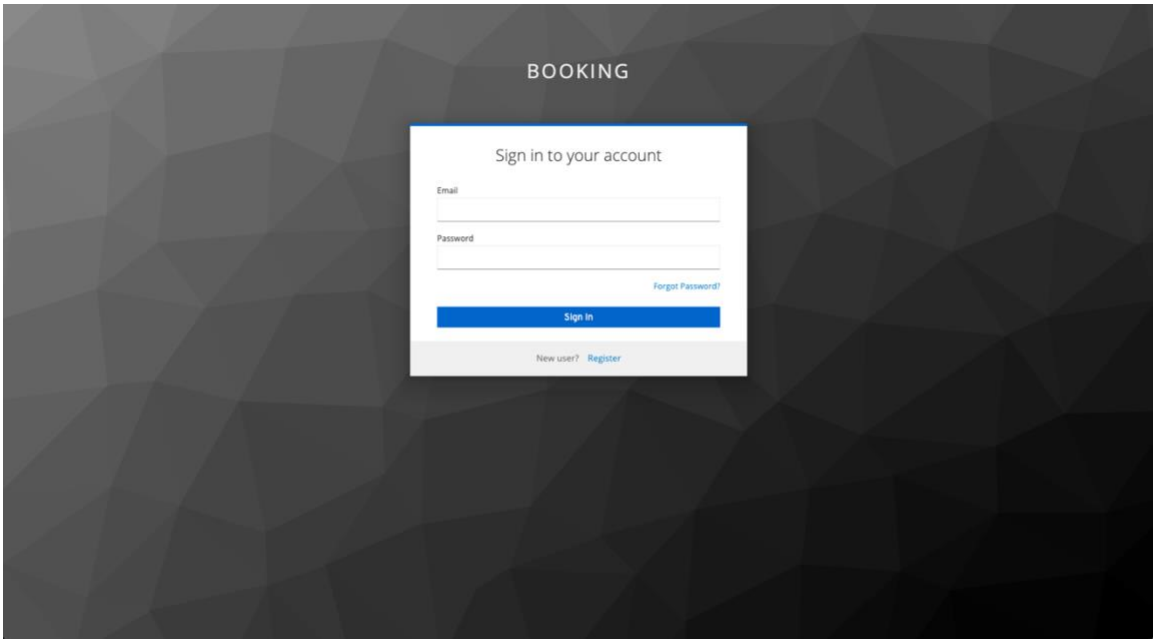
Αν ο χρήστης έχει βρει μια κατοικία που του αρέσει και θέλει να προχωρήσει στην κράτηση της, πρέπει να πατήσει το κουμπί "Reserve" (Κράτηση) που βρίσκεται στην σελίδα παρουσίασης της κατοικίας που επέλεξε.



Μόλις ο χρήστης πατήσει το κουμπί "Reserve" (Κράτηση), μεταφέρεται σε μια σελίδα ευχαριστίας όπου ενημερώνεται ότι η κράτησή του πραγματοποιήθηκε με επιτυχία. Επιπλέον, η σελίδα αυτή περιλαμβάνει ένα κουμπί που επιτρέπει στον χρήστη να επιστρέψει στην αρχική σελίδα.



Αν ο χρήστης δεν είναι συνδεδεμένος ή δεν έχει λογαριασμό στην εφαρμογή και πατήσει το κουμπί "Reserve" (Κράτηση), ο χρήστης θα μεταφερθεί σε μια σελίδα όπου μπορεί να συνδεθεί στον υπάρχον λογαριασμό του ή να δημιουργήσει ένα νέο λογαριασμό, αν δεν έχει ήδη έναν. Μετά μπορεί να πραγματοποιήσει κράτηση στην κατοικία που επιθυμεί.



Εάν ο χρήστης πραγματοποιήσει μια κράτηση με επιτυχία, η εφαρμογή αποστέλλει αυτόματα ένα email στον ιδιοκτήτη της κατοικίας για να ενημερωθεί για τη νέα κράτηση.

You have a new reservation at Fairytale City Loft Apartment, Vesta Philoxenia

---

Customer's Details

Name: nikolas paterakis

Email: nikolas.paterakis@gmail.com

Reservation's Details

Check In: 2023-05-16

Check Out: 2023-05-18

Price: 100€

Property's Information

Title: Fairytale City Loft Apartment, Vesta Philoxenia

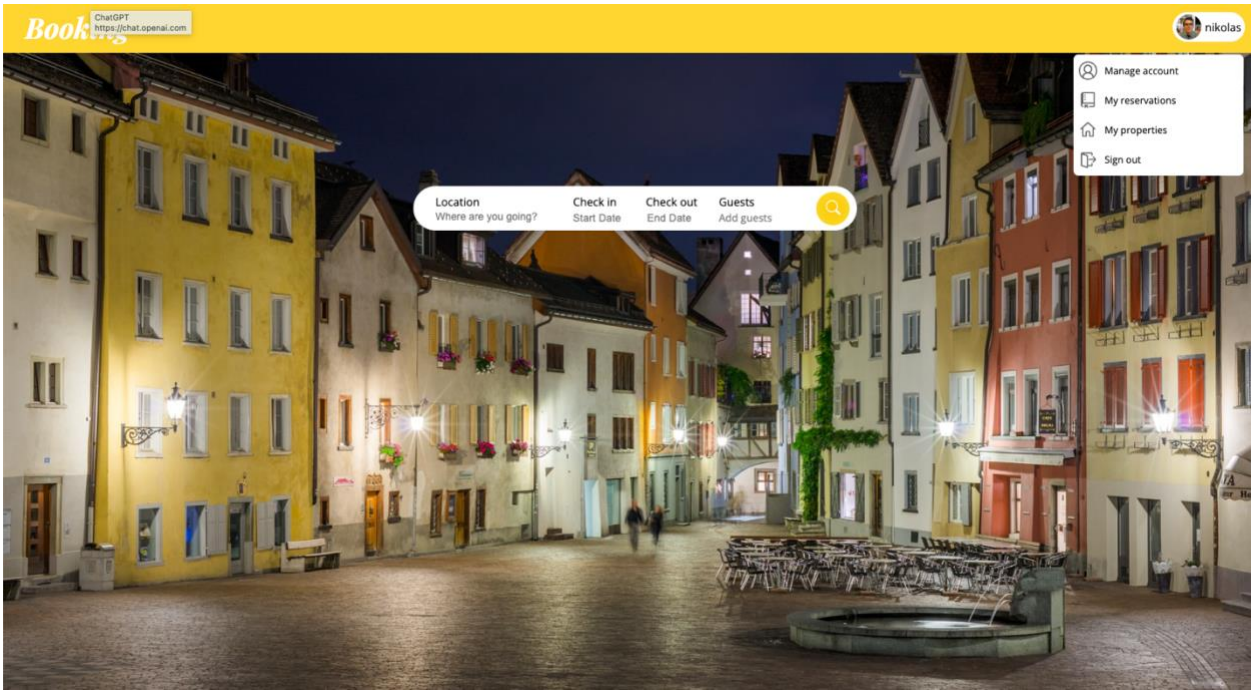
Location: Greece

Property Type: Secondary unit

Price Per Night: 50.0€

5.1.8 Διαχείριση Κρατήσεων

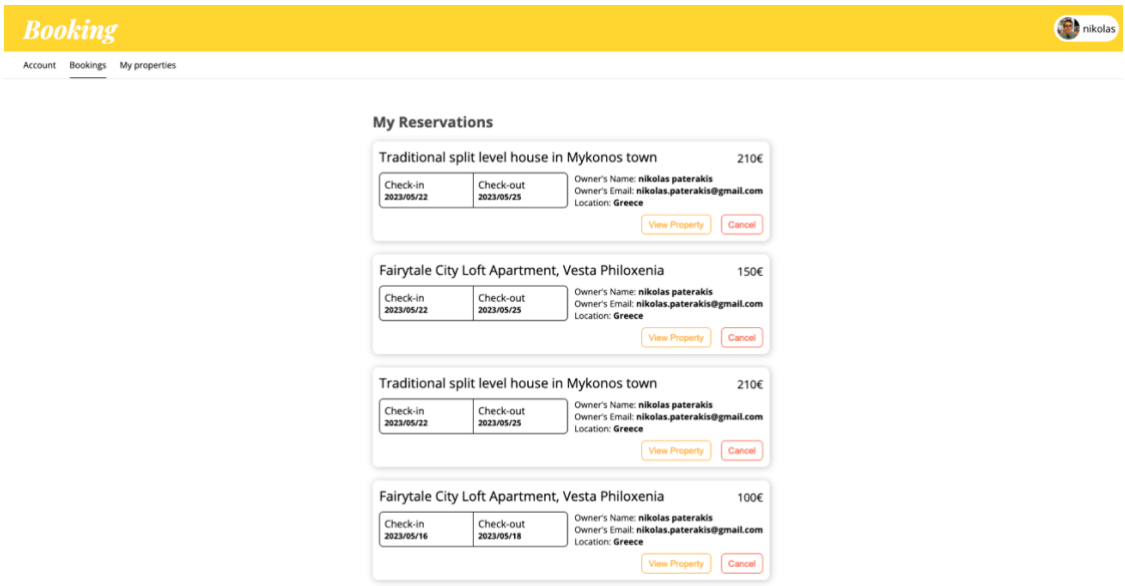
Η εφαρμογή παρέχει στον χρήστη τη δυνατότητα να δει τις κρατήσεις που έχει κάνει. Για να δει τις κρατήσεις του, ο χρήστης πρέπει να κάνει κλικ στο κουμπί "Προφίλ" που βρίσκεται στην πάνω δεξιά γωνία της σελίδας και έπειτα να επιλέξει την επιλογή "My Reservations" από το μενού που εμφανίζεται. Σε αυτήν τη σελίδα, ο χρήστης μπορεί να δει όλες τις κρατήσεις που έχει κάνει, καθώς και τις λεπτομέρειες κάθε κράτησης, όπως ημερομηνίες διαμονής, αριθμός ατόμων, κόστος κλπ.



Όταν κάνεις κλικ στο κουμπί "My Reservations", μεταφέρεσαι σε μια νέα σελίδα όπου βλέπεις μια λίστα με όλες τις κρατήσεις που έχεις κάνει. Σε αυτήν τη λίστα μπορείς να δεις λεπτομέρειες σχετικά με κάθε κράτηση, όπως ημερομηνίες διαμονής, αριθμός ατόμων, κόστος κλπ. Μπορείς επίσης να δεις πληροφορίες για τον ιδιοκτήτη της κατοικίας που έχει γίνει η κράτηση.

Εάν η κράτηση είναι σε μελλοντική ημερομηνία, μπορείς να ακυρώσεις την κράτηση κάνοντας κλικ στο κουμπί "Cancel". Αν το κουμπί δεν είναι διαθέσιμο, σημαίνει ότι η κράτηση δεν μπορεί να ακυρωθεί.

Επιπλέον, για κάθε κράτηση που έχει γίνει, υπάρχει ένα κουμπί "View Property". Όταν πατάς το κουμπί "View Property", μεταφέρεσαι σε μια νέα σελίδα όπου μπορείς να δεις λεπτομέρειες σχετικά με την κατοικία που έχει γίνει η κράτηση. Σε αυτήν τη σελίδα μπορείς να δεις φωτογραφίες της κατοικίας, λεπτομέρειες για τις παροχές της και πληροφορίες σχετικά με την τοποθεσία της.



Αν ο χρήστης αποφασίσει να ακυρώσει μια κράτηση και πατήσει το κουμπί "Cancel", η εφαρμογή θα στείλει αυτόματα ένα email στον ιδιοκτήτη της κατοικίας για να τον ενημερώσει για την ακύρωση. Το email θα περιέχει πληροφορίες σχετικά με την ακύρωση, όπως το όνομα του πελάτη, το όνομα της κατοικίας, οι ημερομηνίες της κράτησης, και το κόστος αυτής.

6. Συμπεράσματα

Στα πλαίσια αυτής της διπλωματικής εργασίας, μελετήθηκε η αρχιτεκτονική των microservices και οι τρόποι ανάπτυξης ενός συστήματος με αυτήν την αρχιτεκτονική. Μέσω της ανάλυσης των θεωρητικών πτυχών του cloud computing, της αρχιτεκτονικής των microservices και των σχετικών πρακτικών, αποκτήθηκε μια πληρέστερη κατανόηση του πώς μπορούμε να υλοποιήσουμε αποτελεσματικά ένα σύστημα με αρχιτεκτονική microservices.

Η παρούσα διπλωματική εργασία παρέχει μια ολοκληρωμένη μελέτη της αρχιτεκτονικής των microservices και των σχετικών πρακτικών, η έρευνα προσφέρει κατανόηση για την υλοποίηση και τη διαχείριση συστημάτων με αρχιτεκτονική microservices.

Οι βασικές ενότητες αυτής της διπλωματικής είναι οι εξής:

1. Ανασκόπηση της αρχιτεκτονικής των microservices: Η διπλωματική παρουσίασε μια λεπτομερή επισκόπηση της αρχιτεκτονικής αυτής, εξηγώντας τις βασικές αρχές και τα πλεονεκτήματα που προσφέρει. Αυτή η ανάλυση βοηθάει στην κατανόηση της λειτουργίας των microservices και στην ανάπτυξη ευέλικτων και αποτελεσματικών συστημάτων λογισμικού.
2. Ανάπτυξη πρακτικής εφαρμογής: Η έρευνα περιλάμβανε την ανάπτυξη μιας πρακτικής εφαρμογής με αρχιτεκτονική microservices. Αυτός ο πρακτικός προσανατολισμός βοήθησε στην εφαρμογή και την αξιολόγηση του θεωρητικού μέρους και την κατανόηση της πρακτικής λειτουργίας και των πλεονεκτημάτων αυτής της αρχιτεκτονικής.

Το πρακτικό μέρος αυτής της διπλωματικής εργασίας αποτελείται από την ανάπτυξη μιας εφαρμογής. Από το πρακτικό μέρος προέκυψαν σημαντικά συμπεράσματα και ευρήματα που αξίζει να αναφερθούν:

1. Επίδοση και αποτελεσματικότητα: Η χρήση της αρχιτεκτονικής microservices επέτρεψε τη δημιουργία ενός ευέλικτου και αποδοτικού συστήματος. Η διάσπαση της εφαρμογής σε μικρά, ανεξάρτητα κομμάτια λογισμικού επέτρεψε την ανάπτυξη, τη συντήρηση και την επεκτασιμότητα του συστήματος με μεγαλύτερη ευελιξία. Αυτό οδήγησε σε βελτιωμένη απόδοση και αποτελεσματικότητα σε σύγκριση με παραδοσιακές μονολιθικές αρχιτεκτονικές.
2. Ευελιξία και κλιμακωσιμότητα: Η αρχιτεκτονική microservices επέτρεψε την επεκτασιμότητα του συστήματος με την προσθήκη νέων υπηρεσιών κατά αίτηση και την αυτόματη κλιμάκωση των υπηρεσιών σύμφωνα με τις ανάγκες φόρτου. Αυτό επέτρεψε στο σύστημα να προσαρμόζεται ευέλικτα σε αλλαγές και αυξημένες απαιτήσεις.
3. Εικονικοποίησης βασισμένης σε περιέκτες: Η χρήση της εικονικοποίησης βασισμένης σε περιέκτες με το εργαλείο docker επέτρεψε την ευκολότερη δημιουργία και ανάπτυξη των microservices. Μέσω της εικονικοποίησης, κάθε service μπορούσε να εκτελεστεί ανεξάρτητα σε ένα περιβάλλον που περιλαμβάνει όλες τις απαιτούμενες εξαρτήσεις. Αυτό οδήγησε σε απλοποίηση της διαδικασίας ανάπτυξης και επιτράπηκε η παραγωγή έτοιμων εικόνων που μπορούν να χρησιμοποιηθούν σε διάφορα περιβάλλοντα.
4. Ορχήστρωση περιεκτών: Η ορχήστρωση περιεκτών με το εργαλείο Kubernetes στο πρακτικό μέρος επέτρεψε τη διαχείριση και την αυτόματη κλιμάκωση των microservices. Μέσω της διαχείρισης ελαστικού χώρου, οι υπηρεσίες μπορούσαν να προσαρμοστούν δυναμικά στις αυξημένες απαιτήσεις φόρτου, εξασφαλίζοντας την υψηλή διαθεσιμότητα και την αποτελεσματική χρήση πόρων. Επιπλέον, η αυτόματη επαναφορά σφαλμάτων και η δυνατότητα αναβάθμισης των υπηρεσιών με ελάχιστο χρόνο αδράνειας βελτίωσαν την αξιοπιστία και τη συνολική απόδοση του συστήματος

Παρά τις προκλήσεις που συνεπάγεται η αρχιτεκτονική των microservices, όπως η περίπλοκη επικοινωνία μεταξύ των services και η ανάγκη για εξειδικευμένο ανθρώπινο δυναμικό, η υιοθέτηση αυτής της προσέγγισης μπορεί να οδηγήσει σε ορισμένα πλεονεκτήματα. Με την αρχιτεκτονική των microservices, είναι δυνατή η επεκτασιμότητα του συστήματος, η αντιμετώπιση γρήγορων αλλαγών και η προσαρμοστικότητα σε νέες απαιτήσεις των χρηστών.

Οι επιπτώσεις ενός σφάλματος σε ένα σύστημα με αρχιτεκτονική microservices περιορίζονται μόνο στο σχετικό service, ενώ τα υπόλοιπα services συνεχίζουν να λειτουργούν. Αυτό οδηγεί σε μια καλύτερη αντοχή του συστήματος σε σφάλματα και μεγαλύτερη αξιοπιστία.

Συνοψίζοντας, η αρχιτεκτονική των microservices προσφέρει πολλαπλά οφέλη για την ανάπτυξη λογισμικού. Με την κατάλληλη υλοποίηση και διαχείριση των microservices, μπορούμε να επιτύχουμε ευελιξία, αντοχή σε σφάλματα και αποτελεσματική ανάπτυξη συστημάτων. Η παρούσα διπλωματική εργασία αποτελεί μια συμβολή στην κατανόηση και εφαρμογή της αρχιτεκτονικής αυτής, και παρέχει μια βάση για μελλοντικές έρευνες και αναπτύξεις σε αυτόν τον σημαντικό τομέα της πληροφορικής.

Αναφορές

1. IBM Cloud. Containerization: A complete guide. Retrieved from https://www.ibm.com/cloud/learn/containerization
2. Red Hat. Containers vs. virtual machines (VMs): What's the difference? Retrieved from https://www.redhat.com/en/topics/containers/containers-vs-vms
3. IBM Cloud. What is cloud computing? Retrieved from https://www.ibm.com/cloud/learn/cloud-computing
4. Wikipedia. Java (programming language). Retrieved from https://en.wikipedia.org/wiki/Java_(programming_language)
5. Wikipedia. IntelliJ IDEA. Retrieved from https://en.wikipedia.org/wiki/IntelliJ_IDEA
6. Wikipedia. Visual Studio Code. Retrieved from https://en.wikipedia.org/wiki/Visual_Studio_Code
7. Microsoft Azure. What is cloud computing? A beginner’s guide. Retrieved from https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#uses
8. Rouse, M. What is DevOps? - Definition from WhatIs.com. Retrieved from https://www.techtarget.com/searchitoperations/definition/DevOps
9. Wikipedia. MySQL. Retrieved from https://en.wikipedia.org/wiki/MySQL
10. Wikipedia. HTML. Retrieved from https://el.wikipedia.org/wiki/HTML
11. Baeldung. Service Discovery in a Microservices Architecture. Retrieved from https://www.baeldung.com/cs/service-discovery-microservices
12. Better Programming. The Complete Guide to OAuth 2.0 and OpenID Connect Protocols. Retrieved from https://betterprogramming.pub/the-complete-guide-to-oauth-2-0-and-openid-connect-protocols-35ebc1cbc11a
13. OpenID. OpenID Connect. Retrieved from https://openid.net/connect/
14. Auth0. What is OAuth 2.0? Retrieved from https://auth0.com/intro-to-iam/what-is-oauth-2/
15. Wikipedia. OAuth. Retrieved from https://en.wikipedia.org/wiki/OAuth
16. Wikipedia. React (JavaScript library). Retrieved from https://en.wikipedia.org/wiki/React_(JavaScript_library)
17. Wikipedia. Sass (stylesheet language). Retrieved from https://en.wikipedia.org/wiki/Sass_(stylesheet_language)
18. Richardson, Chris. "Microservices Patterns: With examples in Java." November 19, 2018
19. Long, Josh. "Reactive Spring." September 10, 2020