ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ και ΥΠΟΛΟΓΙΣΤΩΝ

ΡΟΗ ΛΟΓΙΣΜΙΚΟΥ και ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή διεπαφής έξυπνων συσκευών με σύστημα υποστήριξης χρηστών & αναφοράς σφαλμάτων.

Νικόλαος Γιαννακόπουλος

Α.Μ: 711-131066

Επιβλέπων: Χρήστος Τρούσσας, Επ. Καθηγητής

Διπλωματική εργασία υποβληθείσα στο Τμήμα

ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και 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 σύνολό 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.

Ο Δηλών

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Νικόλαος Γιαννακόπουλος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ και ΥΠΟΛΟΓΙΣΤΩΝ

ΡΟΗ ΛΟΓΙΣΜΙΚΟΥ και ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Η παρούσα διπλωματική εργασία παρουσιάστηκε από τον

Γιαννακόπουλος Νικόλαος

Α.Μ: 711-131066

την [27/10/2022]

Εισηγητής

Χρήστος Τρούσσας, Επίκουρος Καθηγητής

Εξεταστική Επιτροπή

Χρήστος Τρούσσας Christos Troussas Digitally signed by Christos Troussas Date: 2022.10.31 15:50:53 +02'00'

Κλειώ Σγουροπούλου Kleio Sgouropoulou Digitally signed by Kleio Sgouropoulou Date: 2022.12.16 13:04:30 +02'00'

Γιώργος Μελετίου Georgios Meletiou Digitally signed by Georgios Meletiou Date: 2022.11.03 15:29:47 +02'00'

Ημερομηνία εξέτασης 20/10/2022

Η έγκριση της διπλωματικής εργασίας δεν υποδηλοί την αποδοχή των γνωμών του συγγραφέα. Κατά τη συγγραφή τηρήθηκαν οι αρχές της ακαδημαϊκής δεοντολογίας.

ΠΕΡΙΛΗΨΗ

Η χρήση συστημάτων αναφοράς σφαλμάτων κρίνεται απαραίτητη σε μια εποχή που μεγάλες και πολύπλοκες υποδομές αντιμετωπίζουν ολοένα και μεγαλύτερες προκλήσεις στην διαχείρηση και οργάνωσή τους. Τέτοιου είδους συστήματα είναι κατά κανόνα υλοποιημένα ως εφαρμογές Web, κάνοντας πιο δύσκολο ο χρήστης να τα εκμεταλλευτεί όπου και όποτε χρειαστεί.

Η διπλωματική αυτή εργασία στοχεύει στη δημιουργία μίας mobile εφαρμογής που να συνδυάζει την φορητότητα των μοντέρνων έξυπνων συσκευων με τα συστήματα αναφοράς σφαλμάτων, προσφέροντας τη δυνατότητα γρήγορης αναφοράς προβλημάτων όταν προκύπτουν με αποτέλεσμα την άμεση επίλυσή τους.

Η εφαρμογή αυτή αναπτύχθηκε στο Android Studio IDE, σχεδιασμένη με βάση το πρότυπο ανθρωποκεντρικού σχεδιασμού και χρησιμοποιεί RESTful API calls για την επικοινωνία με το ανοιχτού κώδικα σύστημα υποστήριξης πελατών osTicket.

Το αποτέλεσμα είναι μία τελική εφαρμογή ικανή να προσαρμοστεί σε οποιαδήποτε εγκατάσταση του osTicket, με χρήση προσωποποιημένων πεδίων για τη δημιουργία αναφορών και την παρακολούθηση της προόδου τρέχοντων ζητημάτων.

Εφαρμογή διεπαφής έξυπνων συσκευών με σύστημα υποστήριξης χρηστών & αναφοράς σφαλμάτων.

Νικόλαος Γιαννακόπουλος

Λέξεις κλειδιά

- API - RESTful API
- Μοντέλο Σχεδιασμού
- Σύστημα Υποστήριξης - osTicket
- Λειτουργικό Σύστημα Android

ABSTRACT

The use of Support Ticketing Systems is crucial at a time when complex, large-scale infrastructure faces ever-increasing challenges to its organisation and administration. These types of systems are, as a rule, implemented as Web applications, making it more difficult for a user to utilise them whenever and wherever necessary.

This paper aims at the creation of a mobile application that combines the portability of modern smart devices with the functionality of Support Ticketing Systems, offering the ability to quickly and easily report issues as they appear, resulting in a quicker resolution process.

This application was developed using the Android Studio IDE, designed in accordance with the Human-Centered Model of Design, and uses RESTful API calls to communicate with the open source customer support system osTicket.

The result is a final application adjustable to any installation of osTicket, which uses personalisable fields for the creation of tickets and in order to track the status of existing tickets.

Customer Support & Support Ticketing System

Interface Application for Smart Devices

Nikolaos Giannakopoulos

Keywords

- API - RESTful API
- Model Design
- Support Ticketing System - osTicket
- Android

ΚΑΤΑΛΟΓΟΣ ΤΜΗΜΑΤΩΝ ΚΩΔΙΚΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΕΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΟΡΟΛΟΓΙΩΝ

Ελληνική Ορολογία	**Αγγλική Ορολογία**	**Συντομογραφία**
Διεπαφή Χρήστη	*User Interface*	*UI*
Εμπειρία Χρήστη	*User Experience*	*UX*
Ανθρωποκεντρική Διαδικασία Σχεδίασης	*Human Centered Designed Process*	*HCDP*
Σύστημα Ονοματοδοσίας Δικτύου	*Domain Name System*	*DNS*
Βάση Δεδομένων	*DataBase*	*DB*
Πρωτόκολλο Μεταφοράς Υπερκειμένου	*Hypertext Transfer Protocol*	*HTTP*
Διεπαφή Προγραμματισμού Εφαρμογών	*Application Programming Interface*	*API*
Αναπαράσταση κατάστασης μεταφοράς	*Representational State Transfer*	*RESTful API*
Δικτυακό Γραφικό Περιβάλλον Διεπαφής	*Web Graphical User Interface*	*Web GUI*
Ολοκληρωμένο Περιβάλλον Προγραμματισμού	*Integrated Development Environment*	*IDE*
Απλό παλιό Αντικείμενο Java	*Plain Old Java Object*	*POJO*

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον κύριο Τρούσσα Χρήστο που ανέλαβε την διπλωματική, τον Φωστίνη Πέτρο για την συνεργασία και βοήθεια που προσέφερε στην ανάπτυξη της εφαρμογής, και τέλος την οικογένεια και τους φίλους που με στήριξαν και υπέμειναν τόσα χρόνια.

1. ΕΙΣΑΓΩΓΗ

Στην σημερινή εποχή, οι έξυπνες φορητές συσκευές αποτελούν αναπόσπαστο κομμάτι της ρουτίνας μας. Τα χαρακτηριστικά τους εμπλουτίζονται και η δυναμική τους μεγαλώνει, βελτιώνοντας τις δυνατότητες που μας προσφέρουν. Η χρησιμότητά τους είναι καθοριστική, αφού αποτελούν το μέσο που παρέχει την κατάλληλη υποδομή για την λειτουργία εφαρμογών με τις οποίες αλληλεπιδρούμε σε καθημερινή βάση.

Οι συσκευές αυτές επιτρέπουν την εύκολη και γρήγορη επικοινωνία, πληροφόρηση, ψυχαγωγία και εξυπηρέτηση του ανθρώπου σε διάφορους τομείς. Εκμεταλλευόμενοι την ικανότητα της πληροφορικής να επεμβαίνει στη ζωή του ανθρώπου με θετικό τρόπο με μόνο στόχο την διευκόλυνσή της, αναπτύξαμε μια εφαρμογή σε λειτουργικό Android, με στόχο την δημιουργία μιας απλής και εύχρηστης διεπαφής χρήστη που θα προσφέρει στον χρήστη τη δυνατότητα αναφοράς σφαλμάτων εν κινήσει. Το τελικό αποτέλεσμα είναι μια εφαρμογή "πασπαρτού", ικανή να προσαρμοστεί στις διαφορετικές ανάγκες των χρηστών και των διαχειριστών.

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η αξιοποίηση ενός Web-based λογισμικού για αναφορά σφαλμάτων με όνομα osTicket, και η δημιουργία μιας mobile εφαρμογής που θα παρέχει στο χρήστη τη δυνατότητα να αναφέρει βλάβες οπουδήποτε και οποτεδήποτε, με σκοπό την άμεση επίλυσή τους και την καλύτερη εξυπηρέτησή τους.

Η χρήση ενός τέτοιου εργαλείου προσφέρει σε απλούς χρήστες την δυνατότητα δημιουργίας δελτίων αναφοράς σφαλμάτων *(Ticket)*, και προσφέρει σε ομάδες υποστήριξης πελατών τη δυνατότητα να δημιουργούν, να διαχειρίζονται και να διατηρούν λίστες από Ticket. Τα συστήματα αυτά είναι ιδιαίτερα χρήσιμα σε μεγάλους οργανισμούς και εταιρίες για την εύκολη και γρήγορη παρακολούθηση διαφόρων ζητημάτων όπως προκύπτουν, αντιστοιχίζοντάς τα γρήγορα και εύκολα με τους κατάλληλους πράκτορες για επίλυση.

Ένα σωστά εφαρμοσμένο σύστημα έκδοσης δελτίων *(Ticket)* δημιουργεί ένα δίκτυο καλύτερης επικοινωνίας μέσα σε έναν οργανισμό, μια εταιρία ή ακόμα και μια ομάδα. Αυτό επιτυγχάνεται με την εξάλειψη όλων των σημείων συμφόρησης που δημιουργούνται όταν κανείς δεν έχει μια γενική εικόνα του προβλήματος που προκύπτει, της προτεραιότητας του, και του ατόμου που το αναφέρει. Η εφαρμογή ενός τέτοιου λογισμικού απαιτεί χρήστες με κατάλληλη εκπαίδευση, αλλά και την εφαρμογήσυγκεκριμένων διαδικασιών για να λειτουργήσει σωστά.

Το εργαλείο αυτό θα μπορούσε φυσικά να ενσωματωθεί και στην υπάρχουσα εφαρμογή του osTicket στους χώρους του Πανεπιστημίου Δυτικής Αττικής, προκειμένου να χρησιμοποιηθεί από καθηγητές, φοιτητές και γενικά από το προσωπικό του Πανεπιστημίου, δίνοντας τους τη δυνατότητα να συνεισφέρουν στην καλή συντήρηση του χώρου του Πανεπιστημίου. έρα από τους χώρους του ΠΑΔΑ μπορεί να χρησιμοποιηθεί και από οποιονδήποτε άλλο φορέα ή υπηρεσία με σκοπό την αναφορά βλαβών στους χώρους υποδομής και όχι μόνο. Το

πλεονέκτημά του είναι η διευκόλυνση που προσφέρει σε ομάδες, εταιρίες, οργανισμούς, με την άμεση ενημέρωση για την άρτια λειτουργία του συστήματος που έχει προσαρμοστεί το εργαλείο αυτό. Ταυτόχρονα προσφέρεται στους χρήστες η δυνατότητα να συνεισφέρουν στην καλή συντήρηση του συστήματος αυτού, με αποτέλεσμα να παρακάμπτονται αργές διαδικασίες, δίνοντας άμεσες λύσεις σε σημαντικά προβλήματα όταν αυτά παρουσιάζονται, και ενισχύοντας παράλληλα την αλληλεγγύη και την κοινωνική ευθύνη.

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1. ΣΥΣΤΗΜΑ ΑΝΑΦΟΡΑΣ ΣΦΑΛΜΑΤΩΝ OSTICKET

Το osTicket είναι ένα ευρέως χρησιμοποιούμενο και αξιόπιστο εργαλείο ανοιχτού κώδικα με σκοπό την υποστήριξη πελατών και την διαχείρηση αναφορών σφαλμάτων. Προσφέρει τη δυνατότητα δημιουργίας δελτίων υποστήριξης, καθώς και εύκολη διαχείριση, οργάνωση και αρχειοθέτηση όλων των αιτημάτων υποστήριξης, όπως και τις απαντήσεις σε αυτά, σε μία ολοκληρωμένη εφαρμογή δικτύου *(WebAPP)*. Συγκεκριμένα το osTicket είναι γραμμένο σε γλώσσα PHP και είναι κατάλληλο για μικρομεσαίες επιχειρήσεις.

Το έναυσμα για την εργασία αυτή ήταν η έλλειψη φιλικής διεπαφής χρήστη του osTicket για κινητές συσκευές, καθώς μέχρι πρότινος ο βασικός τρόπος διεπαφής ήταν μέσω δικτυακής γραφικής διεπαφής χρήστη *(Web GUI)*. Σκοπός της εφαρμογής αυτής, λοιπόν, είναι να ικανοποιεί τις ανάγκες αυτές.

Κύριο μέλημά μας ήταν η υποστήριξη των λειτουργιών δημιουργίας και ελέγχου δελτίων υποστήριξης σε φιλική για τον χρήστη διεπαφής, για έξυπνες συσκευές Android, χρησιμοποιώντας κλήσεις RESTFul API.

Για την δημιουργία της εφαρμογής αυτής χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον προγραμματισμού Android Studio με γλώσσα προγραμματισμού Java. Για τον σκοπό της επικοινωνίας μεταξύ του πελάτη *(Client)* και του διακομιστή *(Server)* χρησιμοποιήθηκε η βιβλιοθήκη Retrofit2, που είναι υπεύθυνη για τις κλήσεις RESTFul API.

2.2. ANDROID STUDIO

Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον προγραμματισμού *(IDE)* για ανάπτυξη εφαρμογών Android, το οποίο αναπτύχθηκε από την Google Inc με βάση το λογισμικό της JetBrains' IntelliJ IDEA.

2.3. ΔΙΕΠΑΦΗ - ΕΜΠΕΙΡΙΑ ΧΡΗΣΗΣ ΚΑΙ ΜΟΝΤΕΛΟ ΣΧΕΔΙΑΣΜΟΥ

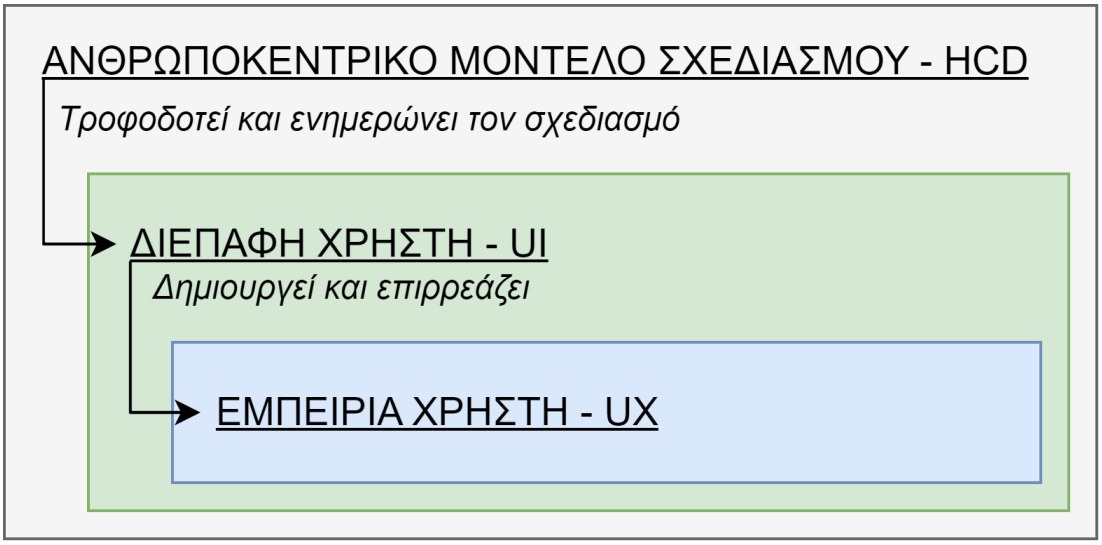
Στην ενότητα αυτή θα γίνει λόγος για κάποιες βασικές αρχές των πρώιμων σταδίων υλοποίησης της εφαρμογής. Πρώτο μέλημα ήταν η δημιουργία μιας κατάλληλης διεπαφής χρήστη *(User Interface - UI)* ώστε να επιτευχθεί η καλύτερη δυνατή εμπειρία χρήσης *(User Experience - UX)*.

Η Διεπαφή Χρήστη αφορά το περιβάλλον εργασίας και διάδρασης που επιτρέπουν στον χρήστη να αλληλεπιδρά με την εφαρμογή. Ο συνολικός σχεδιασμός, τα γραφικά και η παρουσίαση της εφαρμογής, συνθέτουν τη βασική δομή της διεπαφης του χρήστη. Για να είναι αποτελεσματικό ένα περιβάλλον εργασίας χρήστη, θα πρέπει να γίνεται εύκολα κατανοητό και ταυτόχρονα ελκυστικό.

Η Εμπειρία Χρήσης συνυπολογίζει τα ανθρώπινα συναισθήματα, την αντίληψη, αλλά και τις προτιμήσεις του χρήστη που δημιουργούνται κατά τη διάρκεια της χρήσης μιας εφαρμογής, καθώς και μετέπειτα. Είναι δηλαδή η συνολική εμπειρία που απολαμβάνει ο χρήστης στο περιβάλλον της εφαρμογής. Η εύκολη προσβαση, η απλότητα και η ευχρηστία μιας εφαρμογής, προσφέρουν μια ικανοποιητική εμπειρία στον χρήστη και φροντίζουν να μην δημιουργούν

αρνητικά συναισθήματα, ώστε να συνεχίσει να την χρησιμοποιεί. Συνεπώς, η διεξοδική έρευνα είναι απαραίτητη ώστε να εντοπιστούν οι ανάγκες των εν δυνάμει χρηστών στους οποίους απευθυνόμαστε και να δημιουργηθεί ένας αποτελεσματικός σχεδιασμός στην Εμπειρία Χρήστη.

Το Μοντέλο Ανθρωποκεντρικού Σχεδιασμού *(Human Centered Design - HCD)* είναι ένα πρότυπο σχεδίασης ανάπτυξης εφαρμογών, που προτρέπει την συλλογή και την ανάλυση κριτικών και εμπειριών από τους χρήστες, προκειμένου αυτές να χρησιμοποιηθούν για τη σχεδίαση και την δημιουργία πολυμεσικών εφαρμογών. Η συνεχής συλλογή των απόψεων χρηστών και η αξιοποίηση τους σε κάθε βήμα της ανάπτυξης εφαρμογών καθιστά το παραπάνω πρότυπο ιδανικό για την ανάπτυξη ενός ιδιαίτερα φιλικού προς τον χρήστη UI.



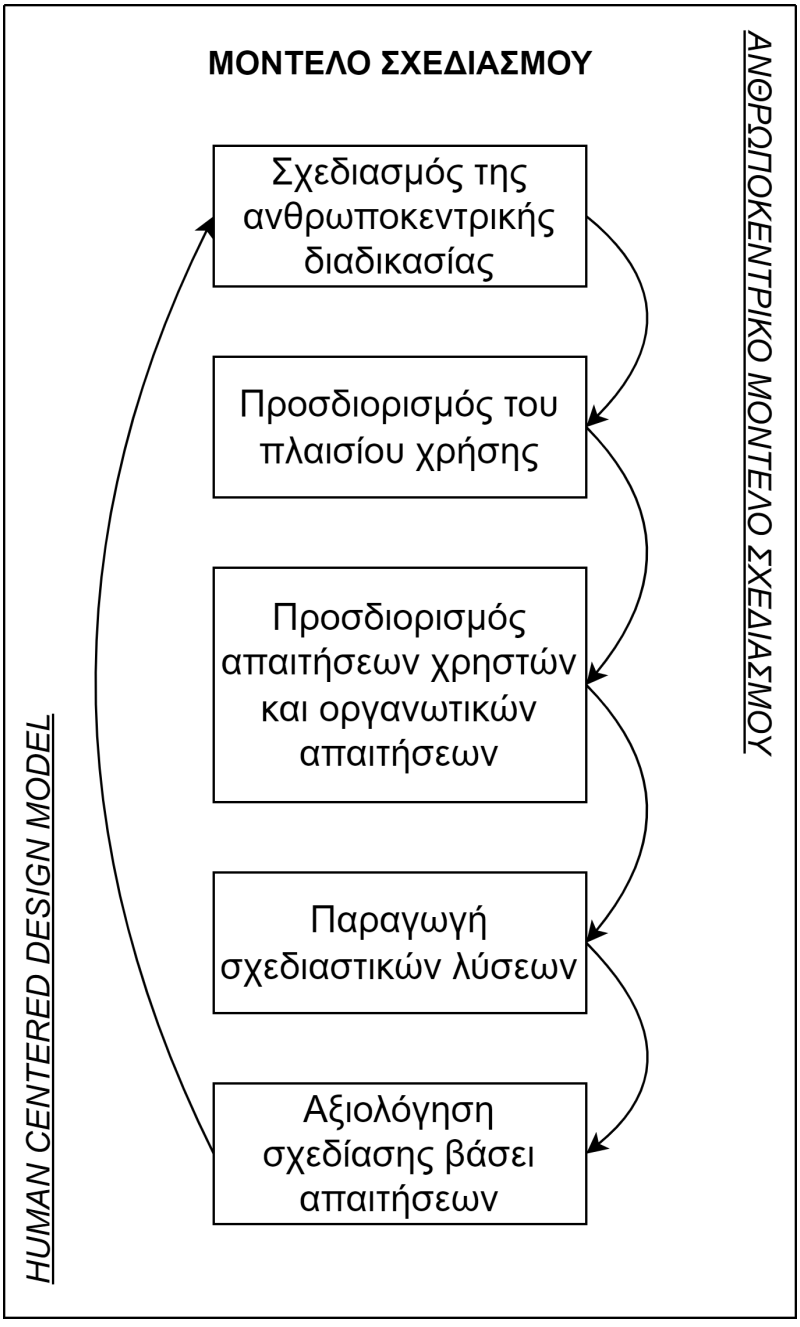
Σχεδιάγραμμα 2.3 - 1 - Συσχέτιση HCD - UI - UX

Ο λόγος που καθίσταται απαραίτητη η εμπλοκή του χρήστη αφορά την ευχρηστία και την βελτιωμένη εργονομία των εφαρμογών, εκπληρώνοντας τις ανάγκες και τις επιθυμίες των χρηστών για τους οποίου προορίζεται η εφαρμογή αυτή. Το πόσο εύχρηστη και προσαρμοσμένη είναι μια εφαρμογή στις ανάγκες του χρήστη κρίνει την αποδοχή και την επιτυχία της.

Είναι λοιπόν προτιμότερο να προβαίνουμε σε αξιολόγηση κάθε σταδίου ανάπτυξης της εφαρμογής και να κάνουμε τυχόν αλλαγές που υποδεικνύονται από τους χρήστες, αντί να υλοποιήσουμε πλήρως την εφαρμογή και στο τέλος να διαπιστώσουμε πως δεν ικανοποιείτο κοινό στο οποίο απευθύνεται.

Το ανθρωποκεντρικό μοντέλο σχεδιασμού, επομένως, ακολουθεί μία διαδικασία επαναλαμβανόμενων σταδίων από την έναρξη μέχρι και το τελικό στάδιο υλοποίησης της εφαρμογής, τα οποία είναι:

- Σχεδιασμός της ανθρωποκεντρικής διαδικασίας
- Προσδιορισμός του πλαισίου χρήσης
- Προσδιορισμός απαιτήσεων χρηστών και οργανωτικών απαιτήσεων
- Παραγωγή σχεδιαστικών λύσεων
- Αξιολόγηση σχεδίασης βάσει απαιτήσεων



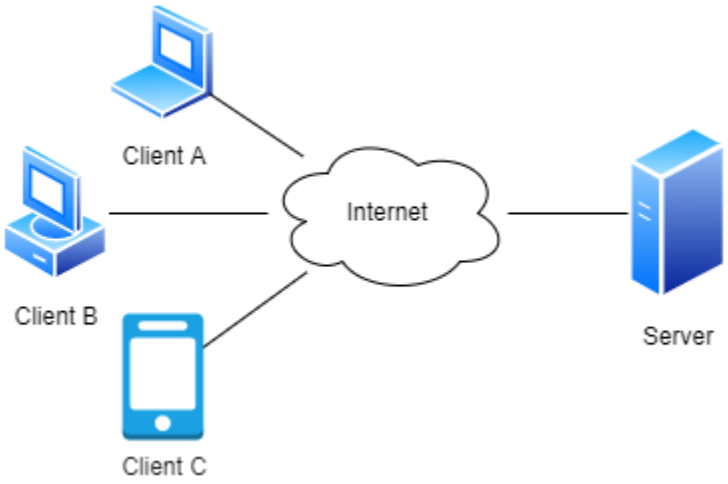
Σχεδιάγραμμα 2.3 - 2 - Ανθρωποκεντρικό Μοντέλο Σχεδιασμού

2.4. ΕΝΔΙΑΜΕΣΗ ΕΠΙΚΟΙΝΩΝΙΑ ΠΕΛΑΤΗ-ΔΙΑΚΟΜΙΣΤΗ ΜΕΣΩ RESTFUL API

Τα API είναι στο επίκεντρο του σημερινού συνεχώς αναπτυσσόμενου οικοσυστήματος λογισμικού. Υπάρχουν ουσιαστικά ατελείωτοι τρόποι σύνδεσης διαφορετικών Web εφαρμογών και τα API τροφοδοτούν αυτά τα integration στο παρασκήνιο. Η συντομογραφία API απαντά στο «Application Programming Interface», και αν θέλαμε να αποδώσουμε τον όρο όσο καλύτερα γίνεται στα ελληνικά θα λέγαμε ότι είναι η «Διεπαφή Προγραμματισμού Εφαρμογών».

Το API είναι ένα ‘‘ενδιάμεσο’’ λογισμικό, το οποίο επιτρέπει την επικοινωνία μεταξύ δύο εφαρμογών. Είναι δηλαδή ο "αγγελιοφόρος" που παραδίδει το αίτημα στον πάροχο, από τον οποίο ζητείται μια πληροφορία, και έπειτα παραδίδει την απάντηση στον αιτούντα. Ένα καλό API βοηθά στην ευκολότερη ανάπτυξη ενός προγράμματος προσφέροντας τα δομικά στοιχεία.

Το API έδωσε την δυνατότητα στους προγραμματιστές να αναπτύσσουν πιο γρήγορα τις εφαρμογές τους, επειδή δεν χρειάζεται να ξεκινήσουν τον κώδικα τους από το μηδέν χάρη στην επαναχρησιμοποίηση των API. Δεν είναι όλα τα API ίδια μεταξύ τους· υπάρχουν διαφορετικές κατηγορίες, η καθεμία εκ των οποίων βασίζεται σε διαφορετικά πρωτόκολλα, λειτουργίες και επίπεδα πρόσβασης, γι’ αυτό και είναι σημαντικό να επιλεχθεί ο σωστός τύπος ανάλογα με τις απαιτήσεις του χρήστη.



Σχεδιάγραμμα 2.4 - 1 - Επικοινωνία Διακομιστή με Πελάτη

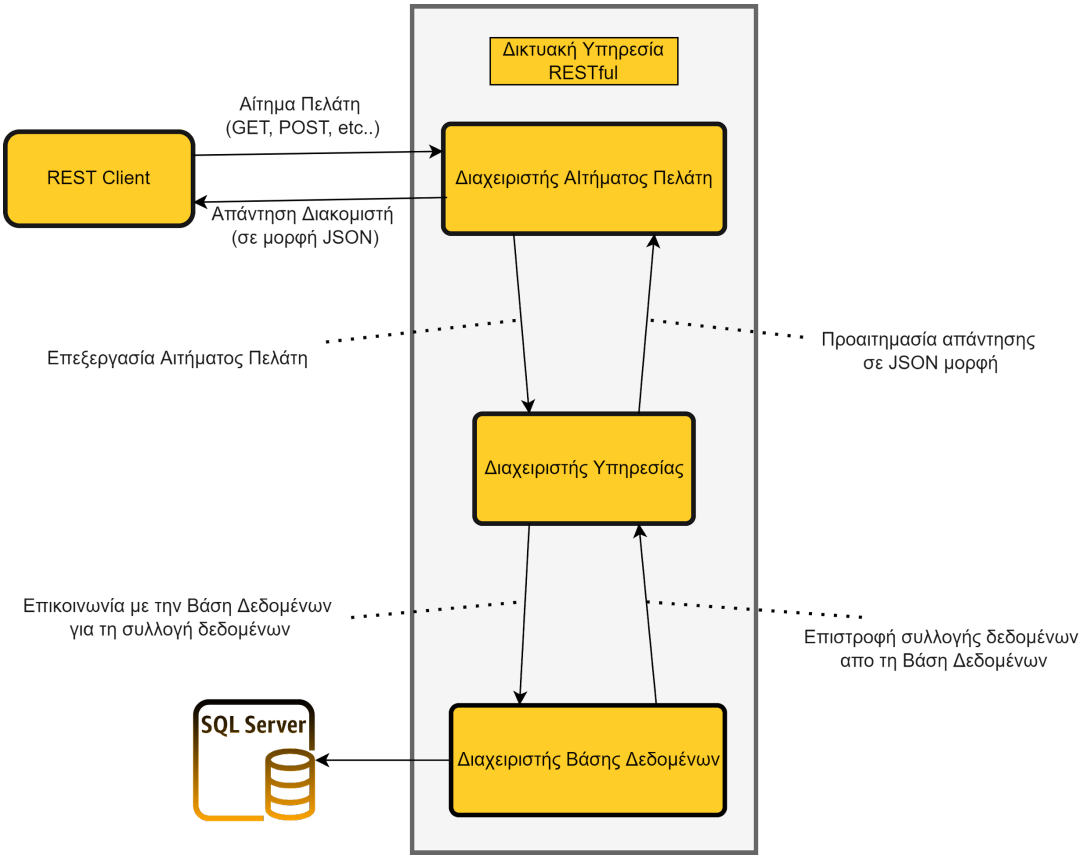
Στην συγκεκριμένη περίπτωση το osTicket βασίζεται στην REST αρχιτεκτονική. Το REST *(Representational State Transfer)* μεταφράζεται ως «αναπαράσταση κατάστασης μεταφοράς» και αποτελεί ένα σύνολο οδηγιών για επεκτάσιμα, ελαφριά και εύχρηστα API. Τα REST API παρέχουν έναν πλήρη οδηγό για την εξοικείωση με τη χρήση τους. Συνοπτικά οι οδηγίες είναι:

- Διαχωρισμός Client-Server: Όλες οι αλληλεπιδράσεις πελάτη-διακομιστή πρέπει να έχουν τη μορφή αιτήματος από τον πελάτη, ακολουθούμενο από απάντηση από τον διακομιστή. Οι διακομιστές δεν μπορούν να ζητήσουν και οι πελάτες δεν μπορούν να απαντήσουν.
- Uniform Interface: Όλα τα αιτήματα και οι απαντήσεις πρέπει να χρησιμοποιούν το HTTP ως πρωτόκολλο επικοινωνίας και να διαμορφώνονται με συγκεκριμένο τρόπο, ώστε να διασφαλίζεται η

συμβατότητα μεταξύ οποιουδήποτε πελάτη και οποιουδήποτε διακομιστή. Οι απαντήσεις του διακομιστή μορφοποιούνται σε JSON.

- Stateless: Κάθε αλληλεπίδραση πελάτη-διακομιστή είναι ανεξάρτητη από κάθε άλλη αλληλεπίδραση. Ο διακομιστής δεν αποθηκεύει δεδομένα από αιτήματα πελατών και δεν θυμάται τίποτα από προηγούμενες αλληλεπιδράσεις.
- Layered System: Τα αιτήματα και οι απαντήσεις πρέπει πάντα να μορφοποιούνται με τον ίδιο τρόπο, ακόμη και όταν περνούν από ενδιάμεσους διακομιστές μεταξύ του πελάτη και του API.
- Cache: Οι απαντήσεις διακομιστή πρέπει να υποδεικνύουν εάν ένας παρεχόμενος πόρος μπορεί να αποθηκευτεί προσωρινά από τον πελάτη και για πόσο χρονικό διάστημα.

Εάν ακολουθηθούν οι παραπάνω οδηγίες σωστά, τα REST API είναι κατάλληλα για χρήση προσφέροντας γρήγορες, εύκολες και ασφαλείς μεταφορές δεδομένων.



Σχεδιάγραμμα 2.4 - 2 - Βασική δομή RESTful API

3. ΑΝΑΣΚΟΠΗΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

Συνοπτικά, ένα Σύστημα Υποστήριξης & Αναφοράς Σφαλμάτων *(Support Ticketing System)* επιτρέπει στους χρήστες του να εκφράσουν οποιοδήποτε ζήτημα τους σε μορφή Ticket, και προσφέρει δυνατότητες διαχείρησης των Ticket από τους αρμόδιους του συστήματος. Το σύστημα παρακολουθεί την κατάσταση κάθε Ticket, καθώς τα μέλη του προσωπικού υποστήριξης εργάζονται για την επίλυση του προβλήματος.

Η αγορά διαθέτει πολλά Συστήματα Υποστήριξης & Αναφοράς Σφαλμάτων, και η πολυπλοκότητά τους, οι δυνατότητες και το κόστος τους είναι κριτήρια που θα καθορίσουν εν τέλει την επιτυχία τους. Μερικά από αυτά τα εργαλεία είναι το Zendesk, το Zoho Desk, το Freshdesk, το HappyFox, το WordPress Advanced Ticket System, το AzureDesk και πολλά άλλα.

3.1. ΕΝΑΛΛΑΚΤΙΚΕΣ ΕΠΙΛΟΓΕΣ ΣΥΣΤΗΜΑΤΟΣ ΑΝΑΦΟΡΑΣ ΣΦΑΛΜΑΤΩΝ

Ένα παράδειγμα ενός τέτοιου εργαλείου, πέρα από το osTicket που χρησιμοποιήθηκε γι'αυτή την εργασία, είναι το Zendesk, όπως αναφέρθηκε παραπάνω. Το Zendesk είναι μια πλατφόρμα εξυπηρέτησης πελατών στον τομέα του IT, και πέρα από τις βασικές λειτουργίες δημιουργίας και ελέγχου Ticket, προσφέρει μια μεγάλη ποικιλία από επιπλέον λειτουργίες που επεκτείνουν τις δυνατότητες του.

Συγκεκριμένα διαθέτει Live chat για την επικοινωνία των χρηστών που έχουν αναφέρει τυχόν πρόβλημα με τους πράκτορες που προσπαθούν να το επιλύσουν. Επίσης διαθέτει Chatbot που, μέσω αυτοματοποιημένων μηνυμάτων, ενημερώνει τους χρήστες για εξελίξεις σχετικά με τα προβλήματα που ανέφεραν.
Ένα τρίτο χρήσιμο feature που διαθέτει είναι Macro που δημιουργούν Ticket preset, τα οποία είναι χρήσιμα για την γρήγορη και εύκολη αναφορά προβλημάτων, χωρίς να χρειάζεται η δημιουργία Ticket απο την αρχή.

Αντιστοίχως το osTicket διαθέτει λειτουργίες που το καθιστούν ελκυστικό, όπως για παράδειγμα η δημιουργία To-Do List για τους πράκτορες, που διευκολύνει την οργάνωση διαχείρισης των προβλημάτων που προκύπτουν. Διαθέτει επίσης τη δυνατότητα εκχώρησης, μεταφοράς και παραπομπής Ticket σε άλλα τμήματα ή πράκτορες για την επίλυσή τους. Πραγματοποιείται έλεγχος της κατάστασης της περάτωσης, ώστε να αποφευχθούν προβλήματα οργάνωσης, και προσφέρεται η δυνατότητα προσθήκης προσαρμοζόμενων ζητούμενων πεδίων για τη δημιουργία Ticket με βάση τις επιθυμίες των διαχειριστών, και άλλα.

3.2. ΑΝΤΙΣΤΟΙΧΕΣ ΕΦΑΡΜΟΓΕΣ ΚΙΝΗΤΩΝ ΣΥΣΚΕΥΩΝ

Για την ικανοποιητική εξυπηρέτηση χρηστών μέσω τέτοιων συστημάτων απαιτείται η προσαρμογή τους στο χρήστη και όχι το αντίστροφο· πρέπει δηλαδή να παρέχεται στο χρήστη η δυνατότητα αναφοράς σφαλμάτων, οποιαδήποτε στιγμή και χωρίς περιορισμούς. Για το λόγο αυτό, η δημιουργία μιας εφαρμογής

για κινητά κρίνεται απαραίτητη, μιας και ο χρήστης δεν είναι υποχρεωμένος να έχει πρόσβαση σε υπολογιστή ανά πάσα στιγμή.

Το Zendesk, όπως αναφέρθηκε παραπάνω, προσφέρει υποστήριξη μέσω εφαρμογών για λειτουργικά Android και iOS, επεκτείνοντας τη λειτουργικότητά του σε χρήστες που βρίσκονται εν κινήσει. Μία απο τις δυνατότητες που διαθέτει η φορητή έκδοση του παραπάνω εργαλείου είναι η δυνατότητα προσθήκης ειδοποιήσεων προς τους πράκτορες για την ανάθεση καινούριων Ticket προς επίλυση. Προσφέρεται επίσης η εύκολη δημιουργία Ticket και η εύκολη προεπισκόπισή τους. Επιπλέον, διαθέτει ιδιαίτερα απλή πλοήγηση στο επιθυμητό Ticket, με τη χρήση πεδίου αναζήτησης αλλά και με τη χρήση υπερσυνδέσμων μέσω email. Τέλος, διαθέτει τη δυνατότητα χρήσης τρίτων εφαρμογών για την επικοινωνία μεταξύ των πρακτόρων και των χρηστών.

Το Zendesk, όπως και δύο εκ των τριών διαθέσιμων πακέτων εγκατάστασης του osTicket, είναι επί πληρωμή. Αν και προτιμήθηκε η επιλογή της δωρεάν έκδοσης του osTicket, όσον αφορά την ανάπτυξη της εφαρμογής αυτής οι επί πληρωμής εκδόσεις μοιράζονται ένα κοινό στοιχείο:. τη δυνατότητα Server Hosting στις προσωπικές πλατφόρμες Cloud Hosting των Zendesk και osTicket.

Έχοντας υπ' όψιν τα παραπάνω, επιλέχθηκε η ανοιχτού κώδικα εγκατάσταση του osTicket σε προσωπικό server του συγγραφέα, λόγω του μηδενικού κόστους και προσωπικής συμπάθειας προς εφαρμογές ανοιχτού κώδικα. Συγκεκριμένα μπορεί να γίνει host είτε μέσω docker είτε απευθείας σε λογισμικό.

3.3. ΕΝΑΛΛΑΚΤΙΚΟΙ ΤΡΟΠΟΙ HOSTING

Ο πρώτος τρόπος εγκατάστασης του osTicket σε προσωπικό server, όπως και αναφέρθηκε παραπάνω, ειναι η χρήση Docker. Το Docker είναι πλατφόρμα λογισμικού ανοιχτού κώδικα που υλοποιεί εικονικοποίηση (Virtualization) σε επίπεδο λειτουργικού συστήματος. Το Docker προσφέρει αυτοματοποιημένες διαδικασίες, για την ανάπτυξη εφαρμογών σε απομονωμένες Περιοχές Χρήστη, που ονομάζονται Software Containers.

Τα containers αυτα έχουν αρκετά θετικά, όπως το πολύ μικρό τους μέγεθος, η ταχύτητά τους και η χρήση λιγότερων πόρων συγκριτικά με εγκατάσταση σε εικονική μηχανή. Ένα από τα αρνητικά που έχουν είναι ότι αν μια εφαρμογή είναι σχεδιασμένη να εκτελείται σε Windows container, δεν μπορεί να εκτελεστεί σε άλλο λειτουργικό σύστημα όπως Linux.

Όσον αφορά την εγκατάσταση απευθείας σε λογισμικό, ιδανικά θα αφιερωνόταν μία ολόκληρη εγκατάσταση λογισμικού στο osTicket, η οποία κατά προτίμηση θα εκτελούνταν εικονοποιημένη *(virtualized)* πάνω σε κάποιον υπερεπόπτη *(Hypervisor)*. Υπάρχουν δύο βασικοί τρόποι χρήσης Hypervisor, ο TYPE-1 και ο TYPE-2 Hypervisor.

Καταρχάς, ένας HyperVisor είναι ένας τύπος λογισμικού υπολογιστή, που δημιουργεί και εκτελεί εικονικές μηχανές. Είναι μια τεχνική που επιτρέπει την συνύπαρξη πολλαπλών λειτουργικών συστημάτων σε έναν Host ταυτόχρονα.

Η πρώτη επιλογή ήταν η χρήση Type 1 HyperVisor, ή αλλιώς *"Bare Metal HyperVisor"*, καθώς εκτελείται απευθείας πάνω στο Host σύστημα, χωρίς την

ανάγκη ύπαρξης κάποιου λειτουργικού συστήματος. Αυτό του επιτρέπει την αυτοδιαχείρηση των πόρων του υλικού συστήματος, και ως αποτέλεσμα να τους εκμεταλλεύεται πλήρως.

Αντίθετα, σε Type 2 HyperVisor, το οποίο προαπαιτεί υπάρχον λειτουργικό σύστημα, οι πόροι είναι περιορισμένοι και έχει συγκριτικά χειρότερες επιδόσεις από τον TYPE 1 HyperVisor. Στην προκειμένη εργασία, έχοντας υπ'όψιν το μικρό φόρτο εργασίας που θα είχε ο Server, την έλλειψη υλικού για dedicated hosting μέσω Type1, και συνυπολογίζοντας την οικειότητα χρήσης Virtual Machines, επιλέχθηκε η χρήση Type 2 Hypervisor για την εγκατάσταση του osTicket.

3.4. ΕΝΑΛΛΑΚΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ ΔΙΕΠΑΦΗΣ OSTICKET

Μία επιπλέον εφαρμογή που αξίζει να αναφερθεί είναι η "*osTicket Admin*", υλοποιημένη απο την εταιρια ανάπτυξης λογισμικού Red4sis, με στόχο την πλατφόρμα του λογισμικού Android. Η συγκεκριμένη εφαρμογή δίνει παρόμοιες δυνατότητες με αυτές που παρέχει η παρούσα εφαρμογή.

Πιο συγκεκριμένα, το osTicket Admin προσφέρει τη δυνατότητα δημιουργίας και ελέγχου Ticket, την δυνατότητα εισόδου του χρήστη μετά τον έλεγχο των στοιχείων του, καθώς και την ενημέρωσή του μέσω push notifications. Προσφέρεται επίσης η δυνατότητα ελέγχου Ticket με 3 διαφορετικά φίλτρα. Το "*open*" για τα μη επιλυμένα Ticket, το "*closed*" για τα επιλυμένα και το "*my Tickets*" για τα Ticket που έχουν δημιουργηθεί απο τον συγκεκριμένο λογαριασμό.

Το osTicket Admin "κρύβει" τη λειτουργία της δημιουργίας Ticket πίσω από έξτρα χρέωσεις, εμπεριέχει διαφημίσεις και ο κώδικας υλοποίησης δεν είναι ανοιχτού κώδικα. Αντίθετα, η εφαρμογή της παρούσας διπλωματικής είναι εντελώς δωρεάν, δεν περιέχει διαφημίσεις και έχει ως μελλοντικό στόχο την μεταφορά σε μορφή ανοιχτού κώδικα.

4. ΜΕΘΟΔΟΛΟΓΙΑ ΕΡΕΥΝΑΣ

Για τη δημιουργία της εφαρμογής αυτής χρειάστηκε πρώτα να εγκατασταθούν κάποια βασικά εργαλεία: το Android Studio IDE για την συγγραφή του κώδικα και το osTicket Ticketing System για την παροχή των λειτουργιών του, συγκεκριμένα την δημιουργία και έλεγχο Ticket.

Η αρχική εγκατάσταση του osTicket δοκιμάστηκε σε λειτουργικό σύστημα Windows, άλλα σύντομα έγινε εμφανές ότι αυτή η προσέγγιση ήταν ασκόπως περιπλοκή και γεμάτη με πιθανά μελλοντικά προβλήματα. Για αυτό το λόγο τελικά επιλέχθηκε ένα λειτουργικό σύστημα βασισμένο σε Linux Kernel.

Έχοντας βρεθεί αντιμέτωποι με τα προβλήματα που προκύπτουν από την προσθήκη τέτοιου συστήματος σε ήδη χρησιμοποιούμενη εγκατάσταση λογισμικού, την δυσκολία διόρθωσης σφαλμάτων εγκατάστασης και αρχικοποίησης τέτοιων συστημάτων, και την έλλειψη δυνατότητας εύκολης επαναφοράς, η εγκατάσταση λειτουργικού συστήματος εκτελέστηκε σε εικονική μηχανή *(Virtual Machine - VM)* πάνω σε υπερεπόπτη *(Hypervisor)*.

Η προσέγγιση αυτή απομονώνει την εγκατάσταση του osTicket χωρίς να το απομονώνει απαραιτήτως από το εξωτερικό δίκτυο, επιτρέπει προσθήκες, ρυθμίσεις και αλλαγές γνωρίζοντας πως δεν θα επηρεάσει κάποια προυπάρχουσα λειτουργία, και προσφέρει δυνατότητα δημιουργίας και επιστροφής σε αντίγραφα ασφαλείας.

Ως Hypervisor επιλέχθηκε το VirtualBox, ως λειτουργικό σύστημα η Linux διανομή *(distribution - distro)* Ubuntu, και πάνω σε αυτά εγκαταστάθηκε το osTicket. Η εγκατάσταση του osTicket είναι μια πολύπλοκη διαδικασία, η οποία θα αναλυθεί παρακάτω.

Για την επικοινωνία της Εφαρμογής με τον osTicket Server χρησιμοποιήθηκε το API του osTicket, πραγματοποιώντας κλήσεις τύπου RESTful API. Το API αυτό περιλαμβάνει τα απαραίτητα PHP αρχεία που περιγράφουν την λειτουργία δημιουργίας Ticket, αλλά δεν περιλαμβάνει τα αρχεία που περιγράφουν την λειτουργία ελέγχου λεπτομερειών και κατάστασης ήδη υπάρχοντων Ticket.

Όντας λογισμικό ανοιχτού κώδικα, συγκεκριμένα διαμοιρασμένο υπό τις συνθήκες της Γενικής Δημόσιας Άδειας GNU *(Generic Public Liscense - GPL)*, υπάρχουν δεσμευμένοι κλάδοι *(commited branches)* στο Github του λογισμικού αυτού που προτείνουν και περιλαμβάνουν πλήρεις υλοποιήσεις για την λειτουργία ελέγχου Ticket, οι οποίες προστέθηκαν στον osTicket Server αυτής της εφαρμογής.

Επιπλέον για την επικοινωνία της εφαρμογής με τον osTicket Server χρειάζεται μια βιβλιοθήκη διαχείρισης API Calls. Αρχικά δοκιμάστηκε η χρήση της βιβλιοθήκης Volley, ανεπτυγμένη από την Google, αλλά λόγω δυσκολιών στην ανάπτυξη της λειτουργίας αποστολής αρχείων στον osTicket Server αντικαταστάθηκε από την βιβλιοθήκη Retrofit2.

Ο λόγος που έδειχνε η Retrofit πολλά υποσχόμενη ήταν η συγκριτικά πιο ολοκληρωμένη και προσεγγίσιμη υποστήριξη αποστολής πακέτων JSON.

Αντίθετα, η δημιουργία πακέτων JSON μέσω της Volley, και συγκεκριμένα στην μορφή που απαιτεί το api του osTicket, έδειχνε υπερβολικά περιπλοκή στην ανάπτυξή της και με τόσο μεγάλη έκταση κωδικά ώστε να τον καθιστά δυσανάγνωστο.

Οι πρώτες δοκιμές με την Retrofit2 έδειχναν εξίσου απογοητευτικές, με πολλαπλά σφάλματα και λανθασμένο formatting των XML πακέτων που στέλνει η εφαρμογή στον server μέσω του RESTful API. Ακολούθησαν πειραματισμοί δημιουργίας διαχειριστή συνδέσεων HTTP και αντικειμένων JSON από το μηδέν, αλλά ευτυχώς επιπλέον πειραματισμοί με την Retrofit2 οδήγησαν σε επιτυχία.

Όσον αφορά την ανάπτυξη του περιβάλλοντος διεπαφής, η αρχή έγινε με ταχύρρυθμη ανάπτυξη κώδικα χωρίς να αφιερωθεί ιδιαίτερη προσοχή στην εμπειρία του χρήστη. Συνεχίζοντας την ανάπτυξη έγινε προφανές πως το αποτέλεσμα ήταν πλήρως δυσνόητο και δύσχρηστο για τον μέσο χρήστη. Για αυτό το λόγο άλλαξε η νοοτροπία σχεδιασμού ώστε να αντιστοιχεί με το πρότυπο ανθρωποκεντρικού σχεδιασμού.

Το πρότυπο ανθρωποκεντρικού σχεδιασμού βασίζεται στην συλλογή και ανάλυση κριτικών και εμπειριών από τους χρήστες, και την χρήση αυτών των πληροφοριών για τη σχεδίαση και την δημιουργία πολυμεσικών εφαρμογών. Θέτοντας προγραμματιστές, φίλους και γνωστούς ως δοκιμαστές, η ανάπτυξη της διεπαφής χρήστη προχώρησε με συνεχείς αλλαγές και επαναλήψεις βάσει των κριτικών που είχαν συλλεχθεί σε κάθε προηγούμενο κύκλο ανάπτυξης.

Είναι λοιπόν προτιμότερο να προβαίνουμε σε αξιολόγηση κάθε σταδίου ανάπτυξης της εφαρμογής και να κάνουμε τυχόν αλλαγές που υποδεικνύονται από τους χρήστες, αντί να ολοκληρώσουμε την εφαρμογή και στο τέλος να διαπιστώσουμε πως δεν ικανοποιεί το κοινό στο οποίο απευθύνεται.

Συγκεκριμένα στην αρχική υλοποίηση, λόγω του ότι βασίστηκε στη γνώση και άνεση των προγραμματιστών με το κατασκεύασμα τους, έλειπαν ανταποκρίσεις ολοκλήρωσης βημάτων, όπως η επισύναψη αρχείου, και δικλείδες ασφαλείας όπως η δυνατότητα αποστολής ελλειπών πακέτων. Τα δύο αυτά παραδείγματα επιλύθηκαν υλοποιόντας προεπισκόπηση τυχόν αρχείων που έχει επισυνάψει ο χρήστης και άρνηση αποστολής ελλιπών πακέτων ενημερώνοντας τον χρήστη για το τυχόν σφάλμα.

5. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΕΠΙΣΚΟΠΗΣΗ ΕΦΑΡΜΟΓΗΣ

Η εφαρμογή που δημιουργήθηκε είναι ένα εργαλείο για έξυπνες συσκευές λογισμικού Android, που εκμεταλλεύεται ένα ήδη υπάρχον σύστημα για την αναφορά σφαλμάτων. Το σύστημα αυτό ονομάζεται osTicket και είναι ένα Support Ticketing System. Η επίτευξη της επικοινωνίας του client με τον osTicket Server έγινε με τη χρήση RESTFul API Calls. Η εφαρμογή υποστηρίζει τις δύο βασικότερες λειτουργίες του συστήματος αυτού, τη δημιουργία και τον έλεγχο Ticket. Ενέργειες "κλειδιά" για την υλοποίηση αυτής της εφαρμογής στα οποία αξίζει να γίνει αναφορά ήταν η δημιουργία ενός τοπικού Server, και η εγκατάσταση και παραμετροποίηση του osTicket που γίνεται παρακάτω, αλλά και η χρήση της βιβλιοθήκης Retrofit για την επιτυχημένη επικοινωνία του Server με τον Client.

5.1. ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΑΡΧΙΚΟΠΟΙΗΣΗ OSTICKET

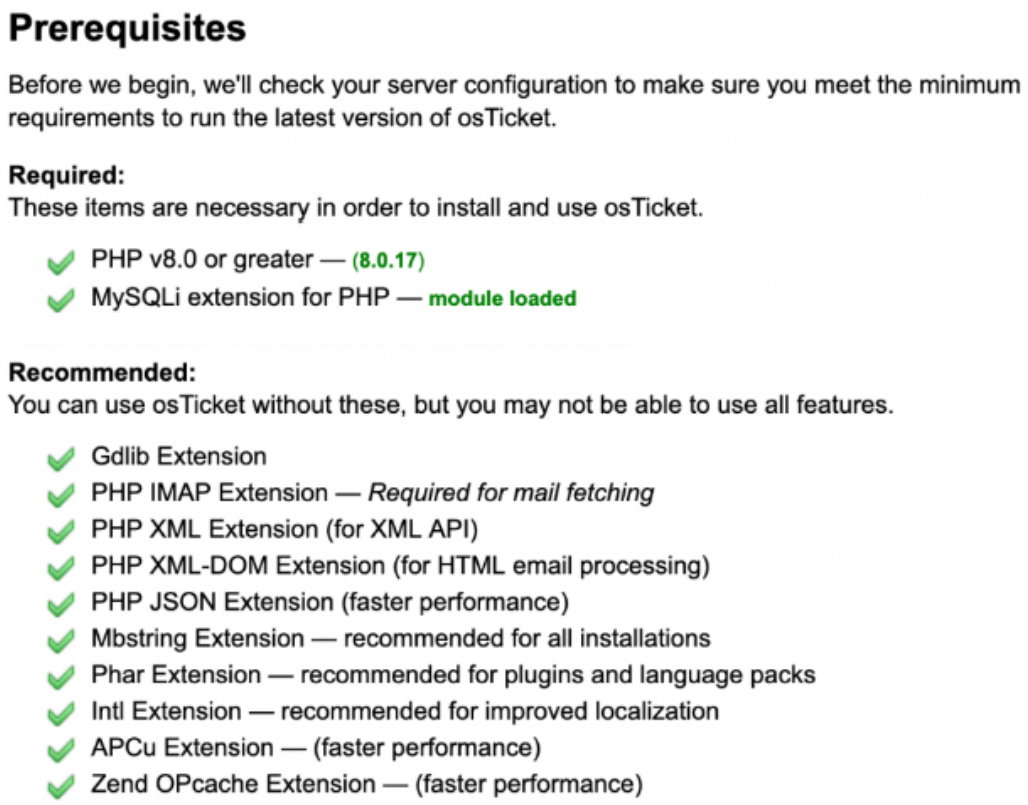
Για την εγκατάσταση του osTicket προαπαιτείται η εγκατάσταση της PHP 8.0 και της MYSQL 5.0 στον Web Server. Την δουλειά του Web Server εκτελεί λειτουργικό σύστημα βασισμένο σε Linux, που ιδανικά τρέχει σε κάποιον Hypervisor. Ταυτόχρονα κρίνεται απαραίτητη η ύπαρξη μιας βάσης δεδομένων, π.χ. MySQL, με έγκυρο χρήστη, κωδικό πρόσβασης και όνομα κεντρικού υπολογιστή κατά την εγκατάσταση, ενώ ταυτόχρονα ο χρήστης της βάσης αυτής πρέπει να έχει πλήρη δικαιώματα πρόσβασης και επεξεργασίας δεδομένων.

Αρχικά, το λογισμικό σύστημα που έχει επιλεχθεί θα πρέπει να είναι πλήρως ενημερωμένο. Έχοντας αυτό σαν βάση, θα πρέπει να γίνουν κάποιες κρίσιμες ρυθμίσεις: θα πρέπει να θέσουμε ένα ταιριαστό και μοναδικό όνομα δικτύου, μια στατική μοναδική IP και να αντιστοιχηθούν μεταξύ τους. Συγκεκριμένη προσοχή θα πρέπει να δώσουμε στην δομή του δικτύου μας, ώστε η IP αυτή να είναι αφιερωμένη στον εν λόγω Server και επιπλέον ο DNS Server του δικτύου να έχει ενημερωθεί για την προαναφερθείσα αντιστοίχηση.

Σαν επόμενο βήμα, προτού να μπορεί να εγκατασταθεί το osTicket πρέπει να εγκατασταθούν διάφορα προαπαιτούμενα προγράμματα. Συγκεκριμένα και πρώτα απ'όλα θα χρειαστεί ένα εργαλείο δημιουργίας και διαχείρισης βάσης δεδομένων. Για τη διπλωματική αυτή χρησιμοποιήθηκαν τα MYSQL και MARIADB. Αφού εγκατασταθούν πρέπει να δημιουργηθεί μια βάση δεδομένων για το osTicket και ένας χρήστης με δικαιώματα ανάγνωσης και επεξεργασίας της βάσης αυτής.

Με το τέλος της εγκατάστασης του διακομιστή της βάσης και παραμετροποίησης των δικαιωμάτων του χρήστη, ακολουθεί η εγκατάσταση του Apache Web Server. Δουλειά του Apache είναι να διαχειρίζεται αιτήματα πρωτοκόλλου μεταφοράς υπερκειμένου *(HTTP)* και να τα διαβιβάζει στον κατάλληλο διακομιστή, στη συγκεκριμένη περίπτωση το osTicket. Τέλος πρέπει να εγκατασταθεί η γλώσσα προγραμματισμού PHP και οι διάφορες επεκτάσεις της, διότι είναι απαραίτητες για τη σωστή λειτουργία του osTicket.

Με όλα τα απαραίτητα στοιχεία εγκατεστημένα, ακολουθεί η εγκατάσταση και παραμετροποίηση του συστήματος osTicket. Αφότου ολοκληρωθεί η εγκατάσταση, γίνεται μεταφορά στο osTicket Admin με τη χρήση της IP που τέθηκε νωρίτερα όπου εμφανίζεται η σελίδα που εντοπίζει την ύπαρξη όλων των απαραίτητων και προτεινόμενων εξωτερικών εργαλείων. Έαν όλα τα προηγούμενα βήματα έχουν εκτελεσθεί επιτυχώς, η σελίδα αυτή θα δείχνει όπως παρακάτω:



Εικόνα 5.1 - 1 - Πλήρης επιτυχημένος έλεγχος extensions osTicket

Έχοντας ολοκληρώσει επιτυχώς την εγκατάσταση και παραμετροποίηση του osTicket και των προαπαιτούμενων στοιχείων του, η ιστοσελίδα του osTicket θα παρουσιάσει την σελίδα βασικών ρυθμίσεών του, και με την ολοκλήρωσή τους θα είναι έτοιμο για χρήση.

5.2. RETROFIT

Η βιβλιοθήκη Retrofit αναπτύχθηκε από την Square Inc. και είναι ένας type-safe REST client για Android και Java. Η Retrofit μετατρέπει ένα HTTP API σε ένα Java Interface και είναι αρκετά απλή στη χρήση. Ουσιαστικά επιτρέπει να αντιμετωπιστούν οι κλήσεις API ως απλές κλήσεις Java methods, ορίζοντας τις διευθύνσεις URL και τους τύπους των παραμέτρων αιτήματος/απόκρισης *(request/response)* ως Java classes. Προσφέρει τη δυνατότητα υλοποίησης ενός σύγχρονου ή ασύγχρονου αιτήματος HTTP χρησιμοποιώντας τη βιβλιοθήκη OkHttp στον απομακρυσμένο Web Server, ενώ διαχειρίζεται ολόκληρη την κλήση δικτύου, όπως επίσης την ανάκτηση και τη μεταφόρτωση δεδομένων JSON ή άλλων δομημένων δεδομένων, μέσω μιας υπηρεσίας web που βασίζεται σε REST.

Στη Retrofit μπορούμε να ρυθμίσουμε ποιος μετατροπέας *(converter)* χρησιμοποιείται για τη σειριοποίηση δεδομένων *(data serialization)*. Συνήθως για JSON χρησιμοποιειται ο GSon converter, αλλά μπορεί να χρησιμοποιηθεί και κάποιος από τους προσαρμοσμένους μετατροπείς για την επεξεργασία XML ή άλλων πρωτοκόλλων. Για να εργαστούμε με τη Retrofit χρειαζόμαστε τα ακόλουθα τρια πράγματα:

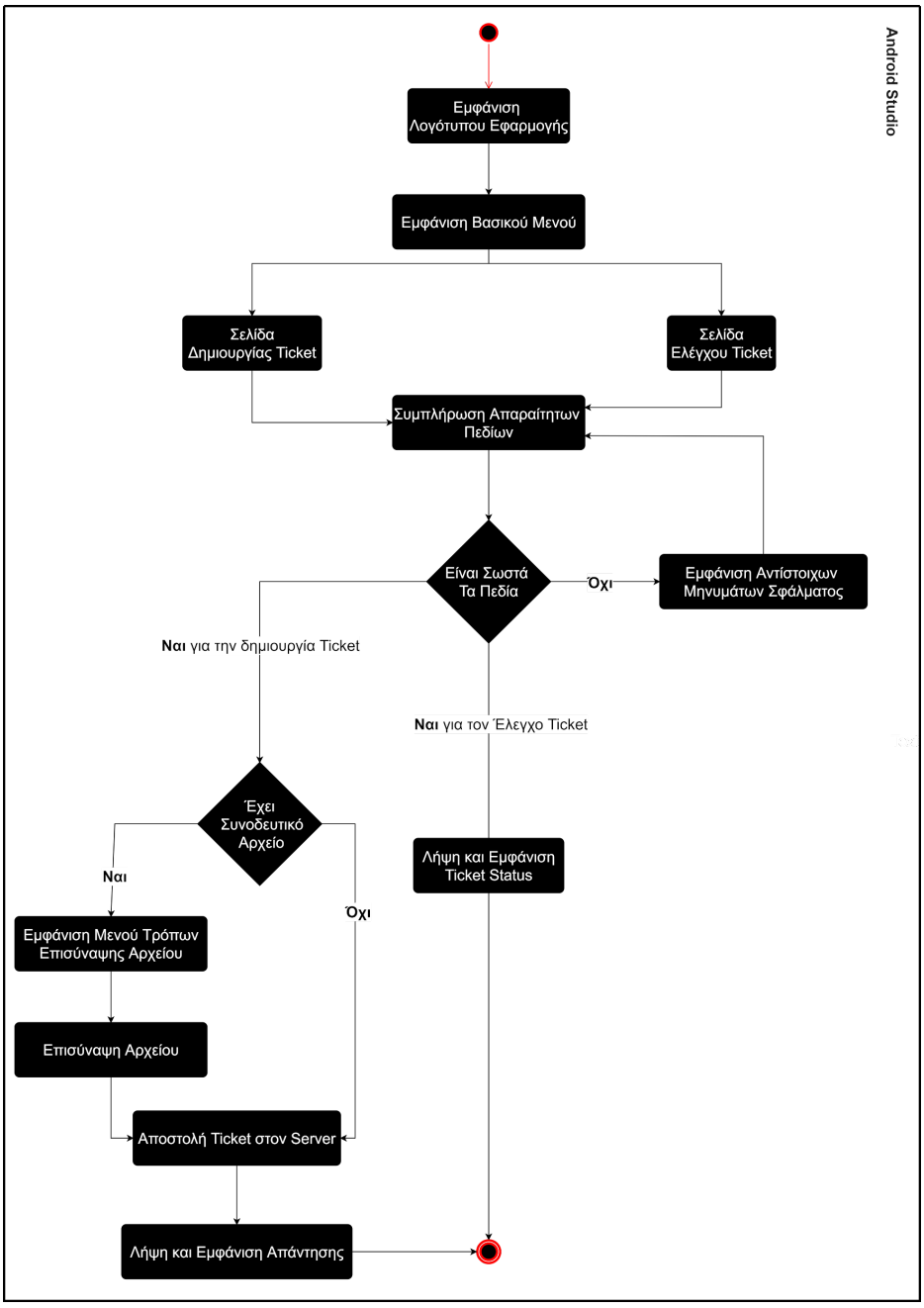
- Την κλάση μοντέλου που χρησιμοποιείται ως μοντέλο JSON.
- Ένα Interface που καθορίζει τις πιθανές λειτουργίες HTTP.
- Ένα Instance της Κλάσης Retrofit.Builder - που χρησιμοποιεί το Interface και το Builder API για να επιτρέψει τον καθορισμό του ENDPOINT της διεύθυνσης URL για τις λειτουργίες HTTP.

Κάθε μέθοδος ενός Interface αντιπροσωπεύει μια πιθανή κλήση API και πρέπει να έχει έναν σχολιασμό HTTP, όπως GET ή POST, για να καθορίσει τον τύπο αιτήματος και τη σχετική διεύθυνση URL. Η επιστρεφόμενη τιμή περικλείει την απόκριση *(response)* σε ένα αντικείμενο Call με τον τύπο του αναμενόμενου αποτελέσματος.

5.3. LOTTIEFILES

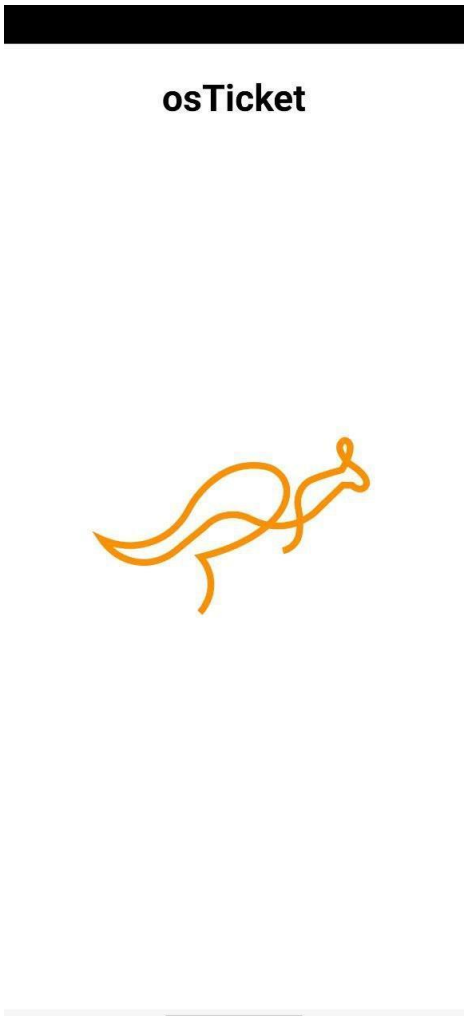
Αρχικά τα Lottie είναι ένας τύπος αρχείου κινουμένων σχεδίων που βασίζεται στον τύπο αρχείου του JSON. Το LottieFiles, πέρα από μία μεγάλη πλατφόρμα που προσφέρει μία τεράστια γκάμα από ελαφριά και *Scalable Animations*, είναι και μία βιβλιοθήκη ανεπτυγμένη από την Airbnb Design, υπεύθυνη για την προσθήκη αυτών των *Animations* στο project αλλά και για τηνπαραμετροποίηση τους. Τα *Animations* αυτά είναι μορφής SVG και είναι βασισμένα στην γλώσσα XML. Το συγκεκριμένο της παρούσας διπλωματικής επιλέχθηκε από τις δωρεάν και ανοιχτών δικαιωμάτων επιλογές της πλατφόρμας LottieFiles.

5.4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ



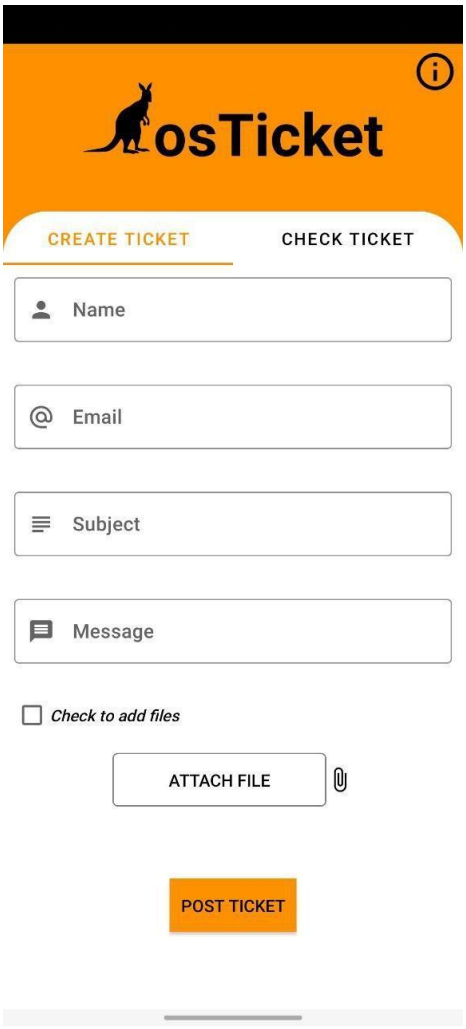
Σχεδιάγραμμα 5.4 - 1 - Λογική Εφαρμογής

Κατά την εκκίνηση της εφαρμογής εμφανίζεται η αρχική οθόνη, η οποία περιλαμβάνει το animated λογότυπο της εφαρμογής όπου μετά το πέρας 2,5 δευτερολέπτων εμφανίζεται η κυρίως οθόνη.

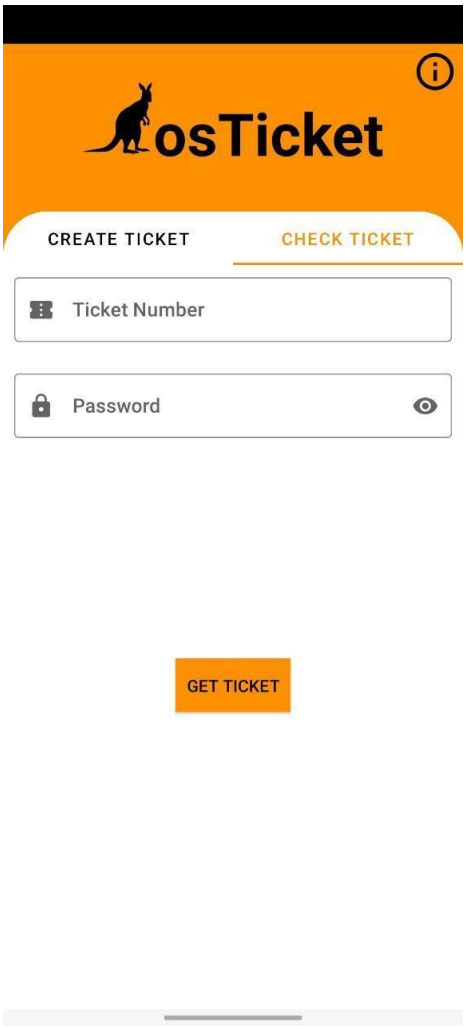


Εικόνα 5.4 - 1: Animated App Logo

Στην κυρίως οθόνη εμφανίζεται το βασικό μενού της εφαρμογής, που χωρίζεται σε δύο μέρη με την χρήση Tabs. Το πρώτο μέρος είναι για την δημιουργία Ticket, ενώ το δεύτερο για τον έλεγχο Ticket.

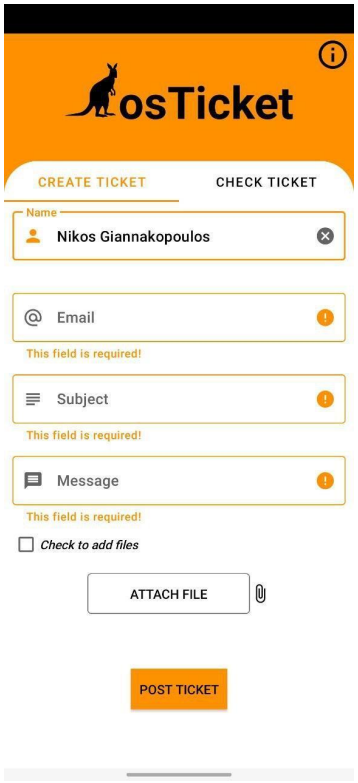


Εικόνα 5.4 - 2:
Σελίδα δημιουργίας Ticket

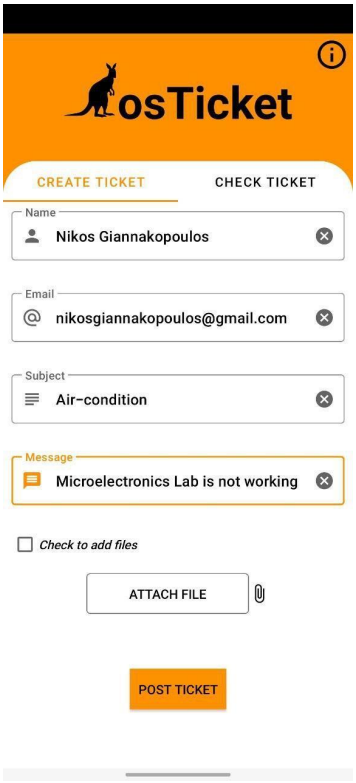


Εικόνα 5.4 - 3:
Σελίδα ελέγχου Ticket

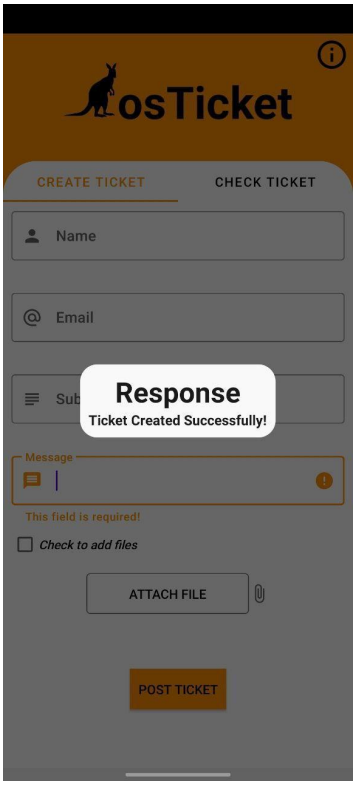
Στο tab *Create Ticket*, όπως φαίνεται και στην *Εικόνα 5.4 - 2*, ο χρήστης οφείλει να συμπληρώσει όλα τα πεδία για να μπορέσει να δημιουργήσει επιτυχώς ενα Ticket. Στην περίπτωση που κάποιο από τα πεδία είναι κενό, εάν ο χρήστης προσπαθήσει να δημιουργήσει το Ticket πατώντας το κουμπί POST TICKET, η εφαρμογή δεν εκτελεί τη λειτουργία δημιουργίας και εμφανίζει το αντίστοιχο μήνυμα σφάλματος.



Εικόνα 5.4 - 4 Σφαλματα μη συμπληρωσης όλων των πεδίων

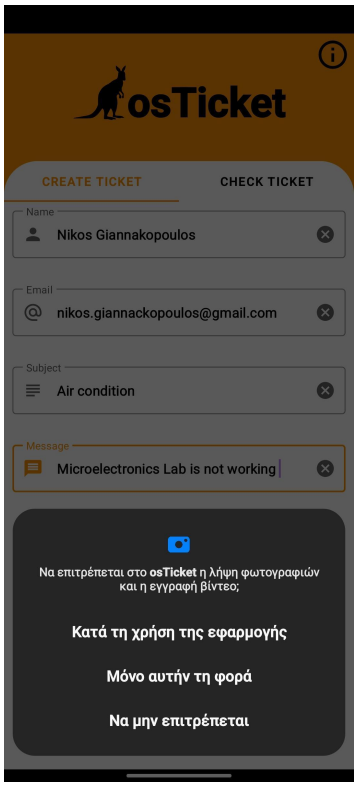


Εικόνα 5.4 - 5 Επιτυχημένη συμπληρωση όλων των πεδίων

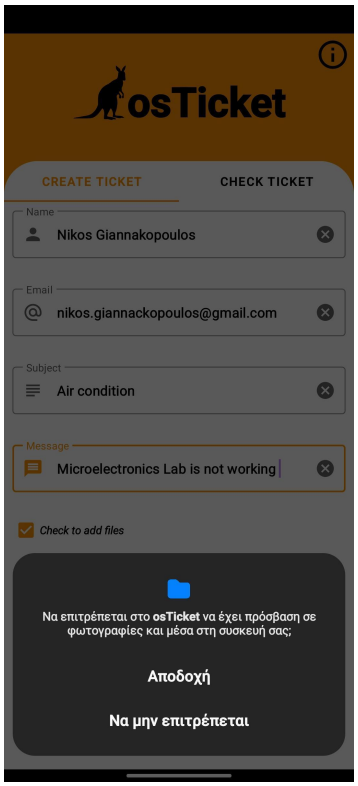


Εικόνα 5.4 - 6 Επιτυχημένη απάντηση διακομιστή

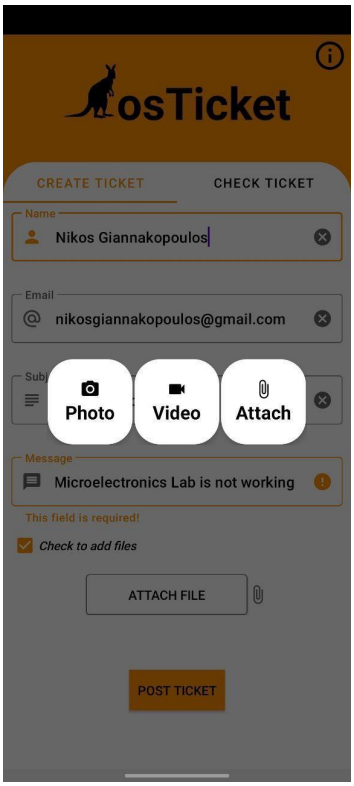
Ο χρήστης, επίσης, έχει τη δυνατότητα να επισυνάψει ένα αρχείο εικόνας ή βίντεο σχετικό με την αναφορά σφάλματος που θέλει να κάνει. Η δυνατότητα αυτή παρέχεται στον χρήστη από τη στιγμή που θα επιλέξει το σχετικό checkbox για την προσθήκη αρχείων. Στην πρώτη προσπάθεια του χρήστη να επισυνάψει ένα αρχείο ζητούνται τα απαραίτητα δικαιώματα χρήσης της κάμερας για τη λήψη φωτογραφιών και εγγραφής βίντεο, αλλά και σχετική πρόσβαση στη συσκευή για την ανάκτηση αρχείων.



Εικόνα 5.4 - 7 Δικαίωμα χρήσης κάμερας

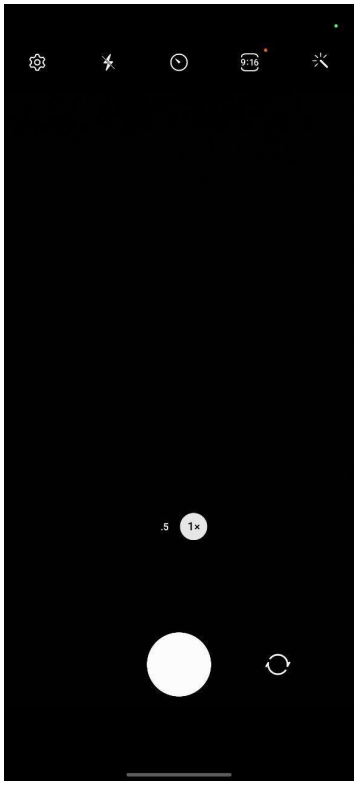


Εικόνα 5.4 - 8 Δικαίωμα πρόσβασης στη συσκευή

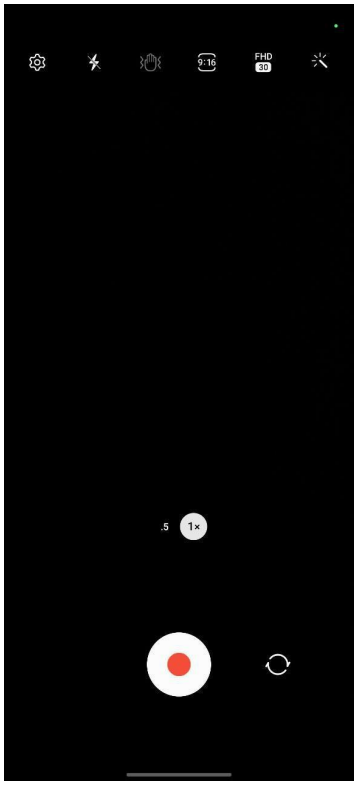


Εικόνα 5.4 - 9 Σελίδα επιλογής επισύναψης αρχείου

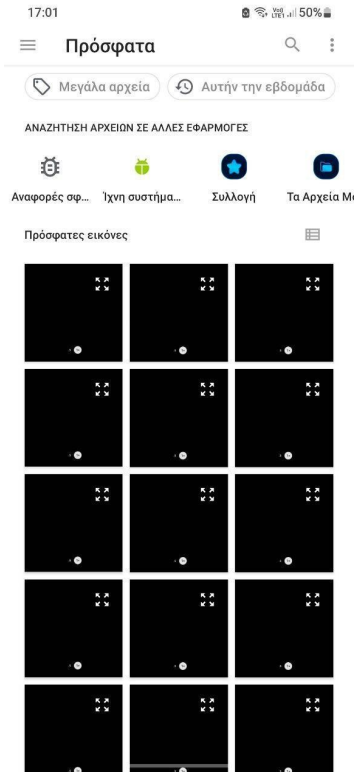
Το αρχείο αυτό μπορεί είτε να δημιουργηθεί χρησιμοποιώντας τις ενσωματωμένες κάμερες της έξυπνης συσκευής για να τραβήξει φωτογραφία ή για να καταγράψει βίντεο, είτε να επισυναφθεί ανοίγοντας τον περιηγητή αρχείων, επιτρέποντας στον χρήστη να επιλέξει το αρχείο που επιθυμεί.



Εικόνα 5.4 - 10
Άνοιγμα κάμερας για λήψη φωτογραφίας

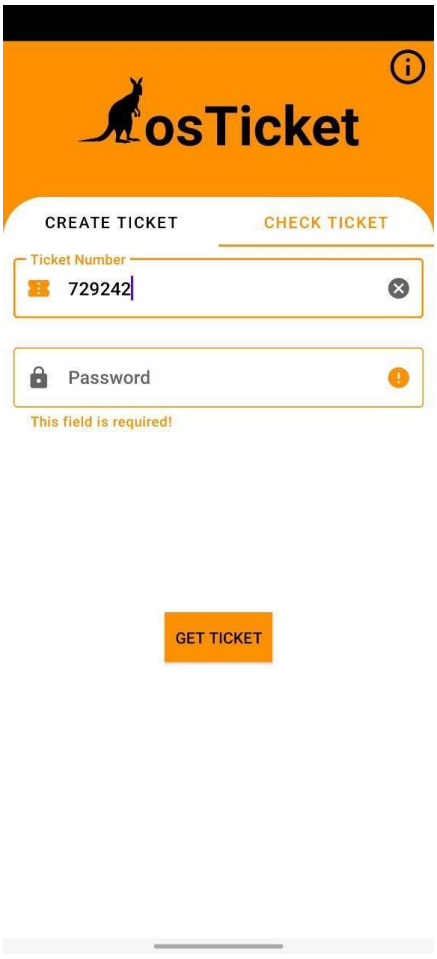


Εικόνα 5.4 - 11
Άνοιγμα κάμερας για καταγραφή βίντεο

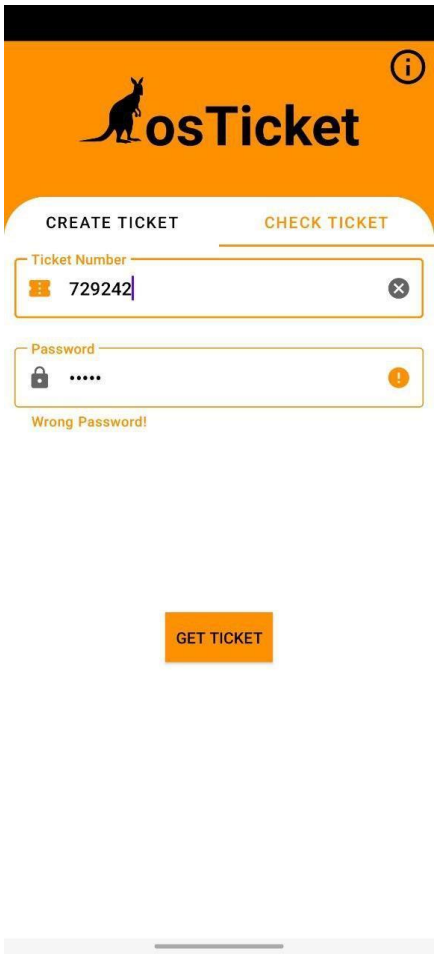


Εικόνα 5.4 - 12
Άνοιγμα περιηγητή αρχείων για επιλογή

Στο tab *Check Ticket,* αντίστοιχα, ο χρήστης οφείλει πάλι να συμπληρώσει όλα τα πεδία για να μπορέσει να ελέγξει επιτυχημένα ενα Ticket. Την λειτουργία του ελέγχου των Ticket δεν μπορεί να την εκτελέσει οποιοσδήποτε χρήστης, αλλά μόνο ο διαχειριστης και το προσωπικό, καθώς απαιτείται κωδικός τον οποίον θα γνωρίζουν μόνο αυτοί.

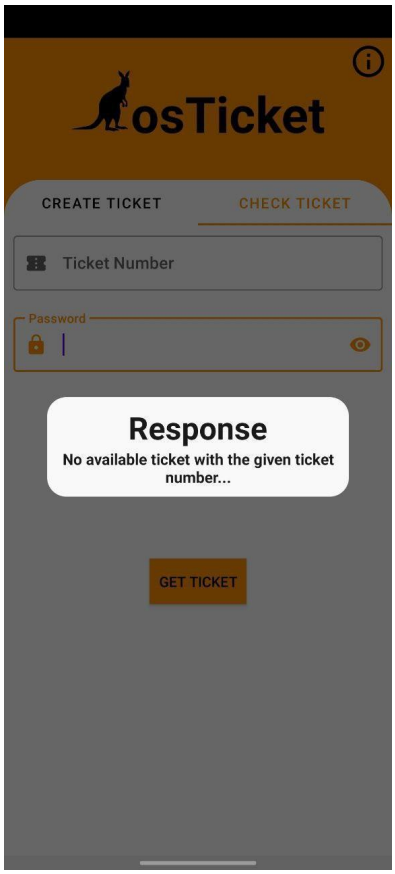


Εικόνα 5.4 - 13: Σφαλμα μη συμπληρωσης κάποιου πεδίου

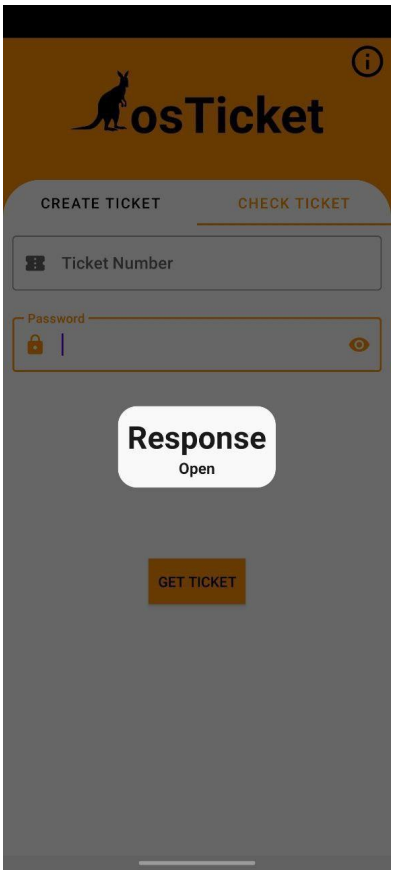


Εικόνα 5.4 - 14: Σφαλμα λανθασμένου κωδικού

Εάν κάποιο από τα πεδία είναι κενό ή ο κωδικός είναι λάθος, και ο χρήστης προσπαθήσει να ελέγξει Ticket πατώντας το κουμπί GET TICKET, η εφαρμογή δεν εκτελεί την λειτουργία του ελέγχου και εμφανίζει το αντίστοιχο μήνυμα σφάλματος. Εάν ο κωδικός είναι σωστός και υπάρχει ο συγκεκριμένος αριθμός δελτίου *(Ticket Number)*, εμφανίζεται η κατάσταση του δελτίου *(Ticket Status)*, ενώ στην περίπτωση που δεν υπάρχει το συγκεκριμένο *Ticket Number*, τότε εμφανίζεται το αντίστοιχο ενημερωτικό μήνυμα.



Εικόνα 5.4 - 15:
Ανταπόκριση μη διαθέσιμου Ticket



Εικόνα 5.4 - 16:
Ανταπόκριση κατάστασης Ticket

5.5. ΑΝΑΛΥΣΗ ΤΜΗΜΑΤΩΝ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ

Στην προηγούμενη ενότητα έγινε η παρουσίαση της εφαρμογής, όπως αυτή εκτελείται σε μια συσκευή, και ο τρόπος με τον οποίο ο χρήστης αλληλεπιδρά με αυτήν. Σε αυτή την ενότητα θα γίνει παρουσίαση και επεξήγηση ορισμένων τμημάτων κώδικα, ώστε να δοθεί μια γενικότερη και πιο ολοκληρωμένη εικόνα όσον αφορά την υλοποίηση της εφαρμογής.

Θα παρουσιαστούν τμήματα κώδικα απο τα αρχεία layout, που είναι υπεύθυνα για την χωρική ταξινόμηση των επιμέρους στοιχείων των διαφορετικών σελιδών που είδαμε παραπάνω. Τον κώδικα των αρχείων layout θα συνοδεύουν τα κομμάτια κώδικα από τα αντίστοιχα activities, τα οποία περιγράφουν τον τρόπο λειτουργίας όλων των features του αντίστοιχου UI κάθε οθόνης. Τέλος, θα αναλυθούν όλες οι *Java Classes* που χρησιμοποιήθηκαν για την επιτυχή επικοινωνία του Client με τον osTicket Server.

Πιο συγκεκριμένα, όπως παρουσιάστηκε παραπάνω στο κομμάτι της παρουσίασης, οι δύο βασικές λειτουργίες της εφαρμογής αυτής χωρίζονται σε δυο διαφορετικά Tabs, προσφέροντας τη δυνατότητα πλοήγησης σε αυτά μέσω σειρομένων κινήσεων χειρός *(Swipe Hand Gestures)*, εμφανίζοντας το σωστό Fragment κάθε φορά.

Η δημιουργία των δύο αυτών διαφορετικών Tabs, δηλαδή της δημιουργίας και ελέγχου Ticket, πραγματοποιήθηκε μέσω του TabLayout, ενώ η σύνδεση του σωστού Fragment με το σωστό Tab, καθώς και η δυνατότητα πλοήγησης μέσω

Swipe Hand Gestures, υλοποιήθηκε με τη χρήση του *ViewPager*. Στα στοιχεία αυτά του UI δόθηκε "ζωή" με τον εξής τρόπο:

Δημιουργήθηκε μια κλάση *ViewPagerAdapter* για τον ορισμό του επιθυμητού πλήθους αλλά και την εμφάνιση του σωστού Fragment ανάλογα με το position. Στη συνέχεια χρησιμοποιήθηκε ένας Fragment Manager, καθώς ήταν απαραίτητος για την διαχείριση των λειτουργιών που μπορούν να πραγματοποιηθούν σε ένα Fragment. Αρχικοποιήθηκε ο *ViewPagerAdapter* με τον προαναφερθέν Fragment Manager, προσφέροντας όλες τις λειτουργίες και τα στάδια ενός Android Activity χρησιμοποιώντας την μέθοδο *getLifecycle()*. Αφού προστεθούν τα δύο Tabs, το position του εκάστοτε επιλεγμένου Tab αποθηκεύεται και στέλνεται σαν όρισμα στον ViewPagerListener για την εμφάνιση του σωστού Fragment. Ο κώδικας αυτός εμφανίζεται αναλυτικά στο *Παράρτημα Κώδικα 1, 2 & 3*.

Μία επιπλέον σημαντική λειτουργία της εφαρμογής είναι η δυνατότητα επισύναψης αρχείου εικόνας ή βίντεο για να συνοδεύσει τα απαραίτητα στοιχεία κατα τη δημιουργία του Ticket. Το αρχείο προς αποστολή μπορεί να δημιουργηθεί κατά τη χρήση της εφαρμογής, αφού προσφέρεται η δυνατότητα ενεργοποίησης της ενσωματωμένης κάμερας της συσκευής για λήψη φωτογραφίας ή καταγραφή βίντεο. Φυσικά, αν η λήψη φωτογραφίας ή καταγραφή βίντεο έχει γίνει εκτός της εφαρμογής αυτής, και το αρχείο που έχει καταγραφεί είναι αποθηκευμένο στη συσκευή του χρήστη, προσφέρεται η δυνατότητα πλοήγησης στα αρχεία του μέσα από την εφαρμογή για να επιλέξει το αρχείο που επιθυμεί.

Για να πραγματοποιηθεί η λειτουργία χρήσης της κάμερας για λήψη φωτογραφίας ή καταγραφή βίντεο απαιτείται η μεταπήδηση από το Activity του βασικού Menu στο Activity της κάμερας με την χρήση του *ActivityResultLauncher*. Το *ActivityResultLauncher* θα μεταφέρει την εφαρμογή στο νέο Activity, θα λάβει το αποτέλεσμα του συγκεκριμένου Activity και θα το επιστρέψει στον χρήστη. Αν έγινε μεταφορά στο Activity της κάμερας για λήψη φωτογραφίας, το αποτέλεσμα που αναμένεται θα είναι μια φωτογραφία, και αντίστοιχα για την καταγραφή βίντεο.

Για την επισύναψη υπάρχοντων αρχείων μέσω του File Explorer χρησιμοποιήθηκε ένα *ActivityResultLauncher*. Λειτουργεί όπως και παραπάνω, με την διαφορά ότι αντί να ανοίγει ένα καινούριο Activity, ανοίγει τον File Explorer και προτρέπει τον χρήστη να επιλέξει ένα αρχείο τύπου δεδομένων που έχει οριστεί στο κώδικα. Στον κώδικα της συγκεκριμένης εφαρμογής ορίζονται μόνο αρχεία JPG, PNG και MP4, όπως φαίνεται στο *Παράρτημα Κώδικα 6α*.

Οι τρεις αυτές περιπτώσεις έχουν κοινά στοιχεία: συγκεκριμένα το όνομα του αρχείου, τον τύπο του και το file path του, πληροφορίες απαραίτητες για το upload των αρχείων στο server (*Παράρτημα Κώδικα 6β*). Τέλος, αν το θελήσει ο χρήστης, μπορεί να εμφανίσει την επιλογή του αγγίζοντας το εικονίδιο της επισύναψης, για να σιγουρευτεί ότι όντως επέλεξε αυτό που ήθελε. Ο κώδικας αυτός αναλύεται στο *Παράρτημα - Κώδικα 4, 5 & 6*.

Έπειτα παρουσιάζεται η υλοποίηση τριών συναρτήσεων πολύ σημαντικών για την εφαρμογή. Η πρώτη είναι υπεύθυνη για την εύρεση του URI των εικόνων, είτε

είναι τραβηγμένες εκείνη τη στιγμή μέσω της ενσωματωμένης κάμερας της συσκευής που έχει γίνει embed στην εφαρμογή είτε έχουν επιλεχθεί απο τον File Explorer. Η δημιουργία της κρίνεται απαραίτητη διότι μετά την επιλογή του χρήστη ή μετά το snapshot δεν γινόταν η αποθήκευση του image path, με αποτέλεσματο URI να χάνεται. Ο κώδικας της παραπάνω συνάρτησης φαίνεται στο *Παράρτημα Κώδικα 7*.

Στη συνέχεια, εμφανίζεται η συνάρτηση υπεύθυνη για τη μετατροπή του URI του αρχείου στο Original Path στη συσκευή, η οποία είναι στοιχείο κλειδί για την επόμενη συνάρτηση.Το path αυτό χρησιμοποιείται για να εντοπιστεί το αρχείο και να το μετατρέψει στο Base64 Format που απαιτεί το osTicket. Ο κώδικας των παραπάνω συναρτήσεων φαίνεται στο *Παράρτημα Κώδικα 8 & 9*.

Η τελευταία συνάρτηση, για την οποία θα γίνει αναφορά παρακάτω, έχει να κάνει με την ολοκλήρωση επιτυχημένης δημιουργίας Ticket. Τα στοιχεία που θα γίνουν upload στον server θα πρέπει να είναι της μορφής JSON για να γίνουν δεκτά, και αυτό περιλαμβάνει και το επιθυμητό αρχείο προς επισύναψη. Η διαδικασία που εκτελεί η συνάρτηση αυτή λοιπόν, είναι η μετατροπή του επιθυμητού επιλεγμένου αρχείου σε JSON format, που ακολουθείται από έναν έλεγχο ώστε να προστεθούν τα σωστά File Extension και Description.

Επίσης, πρέπει να μετατρέψουμε και τις τιμές των *textviews* σε JSON format, συν να ελεγχθεί αν ο χρήστης έχει επισυνάψει κάποιο αρχείο. Όλα αυτά "δένονται" σε ένα jsonObject το οποίο γίνεται upload στο server περικλείεται σε ένα αντικείμενο *RequestBody* με τα απαραίτητα headers ώστε να γίνει το HTTP

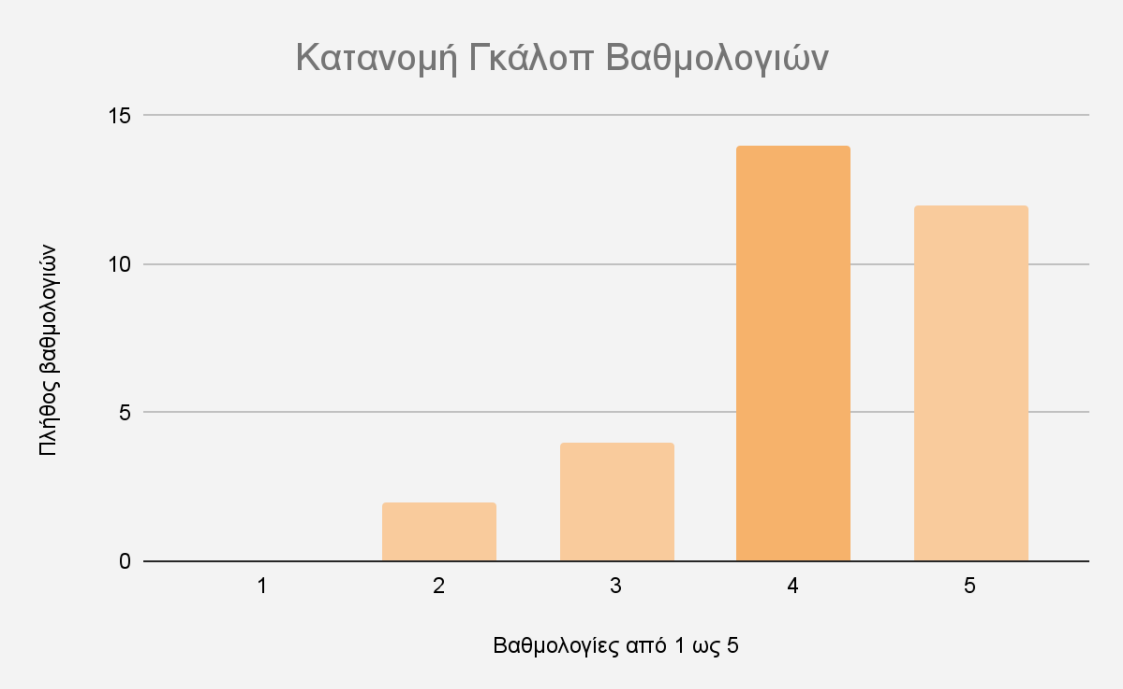
request. Έπειτα γίνεται ένα ασύγχρονο αίτημα μέσω της Retrofit, χρησιμοποιώντας ένα αντικείμενο *Call* με την μέθοδο enqueue, η οποία καλεί την μέθοδο POST που χρησιμοποιείται για τη δημιουργία Ticket. Η παραπάνω διαδικασία υλοποιήθηκε όπως φαίνεται στο *Παράρτημα Κώδικα 11*.

Τέλος, κατασκευάστηκε η POJO class η οποία αντιστοιχεί στο JSON format που χρειάζεται ένα Ticket προκειμένου να αποσταλεί επιτυχώς. Η δημιουργία αυτής της κλάσης-μοντέλου ήταν απαραίτητη για τη σωστή λειτουργία της επικοινωνίας του Server με τον Client, κυρίως για τον έλεγχο των Ticket. Έπειτα ορίστηκε το Interface που καθορίζει τις πιθανές λειτουργίες HTTP. Κάθε μέθοδος του Interface αντιπροσωπεύει μια πιθανή κλήση API, και πρέπει να έχει έναν σχολιασμό HTTP, όπως *GET* ή *POST*, για να καθορίσει τον τύπο αιτήματος και τα αντίστοιχα *ENDPOINTS* κάθε συνάρτησης. Η επιστρεφόμενη τιμή της συνάρτησης enqueue περικλείει την απόκριση *(response)* σε ένα αντικείμενο Call, το οποίο αντιστοιχεί στον τύπο του αναμενόμενου αποτελέσματος. Για να ολοκληρώσουμε τα απαραίτητα βήματα επιτυχημένης χρήσης της Retrofit δημιουργούμε ένα Instance της Κλάσης *Retrofit.Builder*, το οποίο χρησιμοποιεί το Interface και το Builder API για να επιτρέψει τον καθορισμό της σχετικής διεύθυνσης URL για τις λειτουργίες HTTP. Ο παραπάνω κώδικας παρουσιάζεται στο *Παράρτημα Κώδικα 12α, 12β,13, 14*.

6. ΑΞΙΟΛΟΓΗΣΗ ΕΦΑΡΜΟΓΗΣ

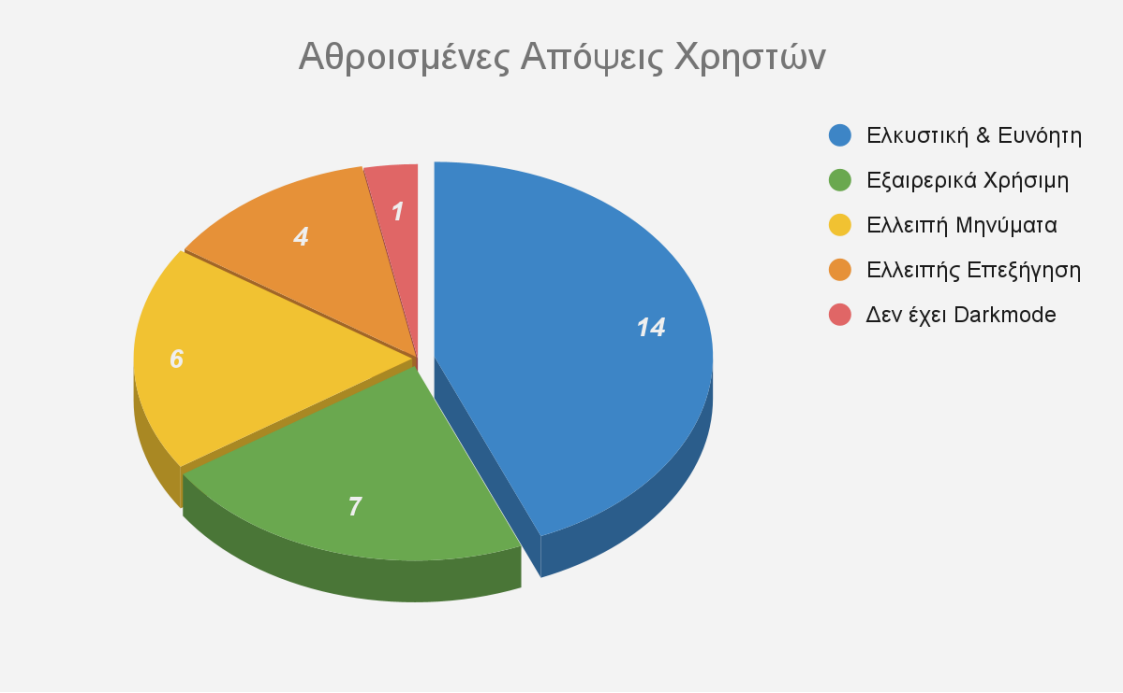
Αφότου υλοποιήθηκε η εφαρμογή, αφιερώθηκε χρόνος στο κομμάτι της αξιολόγησής της. Ζητήθηκε η άποψη φίλων, γνωστών και συμφοιτητών. Σε μία κλίμακα μέτρησης απο το 1 έως το 5, με το 1 να αντιπροσωπεύει τη χείριστη βαθμολογία/κριτική και το 5 να αντιπροσωπεύει την βέλτιστη, κλήθηκαν να αξιολογήσουν την εφαρμογή σύμφωνα με το πόσο ικανοποιημένοι έμειναν.

Παρακάτω παρουσιάζεται ένα σχεδιάγραμμα που απεικονίζει τις βαθμολογίες. Απο το σχεδιάγραμμα προκύπτει πως το μεγαλύτερο μέρος των χρηστών φάνηκε ικανοποιημένο, βαθμολογώντας την εφαρμογή με 4. Ο μέσος όρος των βαθμολογιών ήταν 4,1 στα 5.



Ενδεχομένως, βέβαια, η αξιολόγηση να μην ήταν πλήρως αντικειμενική, δεδομένου της έλλειψης ανωνυμίας αλλά και λόγω των φιλικών σχέσεων με μεγάλο μέρος των βαθμολογητών. Γι'αυτό το λόγο, οι βαθμολογητές προτράπηκαν στη προσθήκη επιπλέον σχολιασμού που να δικαιολογεί την τελική τους βαθμολογία.

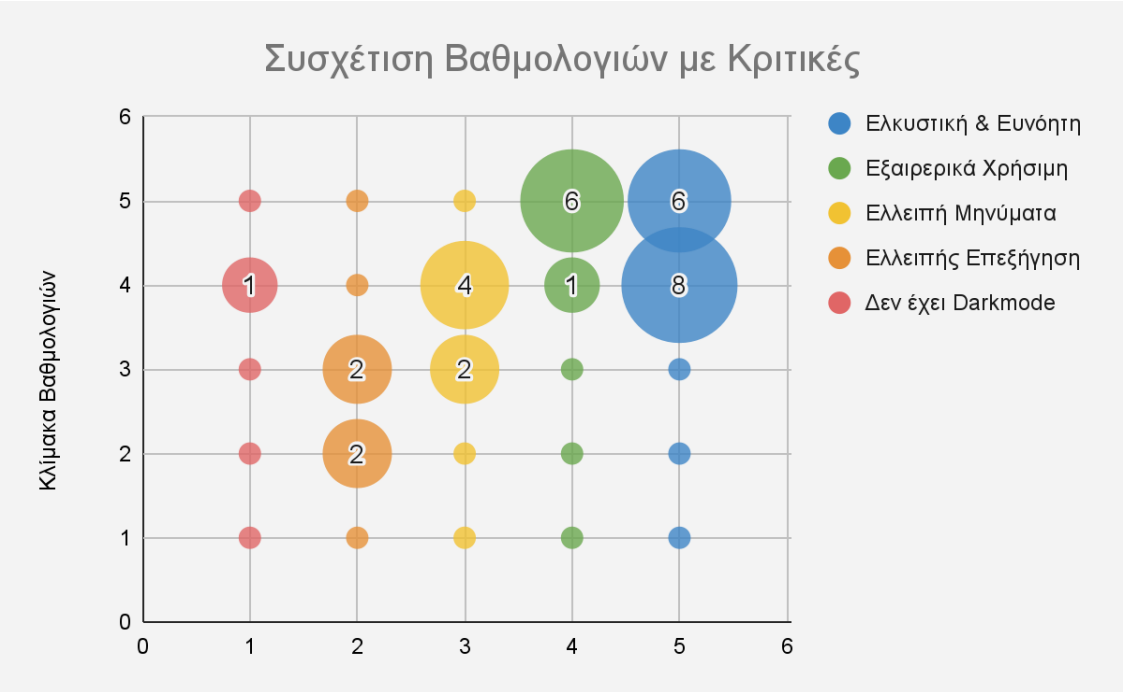
Όπως παρουσιάζεται στη συνέχεια, το παρακάτω σχεδιάγραμμα απεικονίζει το πλήθος κριτικών ομαδοποιημένο με βάση τους λόγους που κατέληξαν στα παραπάνω αποτελέσματα.



Η παραπάνω ομαδοποίηση έγινε με 5 διαφορετικά κριτήρια, που αντιστοιχούν στους λόγους της τελικής βαθμολογίας των χρηστών. Τα κριτήρια αυτά ήταν:

- Η αισθητική της
- Η χρησιμότητά της
- Ανεπαρκείς απαντήσεις
    - Ανταποκρίσεις (Responses) από την επικοινωνία με τον osTicket Server
- Ανεπαρκής καθοδήγηση
    - Επαρκή μηνύματα πληροφόρησης και σφάλματος για την καθοδήγηση του χρήστη σε κάθε λειτουργία της εφαρμογής
- Έλλειψη Dark Mode

Με βάση τις κριτικές που λήφθηκαν, ακόμα και αν το μέγεθος του δείγματος δεν ήταν επαρκές για να παραχθούν αντιπροσωπευτικά αποτελέσματα για την αγορά που προορίζεται, μπορεί να γίνει η υπόθεση ότι η εφαρμογή έχει τη σωστή προσέγγιση. Μεγαλύτερη βαρύτητα όμως δίνεται στις κριτικές με τη χαμηλότερη βαθμολογία και τους λόγους που επέφεραν το συγκεκριμένο αποτέλεσμα, καθώς θα γίνουν οι στόχοι των μελλοντικών επεκτάσεών της.



Τέλος, παρουσιάζεται το σχεδιάγραμμα συσχετισμού των τελικών βαθμολογιών όλων των χρηστών ξεχωριστά με τους σχολιασμούς τους πάνω στην εφαρμογή, κάτι που προσφέρει μια γενικότερη και πιο καθαρή εικόνα της κατάστασής της. Προσφέρει τη δυνατότητα οργάνωσης και διαχείρησης προτεραιοτήτων όσον αφορά τις επικείμενες αλλαγές και προσθήκες που θα βελτιώσουν σημαντικά την αξιολόγηση της εφαρμογής.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Αν και ο αρχικός στόχος του εργαλείου ήταν η χρήση από το προσωπικό και τους φοιτητές του Πανεπιστημίου Δυτικής Αττικής, μπορεί να χρησιμοποιηθεί και από οποιοδήποτε άλλο φορέα ή υπηρεσία με σκοπό την αναφορά βλαβών στους χώρους υποδομής και όχι μόνο.

Το πλεονέκτημα του είναι η διευκόλυνση που προσφέρει σε ομάδες, εταιρίες, οργανισμούς, με την άμεση ενημέρωση για την άρτια λειτουργία του συστήματος στο οποίο έχει εφαρμοστεί το εργαλείο αυτό. Ταυτόχρονα προσφέρεται η δυνατοτήτα σε απλούς χρήστες να συνεισφέρουν στην καλή συντήρηση του συστήματος, με αποτέλεσμα να παρακάμπτονται αργές διαδικασίες και να δίνονται άμεσες λύσεις σε σημαντικά προβλήματα όταν αυτά παρουσιάζονται, ενισχύοντας παράλληλα την αλληλεγγύη και την κοινωνική ευθύνη.

Όσον αφορά μελλοντικές επεκτάσεις της εφαρμογής, θα μπορούσαν να προστεθούν επιπλέον πεδία προς συμπλήρωση για την δημιουργία Ticket. Ακόμη, θα μπορούσε να δημιουργηθεί ένα login/sign up page, για να μην χρειάζεται η συμπλήρωση κωδικού στην περίπτωση που κάποιος αρμόδιος επιθυμεί τον έλεγχο κάποιου Ticket. Επιπλέον, θα μπορούσαν να προστεθούν πολλαπλοί τρόποι αναζήτησης Ticket είτε μέσω email είτε με πολλαπλά φίλτρα αναζήτησης, δυνατότητα αυτοματοποιημένων διαδικασιών ενημέρωσης του χρήστη για την εξέλιξη του εκάστοτεπροβλήματος, καθώς και δυνατότητα άμεσης επικοινωνίας, είτε μέσω της εφαρμογής είτε μέσω τρίτων εφαρμογών, μεταξύ του

χρήστη που ανέφερε το πρόβλημα και του πράκτορα που φροντίζει για την επίλυσή του.

Τέλος, αρνητικό της εφαρμογής που οφείλεται σε έλλειψη του osTicket API, είναι η ανάγκη ύπαρξης ξεχωριστού API KEY για κάθε διαφορετική συσκευή στο δίκτυο που είναι εγκατεστημένο το osTicket. Η άρρηκτη αυτή σύνδεση μεταξύ των παραγόμενων API KEYS και της συσκευής που σκοπεύει να αξιοποιήσει την εφαρμογή δυσκολεύει αρκετά την όλη διαδικασία, καθώς η έλλειψη ενός αυτοματοποιημένου τρόπου παραγωγής API KEYS αναγκάζει τους developers να δηλώνουν ρητά στο κώδικα το κάθε διαφορετικό API KEY, με αποτέλεσμα να υπάρχουν πολλαπλές εκδόσεις της εφαρμογής για κάθε διαφορετικό χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Επίσημο εγχειρίδιο χρήσης osTicket, από: https://docs.osticket.com/
2. Επίσημο εγχειρίδιο χρήσης Android Studio, από: https://developer.android.com/docs
3. Tanembaum Andrew S, Wetherall David J, (2011), Δίκτυα Υπολογιστών Πέμπτη Έκδοση, Εκδόσεις Κλειδάριθμος
4. Επίσημο εγχειρίδιο χρήσης βιβλιοθήκη Volley, από: https://google.github.io/volley/
5. Επίσημο εγχειρίδιο χρήσης βιβλιοθήκη Retrofit, από: https://square.github.io/retrofit/
6. Amalmagdy, ticket APIs (get ticket info, list agent or user tickets and post reply to ticket with new status), από: https://github.com/osTicket/osTicket/pull/4361/files
7. Οδηγός εγκατάστασης MariaDB σε Linux Ubuntu, από: https://www.digitalocean.com/community/tutorials/how-to-install-mariadb-on-ubuntu-22-04
8. Οδηγός εγκατάστασης osTicket σε Linux Ubuntu, από: https://computingforgeeks.com/how-to-install-osticket-on-ubuntu-linux/

9. Tayba Farooqui, Tauseef Rana, Fakeeha Jafari, Impact of Human-Centered Design Process (HCDP) on Software Development Process, από: https://ieeexplore.ieee.org/abstract/document/8680978/authors#authors
10. Αβούρης, Ν., Κατσάνος, Χ., Τσέλιος, Ν., & Μουστάκας, Κ. (2015). Εισαγωγή στην Αλληλεπίδραση Ανθρώπου-Υπολογιστή. ΣΥΝΔΕΣΜΟΣ ΕΛΛΗΝΙΚΩΝ ΑΚΑΔΗΜΑΪΚΩΝ ΒΙΒΛΙΟΘΗΚΩΝ.
11. Shneiderman, B., & Plaisant, C. Σχεδίαση Διεπαφής Χρήστη. Α. Τζιόλα, Θεσσαλονίκη.
12. Mark Masse, (2011), REST API Design Rulebook, εκδόσεις O'Reilly
13. Οδηγός επιλογής και σύγκρισης συστημάτων αναφοράς σφαλμάτων με βάση το Zendesk ένα απο τα επικρατέστερα συστήματα στην αγορά: https://www.zendesk.com/help-desk-software/ticketing-system/

ΠΑΡΑΡΤΗΜΑ - ΚΩΔΙΚΑΣ

```
<RelativeLayout
  ...
  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/top"
    android:background="@drawable/form_background">
    <com.google.android.material.tabs.TabLayout
      android:id="@+id/tabLayout"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="@drawable/form_background"
      app:tabIndicatorColor="#ff9100"
      app:tabSelectedTextColor="#ff9100"
      app:tabTextColor="@color/black" />
    <androidx.viewpager2.widget.ViewPager2
      android:id="@+id/viewPager2"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_below="@+id/tabLayout"
      android:layout_marginHorizontal="10dp"
      android:layout_marginVertical="5dp" />
  </RelativeLayout>
  ...
</RelativeLayout>
```

Κώδικας 1 - Ορισμός Tab Layout και View Pager

```
FragmentManager fragmentManager = getSupportFragmentManager();
viewPagerAdapter = new ViewPagerAdapter(fragmentManager,
                   getLifecycle());
viewPager2.setAdapter(viewPagerAdapter);

tabLayout.addTab(tabLayout.newTab()
         .setText("Create Ticket"));
tabLayout.addTab(tabLayout.newTab().setText("Check Ticket"));

tabLayout.addOnTabSelectedListener(
  new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
      viewPager2.setCurrentItem(tab.getPosition());
    }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {}

    @Override
    public void onTabReselected(TabLayout.Tab tab) {}
});

viewPager2.registerOnPageChangeCallback(new
ViewPager2.OnPageChangeCallback() {
  @Override
  public void onPageSelected(int position) {
    tabLayout.selectTab(tabLayout.getTabAt(position));
  }
});
```

Κώδικας 2 - Λειτουργικότητα Tab Layout και View Pager

```
public class ViewPagerAdapter extends FragmentStateAdapter {
  public ViewPagerAdapter(@NonNull FragmentManager fragmentManager,
                          @NonNull Lifecycle lifecycle) {
    super(fragmentManager, lifecycle);
  }

  @NonNull
  @Override
  public Fragment createFragment(int position) {
    if (position == 0) {
      return new CreateTicketFragment();
    }
    return new CheckTicketFragment();
  }

  @Override
  public int getItemCount() {
    return 2;
  }
}
```

Κώδικας 3 - Διαχείριση Fragments

```
activityResultLauncherTakePhoto = registerForActivityResult(
  new ActivityResultContracts.StartActivityForResult(), result -> {
    if (result.getResultCode() == RESULT_OK && result.getData()
        != null) {
      bitmap = (Bitmap) result.getData().getExtras().get("data");
      uri = getImageUri(getContext(), bitmap);
      filePath = getPathFromURI(uri);
      fileName = uri.getLastPathSegment();
      fileExtension = filePath.substring(filePath.lastIndexOf("."));
      ImageViewCompat.setImageTintList(attachedFile,
```

```
        ColorStateList.valueOf(Color.rgb(255, 145, 0)));
      videoForUpload.setVisibility(View.GONE);
      imageForUpload.setVisibility(View.VISIBLE);
      imageForUpload.setImageURI(uri);
      uri = null;
    } else {
      startActivity(new Intent(getActivity(), MainActivity.class));
    }
});
```

Κώδικας 4 - Άνοιγμα Κάμερας και αποθήκευση Εικόνας

```
activityResultLauncherCaptureVideo = registerForActivityResult(
  new ActivityResultContracts.StartActivityForResult(), result -> {
    if (result.getResultCode() == RESULT_OK && result.getData()
        != null) {
      uri = result.getData().getData();
      filePath = getPathFromURI(uri);
      fileName = uri.getLastPathSegment();
      fileExtension = filePath.substring(filePath.lastIndexOf("."));
      ImageViewCompat.setImageTintList(attachedFile,
        ColorStateList.valueOf(Color.rgb(255, 145, 0)));
      imageForUpload.setVisibility(View.GONE);
      videoForUpload.setVisibility(View.VISIBLE);
      videoForUpload.setVideoURI(uri);
      MediaController mediaController =
        new MediaController(getContext());
      videoForUpload.setMediaController(mediaController);
      mediaController.setAnchorView(videoForUpload);
      videoForUpload.start();
      uri = null;
    } else {
      startActivity(new Intent(getActivity(), MainActivity.class));
```

```
    }
});
```

Κώδικας 5 - Άνοιγμα Κάμερας και αποθήκευση Βίντεο

```
activityResultLauncherAttachFile = registerForActivityResult(
  new ActivityResultContracts.GetContent(), result -> {
    if (result == null) {
      startActivity(new Intent(getActivity(), MainActivity.class));
    } else {
      String mime =
        getContext().getContentResolver().getType(result);
      uri = result;
      if (mime.toLowerCase().contains("video")) {
        imageForUpload.setVisibility(View.GONE);
        videoForUpload.setVisibility(View.VISIBLE);
        videoForUpload.setVideoURI(uri);
        MediaController mediaController =
          new MediaController(getContext());
        videoForUpload.setMediaController(mediaController);
        mediaController.setAnchorView(videoForUpload);
        videoForUpload.start();
      } else if (mime.toLowerCase().contains("image")) {
        videoForUpload.setVisibility(View.GONE);
        imageForUpload.setVisibility(View.VISIBLE);
        imageForUpload.setImageURI(uri);
        try {
          bitmap = MediaStore.Images.Media.getBitmap(
            getContext().getContentResolver(), result);
          uri = getImageUri(getContext(), bitmap);
        } catch (IOException e) {
          e.printStackTrace();
        }
```

```
        }
        filePath = getPathFromURI(uri);
        fileName = uri.getLastPathSegment();
        fileExtension = filePath.substring(filePath.lastIndexOf("."));
        ImageViewCompat.setImageTintList(attachedFile,
          ColorStateList.valueOf(Color.rgb(255, 145, 0)));
        uri = null;
      }
});
```

Κώδικας 6 - Άνοιγμα περιηγητή για επιλογή αρχείου

```
attach_file = menuDialog.findViewById(R.id.attach_file);
attach_file.setOnClickListener(view13 ->
  activityResultLauncherAttachFile.launch("image/*;video/*"));
```

Κώδικας 6α - Εφαρμογή φίλτρου τύπου αρχείων

```
filePath = getPathFromURI(uri);
fileName = uri.getLastPathSegment();
fileExtension = filePath.substring(filePath.lastIndexOf("."));
```

Κώδικας 6β - Ανάκτηση χαρακτηριστικών από αρχεία

```
public Uri getImageUri(Context context, Bitmap bitmap) {
  String path = MediaStore.Images.Media.insertImage(
                context.getContentResolver(), bitmap, null, null);
  return Uri.parse(path);
}
```

Κώδικας 7 - Μετατροπή bitmap εικόνας σε uri

```
public String getPathFromURI(Uri uri) {
  Cursor cursor = getContext().getContentResolver().query(
                  uri, null, null, null, null);
  cursor.moveToFirst();
  int index = cursor.getColumnIndex(MediaStore.MediaColumns.DATA);
  return cursor.getString(index);
}
```

Κώδικας 8 - Μετατροπή uri στο original path - CreateTicketFragment.java

```
public static String getBase64FromFilePath(String path) {
  File file = new File(path);
  byte[] bytesArray = new byte[(int) file.length()];
  FileInputStream fis;
  try {
    fis = new FileInputStream(file);
    fis.read(bytesArray);
    fis.close();
  } catch (IOException e) {
    e.printStackTrace();
  }
  return Base64.getMimeEncoder().encodeToString(bytesArray);
}
```

Κώδικας 9 - Μετατροπή αρχείου σε base64 format

```
private JsonElement getFileToJsonFormat() {
  JsonParser parser = new JsonParser();
  JsonElement jsonElement = null;
  switch (fileExtension) {
    case JPEG:
      jsonElement = parser.parse("[{\"" + fileName
                    + fileExtension + JPEG_Description
                    + getBase64FromFilePath(filePath) + "\"}]");
      break;
    case PNG:
      // Ομοιώς για PNG
      break;
    case MP4:
      // Ομοιώς για MP4
      break;
  }
  return jsonElement;
}
```

Κώδικας 10 - Μετατροπή απο base64 format σε JSON

```
JsonObject ticket = new JsonObject();
ticket.addProperty("name", String.valueOf(
                    nameTextInputEditText.getText()));
ticket.addProperty("email", String.valueOf(
                    emailTextInputEditText.getText()));
ticket.addProperty("subject", String.valueOf(
                    subjectTextInputEditText.getText()));
ticket.addProperty("message", String.valueOf(
                    messageTextInputEditText.getText()));
if (attachFileActivation.isChecked())
  ticket.add("attachments", getFileToJsonFormat());

RequestBody body = RequestBody.create(
          okhttp3.MediaType.parse("application/json; charset=utf-8"),
          String.valueOf(ticket));

Call<ResponseBody> call = Client.getService().postTicket(body);
call.enqueue(new Callback<ResponseBody>() {
  @Override
  public void onResponse(@NonNull Call<ResponseBody> call,
                         @NonNull Response<ResponseBody> response) {
    if (response.isSuccessful()) {
      getResponse.setText("Ticket Created Successfully!");
      responseDialog.show();
    }
  }
  @Override
  public void onFailure(@NonNull Call<ResponseBody> call,
                        @NonNull Throwable t) {
    getResponse.setText(t.getMessage());
    responseDialog.show();
  }
});
```

Κώδικας 11 - Κατασκευή JSON Object και αποστολή του

```
public class Ticket {
  @SerializedName("ticket")
  @Expose
  private TicketData ticket;

  public TicketData getTicket() {
    return ticket;
  }

  public void setTicket(TicketData ticket) {
    this.ticket = ticket;
  }
}
```

Κώδικας 12α - POJO Class του JSON format του ticket

```
public class TicketData {

  @SerializedName("ticket_status")
  @Expose
  private String ticketStatus;

  public String getTicketStatus() {
    return ticketStatus;
  }

  public void setTicketStatus(String ticketStatus) {
   this.ticketStatus = ticketStatus;
  }
}
```

Κώδικας 12β - POJO Class του JSON format του ticket

```
public interface API {
  @Headers("X-API-Key: D32D58367C68FB923EB416F30276FFA1")
  @GET("tickets/ticketInfo")
  Call<Ticket> getTicket(@Query("ticketNumber") String ticketNumber);

  @Headers("X-API-Key: D32D58367C68FB923EB416F30276FFA1")
  @POST("tickets.json")
  Call<ResponseBody> postTicket(@Body RequestBody params);
}
```

Κώδικας 13 - Ορισμός μεθόδων για τα Api Calls

```
public class Client {
  public static Retrofit getInstance() {
    String API_BASE_URL = "http://192.168.2.200/api/http.php/";
    return new Retrofit.Builder().baseUrl(API_BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }

    public static API getService() {
      return getInstance().create(API.class);
    }
}
```

Κώδικας 14 - Retrofit builder instance με ορισμό του base URL