ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Οδηγός για τους φοιτητές του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

Του φοιτητή
Τριανταφυλλίδη Ραφαήλ
Αρ. Μητρώου: 154550

Επιβλέπων
Κεραμόπουλος Ευκλείδης
Αναπληρωτής Καθηγητής

Ημερομηνία 13-01-2021

Τίτλος Δ.Ε. Οδηγός για τους φοιτητές του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων (ΔΙΠΑΕ) (εφαρμογή android)
Κωδικός Δ.Ε. 20160
Ονοματεπώνυμο φοιτητή Ραφαήλ Τριανταφυλλίδης
Ονοματεπώνυμο εισηγητή Ευκλείδης Κεραμόπουλος
Ημερομηνία ανάληψης Δ.Ε. 14-04-2020
Ημερομηνία περάτωσης Δ.Ε. 13-01-2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Για τους στενούς μου ανθρώπους που ήταν δίπλα σε όλες τις δύσκολες στιγμές»

ΠΡΟΛΟΓΟΣ

Η ζωή ενός νέου φοιτητή είναι δύσκολη. Αντιμετωπίζει πάρα πολλές δυσκολίες που σχετίζονται με τεχνικά θέματα του τμήματος και της σχολής. Ο λόγος που επέλεξα να κάνω αυτή την εργασία, είναι για να μπορέσω να ενημερώσω και να διευκολύνω τους φοιτητές του τμήματος με επίκαιρες και εύκολα προσβάσιμες πληροφορίες, ώστε να ελαχιστοποιηθούν τα προβλήματα. Τα οφέλη αυτής της εργασίας και της εφαρμογής που παράχθηκε είναι πολλά. Κυρίως, είναι μια εφαρμογή στην οποία είναι μαζεμένες πολλές πληροφορίες σχετικά με το τμήμα, οπότε δε χρειάζεται η αναζήτηση πληροφοριών σε διάφορες πηγές. Επίσης, παρέχει στον φοιτητή κατανοητές οδηγίες για την επίλυση όλων αυτών των προβλημάτων.

ΠΕΡΙΛΗΨΗ

Το θέμα αυτής της εργασίας είναι η δημιουργία ενός εργαλείου για την καθοδήγηση των νέων φοιτητών από την εισαγωγή τους στο τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων, μέχρι να πάρουν πτυχίο. Θεώρησα ότι το εργαλείο αυτό θα πρέπει να είναι εύκολα προσβάσιμο από όλους οπότε κατέληξα ότι θα πρέπει να γίνει σαν εφαρμογή για κινητά. Στόχος της συγκεκριμένης εργασίας είναι η δημιουργία μια εφαρμογής Android, παρόλα αυτά, η ανάπτυξη του λογισμικού έγινε με τέτοιο τρόπο, ώστε να υπάρχουν περιθώρια μετατροπής της σε οποιαδήποτε άλλη πλατφόρμα (όπως εφαρμογή iOS ή διαδικτυακή εφαρμογή). Η εφαρμογή καλύπτει πολλούς τομείς προβλημάτων του φοιτητή όπως η πρώτη του γνωριμία με τους χώρους που διεκπεραιώνονται οι υπηρεσίες του τμήματος, όπως οι αίθουσες μαθημάτων, οι χώροι που λειτουργεί η γραμματεία κλπ. Επίσης, αναφέρεται σε συγκεκριμένα κοινά και μη προβλήματα και βήματα επίλυσης τους.

GUIDE FOR NEW STUDENTS OF DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

TRIANTAFILLIDIS RAFAEL

ABSTRACT

The goal of this thesis is the development of a tool that will guide the new students of the Department of Information and Electronic Engineering, from their registration, till they graduate. I considered that this tool should be easily accessible from everyone so concluded that it should be a mobile application. For this writing specifically, the goal is the development of an Android application, although the methods that were followed during the development, were carefully planned, so it should be easy to convert it to another platform(like iOS or web application). The application covers a lot of different sectors that are relevant to students like their first meet with the locations of the faculty like the amphitheaters or the secretariat office. Also, it refers to several common problems and steps to their solution.

ΕΥΧΑΡΙΣΤΙΕΣ

Υπάρχουν πολλοί παράγοντες που βοήθησαν στη διεκπαιρέωση αυτής της εργασίας. Πρώτα από όλα θα ήθελα να ευχαριστώ τον κ.Κεραμόπουλο ο οποίος είναι ο επιβλέπων αυτής της εργασίας και με βοήθησε σε όλα τα στάδια της από την αρχή μέχρι το τέλος. Επίσης, θα ήθελα να ευχαριστήσω τους την οικογένεια μου και τους στενούς μου φίλους για την ηθική υποστήριξη καθ όλη τη διάρκεια της εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Το θέμα της πτυχιακής εργασίας είναι η υλοποίηση ενός οδηγού για τους φοιτητές του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙΠΑΕ Θεσσαλονίκης. Η αρχική σχεδίαση περιελάμβανε μια εφαρμογή Android. Η εφαρμογή όμως δημιουργήθηκε με τέτοιο τρόπο ώστε να μπορεί να μεταφερθεί σε οποιαδήποτε πλατφόρμα με ευκολία. Επίσης, χρησιμοποιήθηκαν τεχνολογίες που προορίζονται για μοντέρνες εφαρμογές που έχουν στόχο την άψογη λειτουργικότητα και αντοχή στον χρόνο μετά από πιθανές ενημερώσεις ή αναβαθμίσεις.

Σαν πρώτο στοιχείο αυτής της εργασίας αναλύεται η εφαρμογή Android και το τι δυνατότητες προσφέρει στον χρήστη.Εδώ δεν αναφέρονται τεχνικά συστατικά που χρησιμοποιήθηκαν παρά μόνο το κέρδος που προσφέρει στους φοιτητές αυτή η εφαρμογή. Το περιεχόμενο της εφαρμογής είναι αποθηκευμένο σε μια διαδικτυακή βάση δεδομένων ώστε να γίνονται άμεσα οι αλλαγές στο περιεχόμενο εφόσον απαιτείται. Έτσι ο χρήστης δε θα χρειάζεται να ενημερώνει την εφαρμογή κάθε φορά που υπάρχει κάποια αλλαγή. Σαν διαδικτυακή βάση δεδομένων επιλέχθηκε το Google Cloud Firestore διότι παρέχει άριστη συνεργασία με τις εφαρμογές Android και όχι μόνο. Η βάση είναι τύπου noSQL που σημαίνει ότι οι διάφορες συλλογές που αποθηκεύει δε σχετίζονται μεταξύ τους με κάποιο δομημένο τρόπο. Επίσης, τα έγγραφα που αποθηκεύονται δεν ακολουθούν πάντα μια οργανωμένη δομή και μπορεί να υπάρχουν ασυμμετρίες μεταξύ τους. Αυτά τα στοιχεία βοήθησαν διότι τα δεδομένα που αποθηκεύτηκαν είχαν διαφορετική δομή μεταξύ τους.

Στη συνέχεια αναλύεται το Android σαν πλατφόρμα και τεχνολογίες χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Αναφέρονται με λεπτομέρεια τα σημαντικότερα στοιχεία που θα πρέπει να προσέξει ο προγραμματιστής ώστε να αναπτύξει με σωστό τρόπο μια εφαρμογή Android. Υπάρχει έμφαση στα στοιχεία από τα οποία αποτελείται το Android, από τι αποτελείται το καθένα και πώς συσχετίζονται μεταξύ τους για τη δημιουργία θαυμαστών εφαρμογών.

Στο επόμενο κεφάλαιο αναλύεται ο Web Client της εφαρμογής. Όπως αναφέρθηκε και παραπάνω η εφαρμογή αναπτύχθηκε με τέτοιο τρόπο ώστε να μπορεί να υλοποιηθεί σε οποιαδήποτε πλατφόρμα με ευκολία. Για τον Web Client χρησιμοποιήθηκε το front-end framework Angular διότι προσφέρει εργαλεία για την οργάνωση του κώδικα και τη μελλοντική επεκτασιμότητα του. Όπως και στο Android χρησιμοποιήθηκαν τεχνολογίες που είναι σχεδιασμένες να αντέχουν στον χρόνο. Το περιεχόμενο της εφαρμογής Web είναι πανομοιότυπο με το περιεχόμενο της εφαρμογής Android αφού προέρχεται από την ίδια βάση δεδομένων.

Το πέμπτο κεφάλαιο έχει στόχο να αναδείξει την τεχνική MVP ως τρόπο ανάπτυξης εφαρμογών που απομονώνουν τα διάφορα συστατικά στοιχεία των εφαρμογών μεταξύ τους. Συγκεκριμένα η τεχνική MVP στοχεύει στην απομόνωση του περιεχομένου, της λογικής και της δομής μεταξύ τους. Με αυτόν τον τρόπο η εφαρμογή μπορεί να συντηρηθεί και να αναβαθμιστεί με μεγαλύτερη ευκολία και αλλάζοντας μόνο τα πλέον απαραίτητα στοιχεία που απαιτούνται κάθε φορά. Και για τις 2 εφαρμογές, Android και Web, χρησιμοποιήθηκε αυτή η τεχνική.

Στη συνέχεια υπάρχει το κεφάλαιο που αναφέρεται στις βιβλιοθήκες που χρησιμοποιήθηκαν στην Android εφαρμογή. Οι καινούργιες εφαρμογές χρησιμοποιούν βιβλιοθήκες για βελτιστοποίηση των επιδόσεων τους και την αποφυγή λαθών. Χωρίς τις βιβλιοθήκες θα ήταν αναγκαστική η υλοποίηση υπηρεσιών που δε σχετίζονται με την τρέχων εφαρμογή. Τέτοιες υπηρεσίες όπως για παράδειγμα η

δημιουργία ενός HTTP request είναι κοινές στις περισσότερες εφαρμογές και έχουν υλοποιηθεί ήδη ώστε να έχουν βέλτιστη απόδοση. Για την ανάπτυξη της Android εφαρμογής χρησιμοποιήθηκαν βιβλιοθήκες που είχαν σαν στόχους τη δημιουργία ενός ενιαίου μενού για τη διεπαφή το οποίο ανταποκρίνεται με εμφανή τρόπο, τη σύνδεση με την απομακρυσμένη βάση δεδομένων firestore, την επικοινωνία με διάφορες υπηρεσίες της Google κ.α.

Όπως στο Android έτσι και στο Web χρησιμοποιήθηκαν βιβλιοθήκες για τους λόγους που προαναφέρθηκαν. Οι βιβλιοθήκες που χρησιμοποιήθηκαν στην Angular εφαρμογή είχαν όμως μια επιπλέον δυσκολία. Αυτή η δυσκολία οφείλεται στο ότι υπάρχουν πάρα πολλές διαφορετικές τεχνολογίες για τη δημιουργία Web εφαρμογών. Κάποιες βιβλιοθήκες δημιουργήθηκαν για να υποστηρίζουν κάποια συγκεκριμένα frameworks και δεν ταιριάζουν με ευκολία σε όλα. Αυτός ο περιορισμός δεν υπήρχε στο Android.

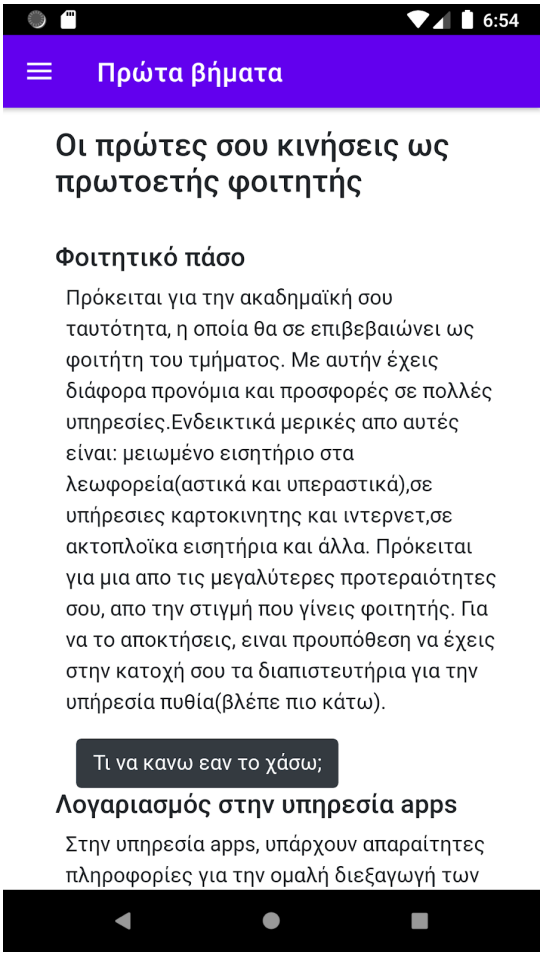
ΚΕΦΑΛΑΙΟ 2. ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

ΕΙΣΑΓΩΓΗ

Η πρώτη κύρια ενότητα αυτής της εργασίας θα αφιερωθεί στην περιληπτική παρουσίαση των διαφόρων λειτουργιών της εφαρμογής.

2.1 ΠΡΩΤΑ ΒΗΜΑΤΑ ΤΟΥ ΦΟΙΤΗΤΗ

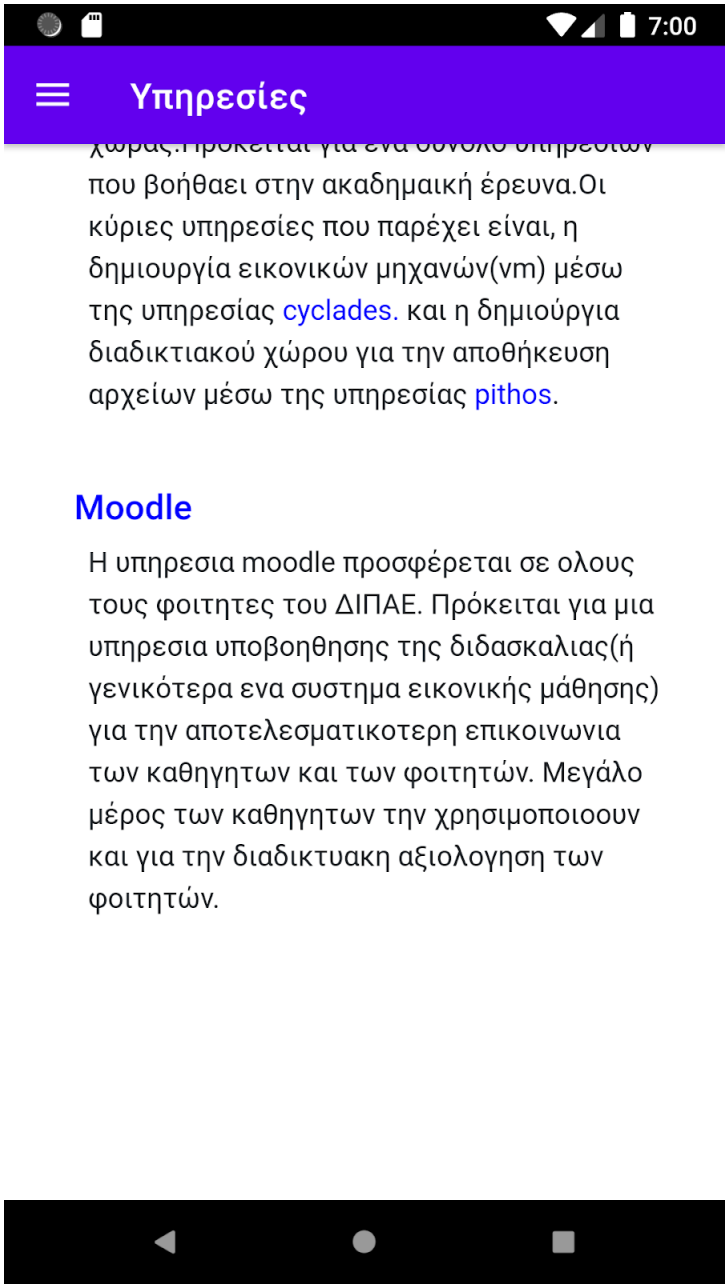
Η πρώτη κύρια οθόνη της εφαρμογής (εκτός της αρχικής οθόνης) είναι η οθόνη που δείχνει τα πρώτα βήματα που πρέπει να ακολουθήσει κάποιος καινούριος φοιτητής του τμήματος. Περιέχει πληροφορίες για την απόκτηση του φοιτητικού πάσου (όπως και τη διαδικασία που απαιτείται σε περίπτωση απώλειας), τις διαδικασίες για την απόκτηση λογαριασμού στις υπηρεσίες apps και πυθία (Σχήμα 2.1). Ακόμα περιέχει πληροφορίες σχετικά με την εγγραφή στο σύστημα eudoxus.



Σχήμα 2.1: Διεπαφή πρώτα βήματα του φοιτητή.

2.2 ΥΠΗΡΕΣΙΕΣ ΤΟΥ ΤΜΗΜΑΤΟΣ

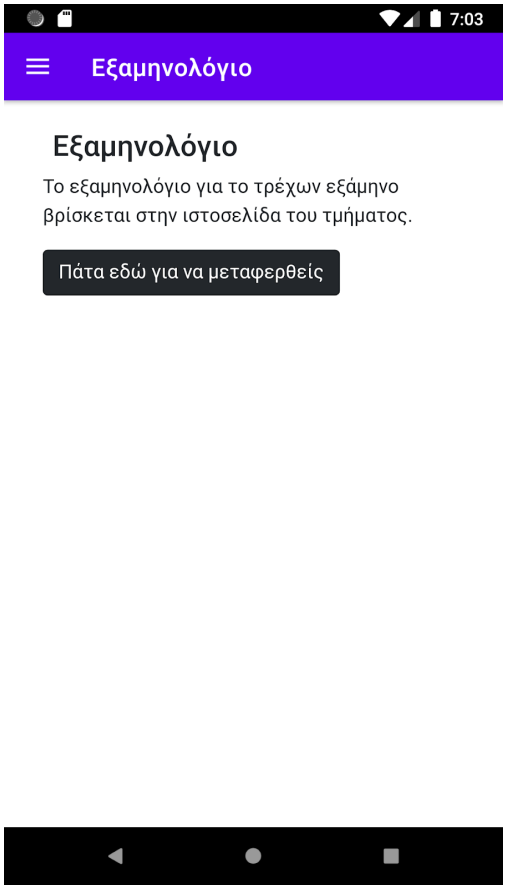
Η λειτουργία *υπηρεσίες* είναι η δεύτερη κύρια λειτουργία της εφαρμογής όπως φαίνεται στο σχήμα 2.2. Σε αυτήν την καρτέλα παρουσιάζονται οι διάφορες υπηρεσίες του τμήματος και τι χρησιμότητα έχουν προς τον φοιτητή. Μεταξύ άλλων παρουσιάζονται οι υπηρεσίες apps, πυθία και moodle.



Σχήμα 2.2: Διεπαφή υπηρεσίες του τμήματος

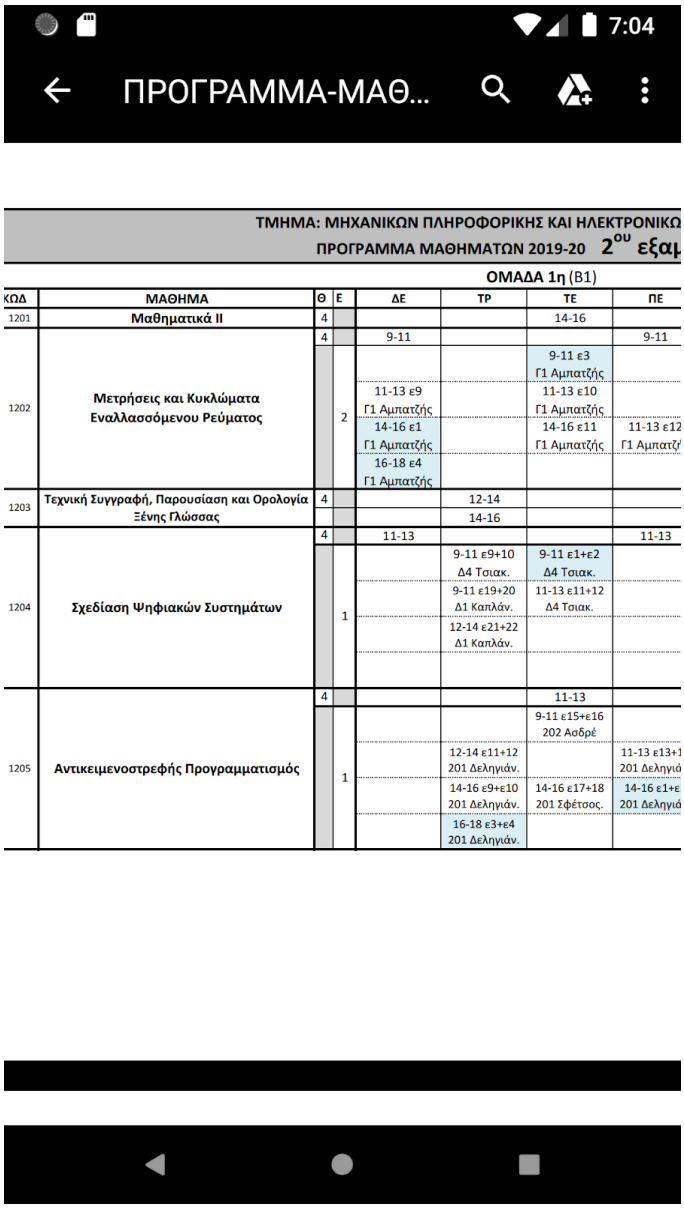
2.3 ΕΞΑΜΗΝΟΛΟΓΙΟ

Σε αυτή τη λειτουργία της εφαρμογής (Σχήμα 2.3) υπάρχει υπερσύνδεσμος για το τρέχον εξαμηνολόγιο.



Σχήμα 2.3: Διεπαφή εξαμηνολογιο

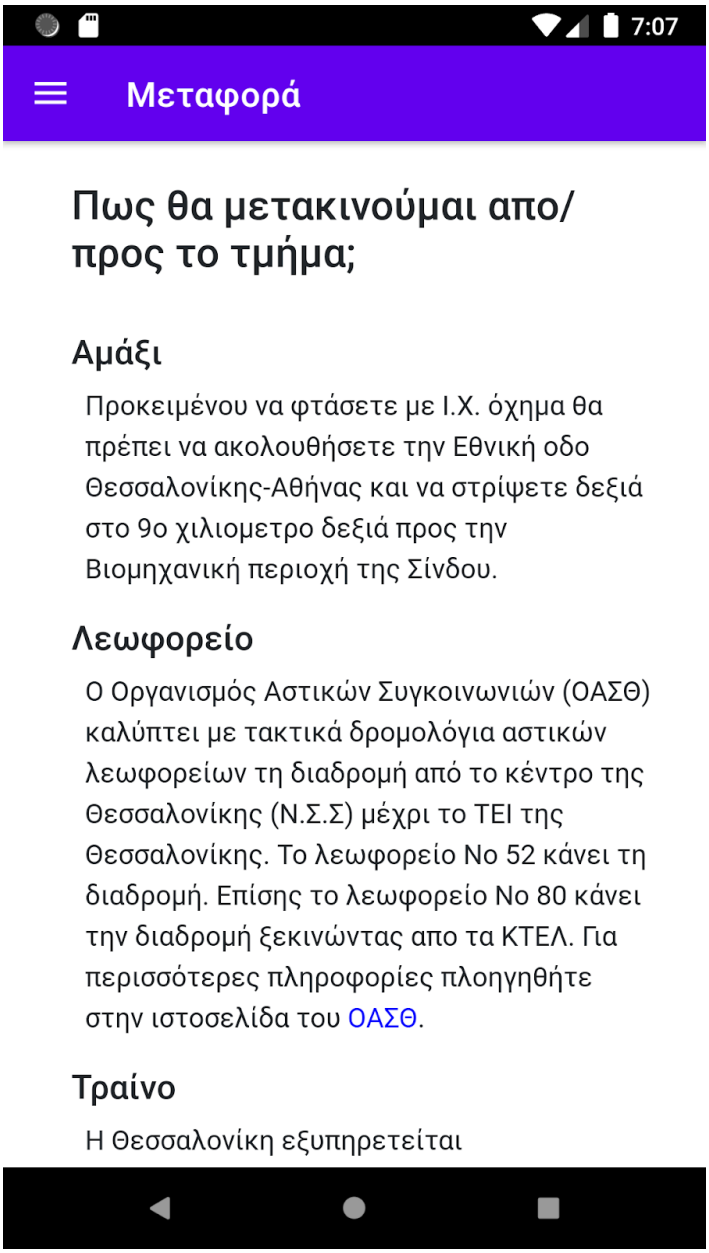
Μόλις πατήσει στον υπερσύνδεσμο ο χρήστης, θα μεταφερθεί στην τοποθεσία να δει το εξαμηνολογιο. Εκεί υπάρχει ένα αρχείο PDF (Σχήμα 2.4) που περιέχει το τρέχον εξαμηνολόγιο κάθε εξαμήνου. Λόγω των υπαρχόντων συστατικών και υπηρεσιών που υπάρχει αυτή τη στιγμή στο τμήμα η διεύθυνση URL για το εξαμηνολόγιο θα πρέπει να αλλάζει από τον διαχειριστή της εφαρμογής κάθε φορά που γίνεται κάποια αλλαγή(π.χ. κάθε εξάμηνο). Αυτό γίνεται μέσω της διαδικτυακής βάσης δεδομένων firestore δυναμικά ώστε να μη χρειάζεται καμία ενέργεια από τους φοιτητές.



Σχήμα 2.4: Εξαμηνολόγιο

2.4 ΜΕΤΑΦΟΡΑ

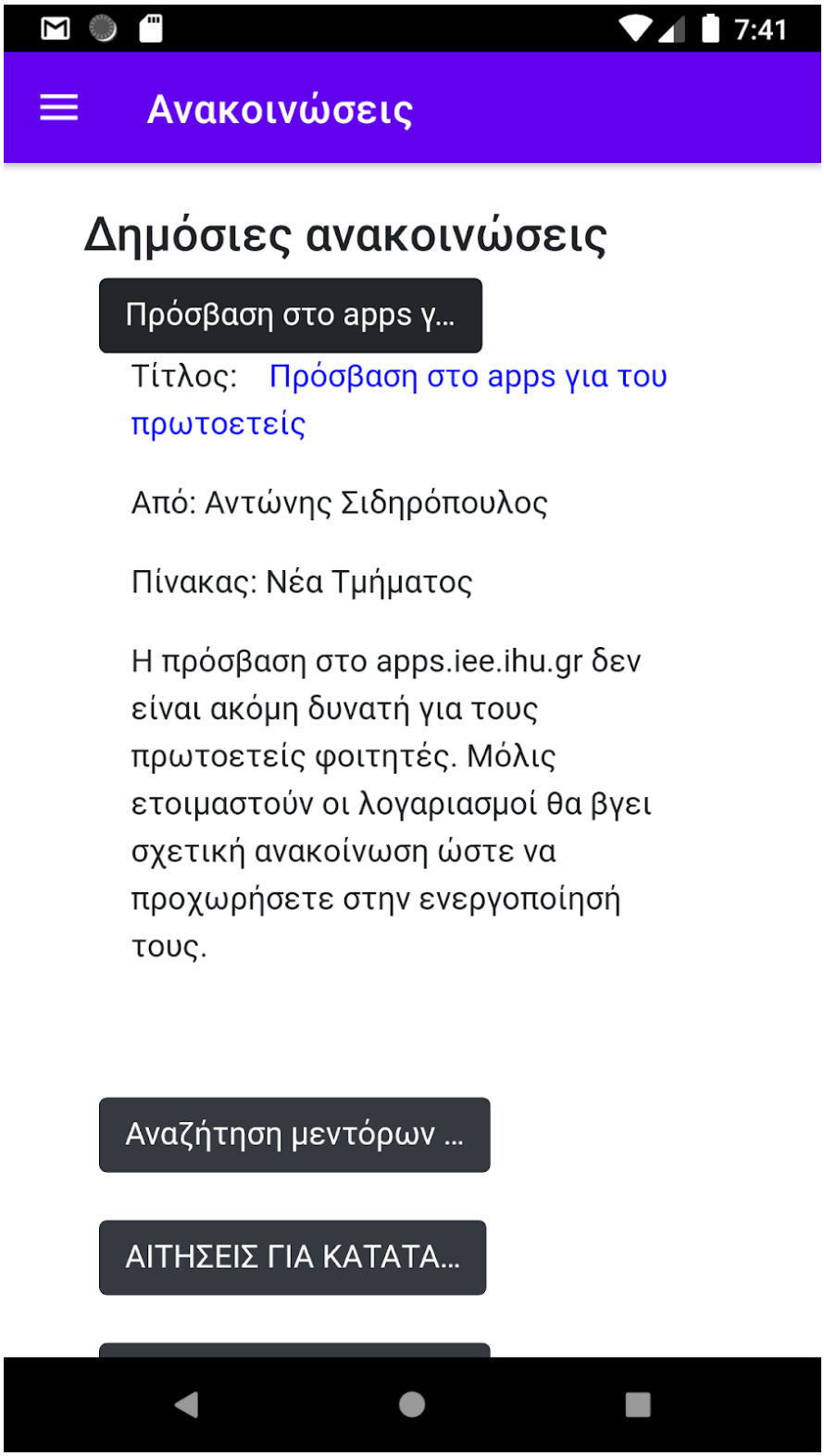
Η τέταρτη κύρια λειτουργία της εφαρμογής (Σχήμα 2.5) αναφέρεται στους τρόπους μετακίνησης προς το τμήμα και γενικότερα στο ΔΙΠΑΕ. Οι τρόπο μετακίνησης περιλαμβάνουν το αμάξι, λεωφορείο και τραίνο.



Σχήμα 2.5: Διεπαφή μεταφορά από/προς το τμήμα

2.5 ΑΝΑΚΟΙΝΩΣΕΙΣ

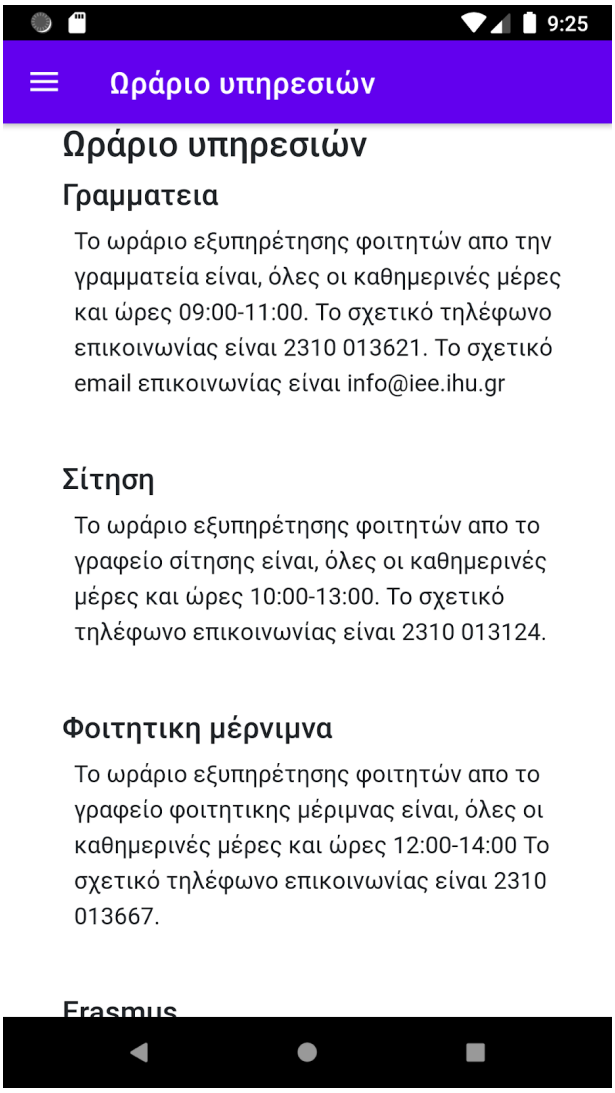
Σε αυτή την καρτέλα (Σχήμα 2.6) παρουσιάζονται όλες οι δημόσιες ανακοινώσεις του τμήματος.



Σχήμα 2.6: Διεπαφή ανακοινώσεις του τμήματος

2.6 ΩΡΑΡΙΟ ΥΠΗΡΕΣΙΩΝ

Σε αυτή την καρτέλα (Σχήμα 2.7) υπάρχουν τα ωράρια λειτουργίας για τις διάφορες υπηρεσίες του τμήματος.

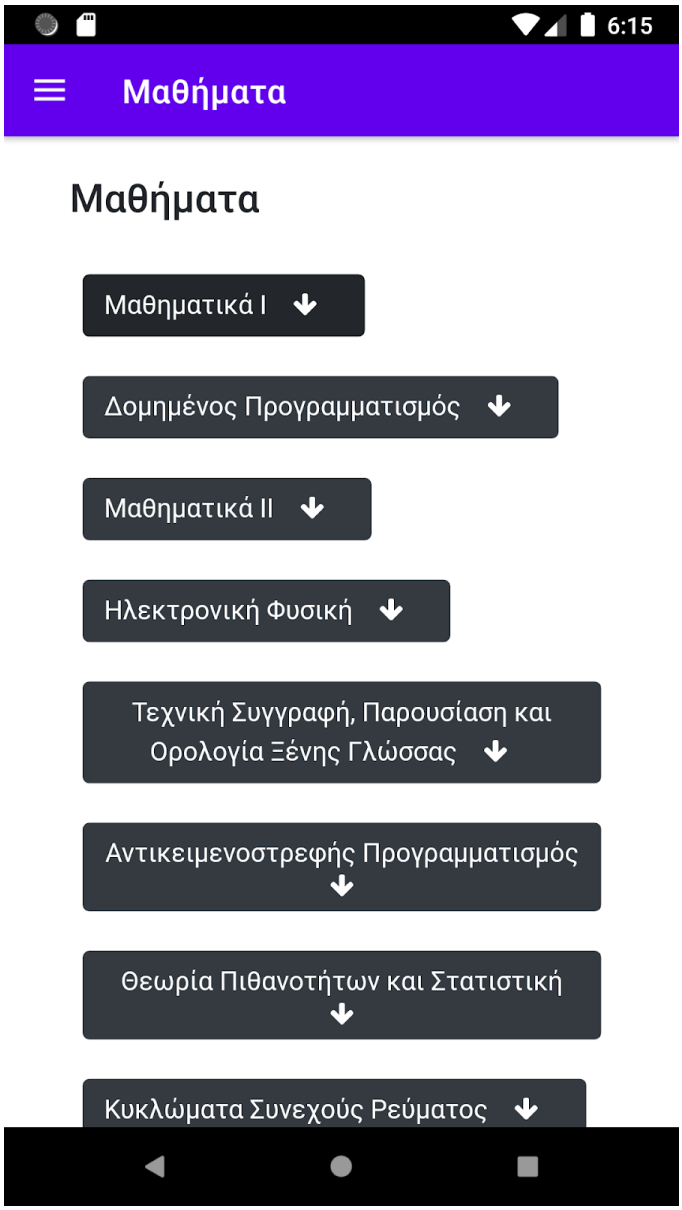


Σχήμα 2.7: Διεπαφή Ωράριο υπηρεσιών

Σε περίπτωση αλλαγής αυτών των στοιχείων επικοινωνίας γίνεται αλλαγή περιεχομένου από τον διαχειριστή της εφαρμογής στη διαδικτυακή βάση δεδομένων firestore ώστε οι χρήστες να παίρνουν άμεσα τα καινούρια στοιχεία.

2.7 ΜΑΘΗΜΑΤΑ

Σε αυτή την καρτέλα (Σχήμα 2.8) παρουσιάζονται τα μαθήματα του τμήματος καθώς και οι προϋποθέσεις πτυχιακής και πρακτικής εργασίας.

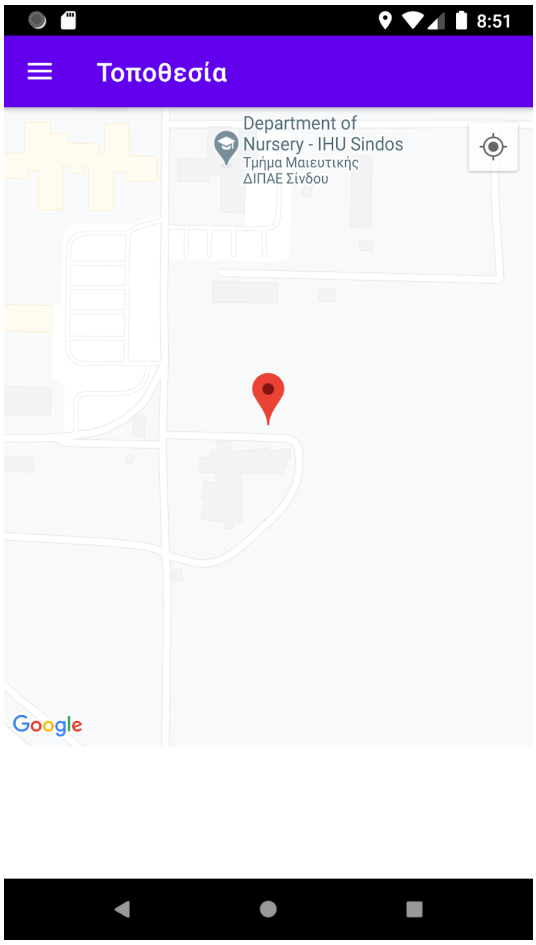


Σχήμα 2.8: Διεπαφή Μαθήματα τμήματος

Τα μαθήματα της σχολής είναι αποθηκευμένα στη βάση δεδομένων firestore σε ξεχωριστό collection. Κάθε μάθημα αποτελείται από τον τίτλο του, την περιγραφή του, τον κωδικό του και το εξάμηνο κατά το οποίο διδάσκεται. Όλες αυτές οι πληροφορίες εμφανίζονται στη διεπαφή ώστε να γίνει κατανοητό το περιεχόμενο του μαθήματος.

2.8 ΤΟΠΟΘΕΣΙΑ

Η καρτέλα τοποθεσία περιέχει χάρτη που δείχνει την τοποθεσία του τμήματος μέσα στην Αλεξάνδρεια Πανεπιστημιούπολη του ΔΙΠΑΕ στη Σίνδο όπως φαίνεται στο Σχήμα 2.9.

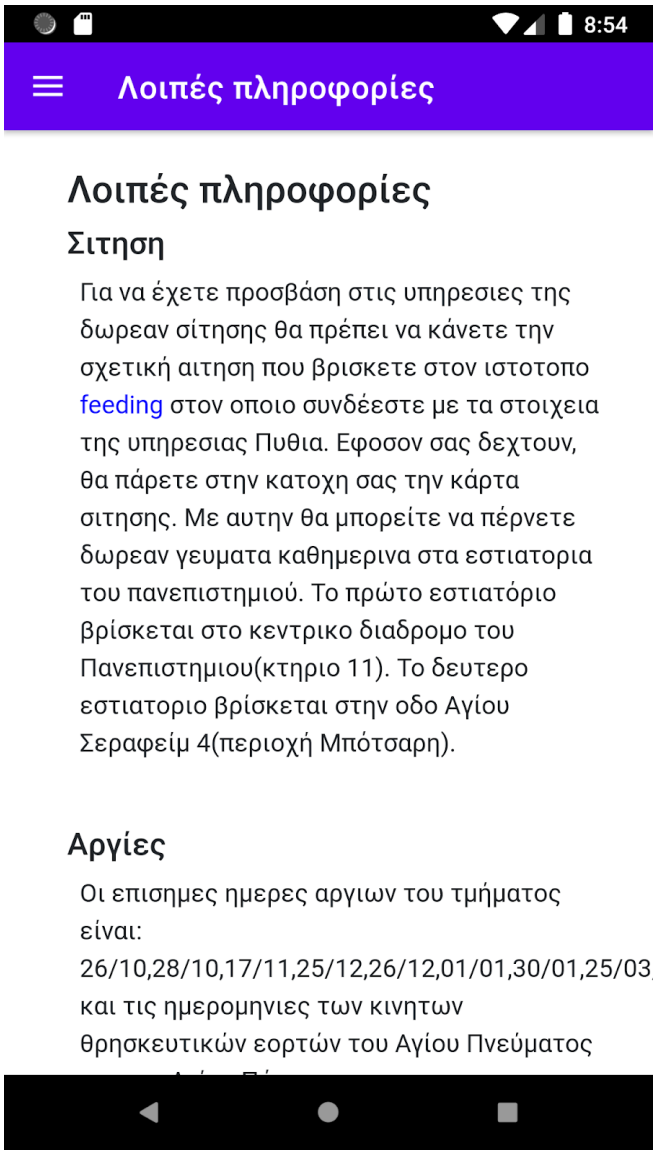


Σχήμα 2.9: Διεπαφή τοποθεσία τμήματος

Για την εμφάνιση της τοποθεσίας του τμήματος χρησιμοποιήθηκαν χάρτες της Google. Λόγω αυτού του στοιχείου είναι δυνατόν για εμφάνιση περισσότερων σχετικών στοιχείων χρησιμοποιώντας την εφαρμογή Google Maps.

2.9 ΛΟΙΠΕΣ ΠΛΗΡΟΦΟΡΙΕΣ

Η καρτέλα λοιπές πληροφορίες (Σχήμα 2.10) περιέχει πληροφορίες σχετικά με γενικά θέματα του τμήματος. Ενδεικτικά περιέχονται πληροφορίες σχετικά με τη δωρεάν σίτιση και το πρόγραμμα erasmus+



Σχήμα 2.10: Διεπαφή λοιπές πληροφορίες

ΕΠΙΛΟΓΟΣ

Σε αυτή την ενότητα παρουσιάστηκαν οι λειτουργίες της εφαρμογής android. Εν συντομία πρόκειται σύμπτυξη πληροφοριών από πολλές πηγές ώστε να δημιουργηθεί μια οργανωμένη δομή που περιέχει περιεκτικά τις σημαντικότερες πληροφορίες.

ΚΕΦΑΛΑΙΟ 3. ANDROID CLIENT

ΕΙΣΑΓΩΓΗ

Σε αυτή την ενότητα θα γίνει μια βασική εισαγωγή στις κύριες τεχνολογίες που σχετίζονται με το Android και χρησιμοποιήθηκαν για την εκπόνηση αυτής της εργασίας.

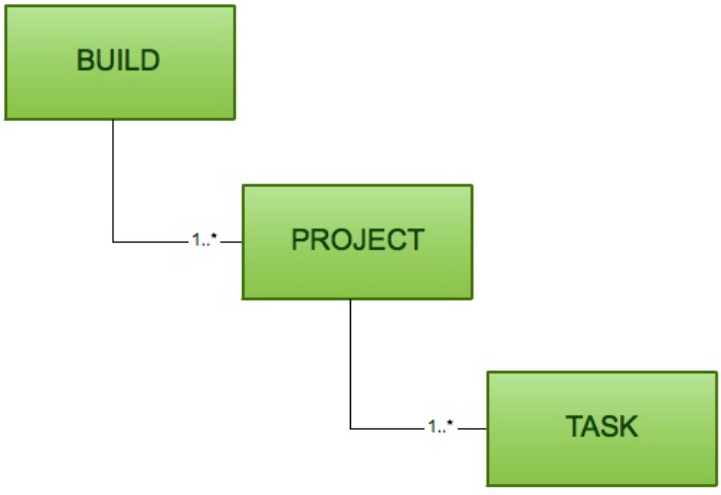
3.1 GRADLE

Το Gradle, είναι ένα εργαλείο ανοιχτού λογισμικού για την κατασκευή προγραμμάτων που στοχεύει στην απόδοση και στην ευελιξία. Η κύρια αιτία πίσω από τη δημιουργία του, ήταν η δημιουργία ενός εργαλείου που δε χρησιμοποιεί τη γλώσσα XML αλλά μια άλλη πιο ευέλικτη και εύχρηστη. Για τον σκοπό αυτό δημιουργήθηκε μια γλώσσα που μοιάζει με τη γλώσσα προγραμματισμού Groovy.

Το Gradle περιέχει δύο βασικές έννοιες, το project και το task (Σχήμα 3.1).

Το project είναι κάτι που κατασκευάζουμε. Πρόκειται δηλαδή για τη μεταγλώττιση του πηγαίου κώδικα σε κάτι που μπορεί να χρησιμοποιηθεί από τον τελικό χρήστη της εφαρμογής (π.χ. ένα .jar ή .apk αρχείο). Αυτή η διαδικασία γίνεται ακολουθώντας κάποια βήματα, τα οποία ονομάζονται tasks.

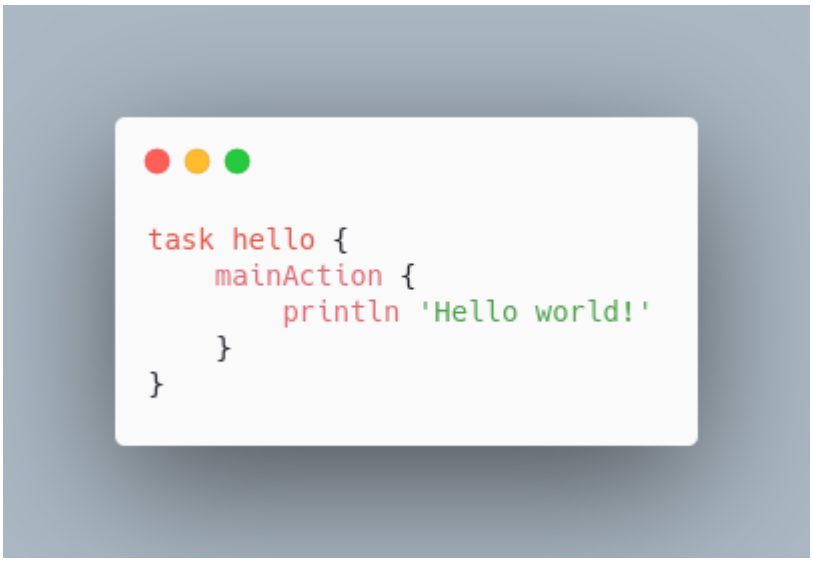
Το task είναι ένα βήμα που περιγράφει μια εξειδικευμένη ενέργεια (π.χ. η περάτωση λειτουργικών test της εφαρμογής).
Χρησιμοποιώντας ένα ή περισσότερα project μπορούμε να δημιουργήσουμε ένα gradle build.



Σχήμα 3.1: Δομή εργασιών gradle

Σχετικά αρχεία με κάθε gradle build είναι τα:

- Αρχείο build.gradle
  Το αρχείο build.gradle δηλώνει ένα project και τα task του (Σχήμα 3.2).



Σχήμα 3.2: gradle task

- Αρχείο gradle.properties
  Το αρχείο gradle.properties (Σχήμα 3.3) δηλώνει τις διαφορές ιδιότητες που σχετίζονται με την κατασκευή του project.

Σχήμα 3.3: Αρχείο gradle.properties

- Αρχείο settings.gradle (Σχήμα 3.4)
  Πρόκειται για ένα προεραιτικό αρχείο. Αν υπάρχουν παραπάνω από ένα gradle projects, είναι υποχρεωτικό. Περιγράφει ποια projects είναι αναγκαία για την κατασκευή.

Σχήμα 3.4: Αρχείο settings.gradle

Για να ξεκινήσουμε ένα task ενός project αρκεί να καλέσουμε την εντολή gradle [task].

3.2 ANDROID MANIFEST

Το AndroidManifest.xml είναι ένα αρχείο που βρίσκεται απαραίτητα στον ριζικό κατάλογο κάθε Android project (Σχήμα 3.5). Αυτό το αρχείο, περιέχει τις απαραίτητες πληροφορίες για την εφαρμογή σχετικά με τις παραμέτρους κατασκευής του και το λειτουργικό σύστημα Android.

Οι κύριες πληροφορίες που δηλώνονται στο AndroidManifest.xml είναι:

- Το όνομα της τελικής εφαρμογής (πως θα εμφανίζεται στον χρήστη)
- Όλα τα συστατικά στοιχεία της εφαρμογής, όπως activities και fragments
- Την ελάχιστη και τη μέγιστη έκδοση του Android API τις οποίες θα χρησιμοποιήσει η εφαρμογή.
- Τις διάφορες άδειες ασφαλείας που απαιτεί η εφαρμογή για τις διάφορες λειτουργίες της.
- Τις διάφορες απαιτήσεις λογισμικού και υλικού που απαιτεί η εφαρμογή για τις διάφορες λειτουργίες της

Σχήμα 3.5: Αρχείο AndroidManifest.xml

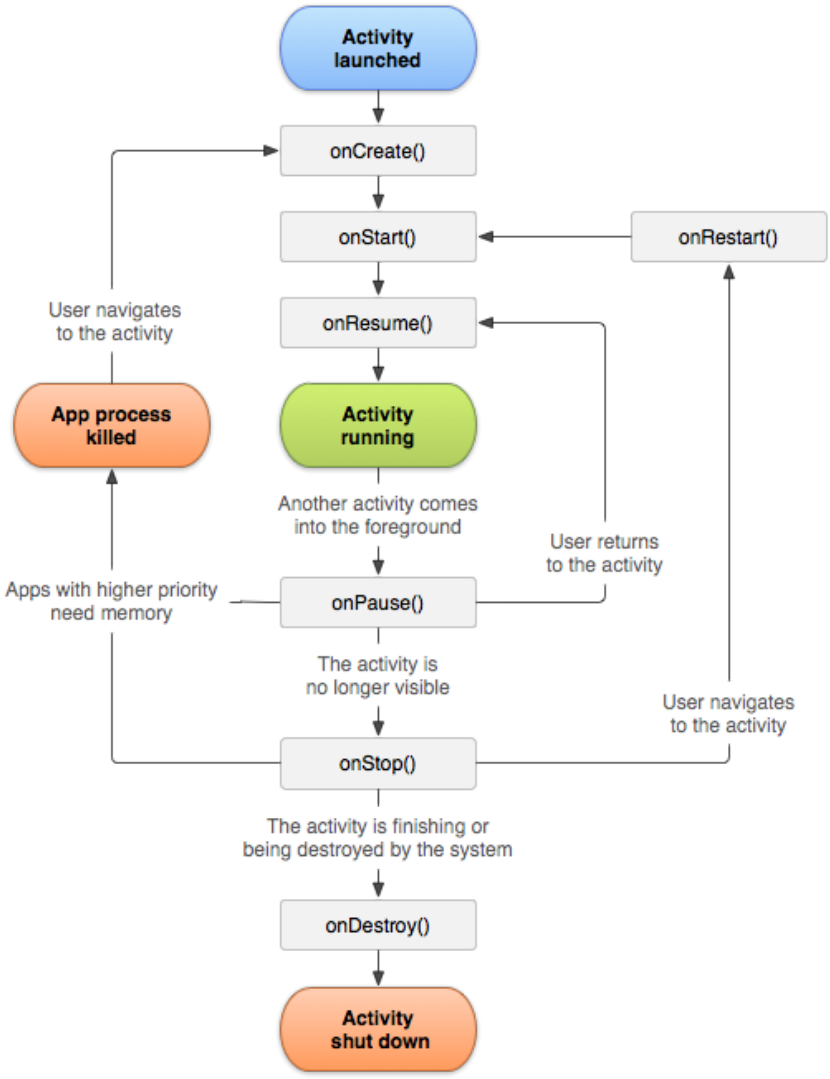
3.3 ACTIVITIES

Ένα android activity αναπαριστά μια διεπαφή που στοχεύει σε μια συγκεκριμένη ενέργεια χρήστη. Μια android εφαρμογή περιέχει τουλάχιστον ένα activity. Μόλις ξεκινάει η εφαρμογή, φορτώνει το βασικό activity.

Συνήθως τα activities ξεκινάνε καλύπτοντας ολόκληρη την οθόνη, αλλά αυτό δε γίνεται πάντα. Ένα activity θα μπορούσε να εμφανιστεί σαν ‘αιωρούμενο’ πάνω σε κάποιο άλλο, σαν μέρος πολλαπλών activities σε ένα πολυ-παραθυρικό περιβάλλον και άλλα.

Το activity έχει ένα κύκλο ζωής (Σχήμα 3.6). Ο προγραμματιστής μπορεί να το ελέγξει κατά τη διάρκεια της ζωής του καλώντας τις διάφορες μεθόδους του κύκλου ζωής του. Πιο αναλυτικά:

- onCreate
  Είναι η πρώτη μέθοδος που καλείται, μόλις ξεκινήσει το activity. Εδώ ο προγραμματιστής θα πρέπει να υλοποιήσει τη βασική αρχική λογική που σχετίζεται με το activity.
- onStart
  Αυτή η μέθοδος καλείται μόλις το activity γίνεται ορατό στον χρήστη για πρώτη φορά.
- onResume
  Αυτή η μέθοδος καλείται μόλις το activity είναι έτοιμο για αλληλεπίδραση με τον χρήστη. Είναι το σημείο στο οποίο η εφαρμογή ζει περισσότερο. Το activity μένει σε αυτή την κατάσταση μέχρι να συμβεί κάτι που να προκαλέσει την εφαρμογή να χάσει το focus(π.χ. σε περίπτωση εισερχόμενης κλήσης).
- onPause
  Κατά τη διάρκεια που ένα activity βρίσκεται σε PAUSED κατάσταση, δεν υπάρχει καθόλου αλληλεπίδραση με τον χρήστη.
- onStop
  Αυτή η μέθοδος καλείται μόλις το activity αλλάζει κατάσταση από ορατό σε μη-ορατό.
- onDestroy
  Αυτή η μέθοδος καλείται πριν το σύστημα σταματήσει το activity.



Σχήμα 3.6: Activity lifecycle

3.3.1 Αλληλεπίδραση με άλλα activities

Κατά τη διάρκεια ενός activity, είναι σύνηθες να θέλουμε να ξεκινήσουμε ένα νέο activity, με το οποίο ο χρήστης θα αλληλεπιδράσει και με βάση τις ενέργειες του να περιμένουμε ένα αποτέλεσμα στο αρχικό activity (Σχήμα 3.10). Για τον σκοπό αυτό υπάρχει η μέθοδος *startActivityForResult* (Σχήμα 3.7). Η μέθοδος αυτή απαιτεί δύο παραμέτρους, ένα αντικείμενο τύπου Intent και ένα int. Το Intent περιέχει τις πληροφορίες για το activity που πρόκειται να ανοίξει, και ένα int που προσδιορίζει αυτό το activity.

```
//Class UsersListActivity
final String userId = "868a7c9b-bc42-400b-91e0-d43fa6238e6a";
final int SHOW_USER_DETAILS_REQUEST = 101;
final Intent intent = new Intent(this,UserDetailsActivity.class);
final Bundle userDetailsBundle = new Bundle();
userDetailsBundle.putString("USER_ID",userId);
startActivityForResult(
         intent,
         SHOW_USER_DETAILS_REQUEST);
```

Σχήμα 3.7: Επικοινωνία των activities(1/3)

Στο παραπάνω παράδειγμα, δημιουργούμε ένα intent για την κλάση UserDetailsActivity. Έπειτα εμπλουτίζουμε το intent με ένα bundle, το οποίο περιέχει το ID του χρήστη για τον οποίο θέλουμε να εμφανίσουμε τα στοιχεία. Επίσης, δημιουργούμε μια σταθερά, που περιέχει το αναγνωριστικό της παραπάνω ενέργειας. Θα χρησιμοποιήσουμε αυτό το αναγνωριστικό στη συνέχεια για να παραλάβουμε το αποτέλεσμα από το καινούργιο activity.

Στο ίδιο activity θα πρέπει να κάνουμε override τη μέθοδο *onActivityResult* (Σχήμα 3.8). Αυτή η μέθοδος καλείται μόλις τελειώσει το activity που ανοίξαμε.

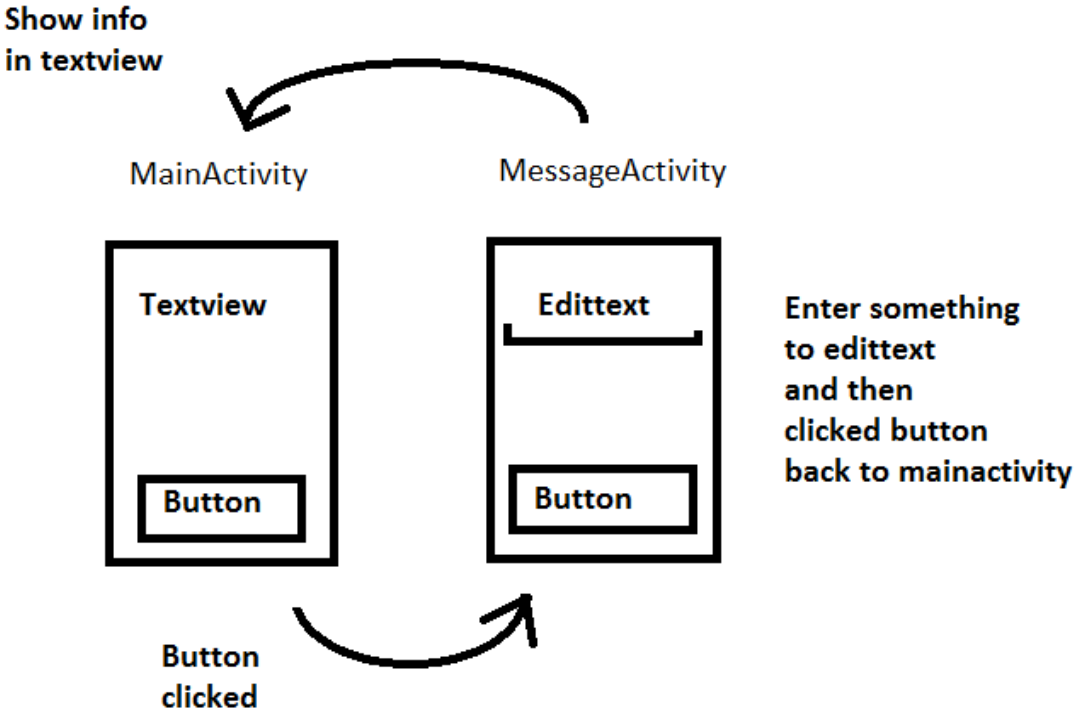
```
//Class UsersListActivity
final int SHOW_USER_DETAILS_REQUEST = 101;
protected void onActivityResult(int requestCode, int resultCode,
             Intent data) {
         if (requestCode == SHOW_USER_DETAILS_REQUEST) {
             if (resultCode == RESULT_OK) {
                 System.out.println("Operation was successful");
             }
         }
     }
```

Σχήμα 3.8: Επικοινωνία των activities(2/3)

Στο δεύτερο activity όταν θέλουμε να επιστρέψουμε μια τιμή, πρέπει να καλέσουμε τη μέθοδο *setResult* και έπειτα τη μέθοδο *finish* (Σχήμα 3.9).

```
//Class UsersDetailsActivity
Intent returnIntent = new Intent();
setResult(RESULT_OK, returnIntent);
finish();
```

Σχήμα 3.9: Επικοινωνία των activities(3/3)



Σχήμα 3.10: Αλληλεπίδραση μεταξύ των activities

3.4 FRAGMENTS

Ένα android fragment, αναπαριστά μια συμπεριφορά ή μέρος μιας διεπαφής. Κάθε android activity, μπορεί να περιέχει ένα ή περισσότερα fragments. Επίσης, ένα fragment, μπορεί να είναι γενικού σκοπού, και να χρησιμοποιείτε σε πολλαπλά διαφορετικά activities. Το fragment, έχει το δικό του lifecycle και μπορεί να δεχτεί διάφορες ενέργειες χρήστη.

Το fragment, δεν είναι δυνατόν να υπάρχει μόνο του. Το κάθε fragment ανήκει σε τουλάχιστον ένα ‘γονέα’ activity, και η ζωή του επηρεάζεται άμεσα από τη συμπεριφορά του. Για παράδειγμα, αν ένα activity μπει σε κατάσταση PAUSED, τότε αναγκαστικά και όλα τα fragments που περιέχει αυτό το activity θα μπουν επίσης σε κατάσταση PAUSED. Παρόλα αυτά, αν ένα activity είναι σε κατάσταση RUNNING, ο χρήστης έχει τη δυνατότητα να αλλάξει την κατάσταση του activity σε όποια θέλει αυτός. Αν ενώ τρέχει το activity, προσθέσουμε ένα καινούργιο fragment ή αφαιρέσουμε ένα ήδη υπάρχων, υπάρχει η δυνατότητα να μπεί επίσης και στο backstack του activity ώστε να μπορεί να περιηγηθεί ο χρήστης μπρός και πίσω.

3.4.1 Η σημαντικότητα των Fragments

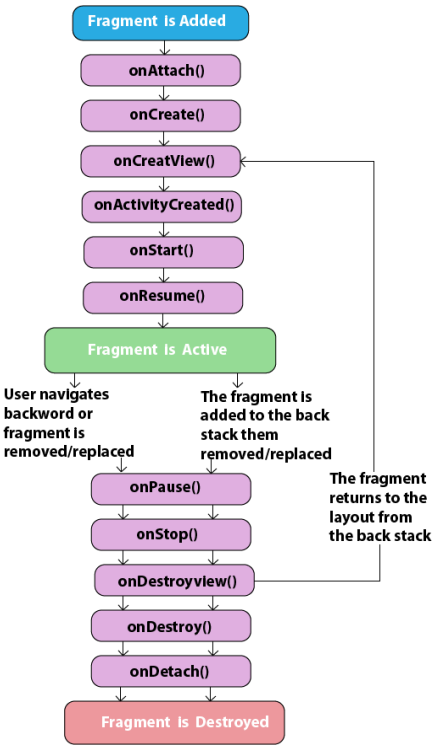
Οι κύριες και πιο κοινές χρήσεις των fragments περιλαμβάνουν:

- **Επαναχρησιμοποίηση συστατικών λογικής και εμφάνισης**
  Τα fragments μπορούν να επαναχρησιμοποιηθούν σε διάφορα σημεία οθονών ή activities.
  Για παράδειγμα, ένα fragment που εμφανίζει σε μορφή λίστας κάποια δεδομένα, θα μπορούσε να χρησιμοποιηθεί για να εμφανίσει τη λίστα των χρηστών, και τη λίστα των μαθημάτων.
- **Υποστήριξη για συσκευές tablet**
  Συνήθως, οι συσκευές tablet, έχουν σε μεγάλο βαθμό, διαφορετική μορφοποίηση σχετικά με τις κινητές συσκευές, κυρίως λόγω μεγέθους οθόνης. Χρησιμοποιώντας τα fragments, θα μπορούσαμε να εμφανίσουμε την ίδια πληροφορία, με διαφορετικό τρόπο, ανάλογα με τις ανάγκες και δυνατότητες κάθε συσκευής.
- **Προσανατολισμός οθόνης**
  Στις περισσότερες περιπτώσεις, όταν η οθόνη της συσκευής βρίσκεται σε προσανατολισμό πορτρέτου, έχει αρκετά μεγάλες διαφορές σε σχέση με όταν βρίσκεται σε οριζόντιο προσανατολισμό. Χρησιμοποιώντας τα fragments, θα μπορούσαμε να εμφανίσουμε την ίδια ή παρόμοια πληροφορία για οποιονδήποτε προσανατολισμό οθόνης.

3.4.2 Ο κύκλος ζωής των Fragments

Ομοίως, με το activity, το fragment έχει ένα κύκλο ζωής (Σχήμα 3.11). Ο προγραμματιστής μπορεί να το ελέγξει κατά τη διάρκεια της ζωής του καλώντας τις διάφορες μεθόδους του κύκλου ζωής του. Πιο αναλυτικά:

- onAttach
  Αυτή η μέθοδος καλείται μόλις ένα fragment ‘συνδεθεί’ με ένα activity
- onCreate
  Αυτή η μέθοδος καλείται μόλις δημιουργείται το fragment. Εδώ ο προγραμματιστής θα πρέπει να υλοποιήσει τυχόν βασική λογική που σχετίζεται με το fragment.
- onCreateView
  Αυτή η μέθοδος καλείται μόλις το fragment συσχετιστεί με ένα view.
- onActivityCreated
  Αυτή η μέθοδος καλείται μόλις το γονικό activity, ολοκληρώσει τη μέθοδο του *onCreate*.
- onStart
  Αυτή η μέθοδος καλείται μόλις το fragment είναι έτοιμο για να εμφανιστεί στην οθόνη.
- onResume
  Σε αυτήν τη μέθοδο ο προγραμματιστής μπορεί να δεσμεύσει διάφορους πόρους που απαιτούνται για τη λειτουργικότητα του fragment.
- onPause
  Σε αυτήν τη μέθοδο ο προγραμματιστής πρέπει να αποδεσμεύσει πόρους που δε χρειάζεται.
- onStop
  Αυτή η μέθοδος καλείται μόλις το fragment αλλάζει κατάσταση από ορατό σε μη-ορατό.
- onDestroyView
  Αυτή η μέθοδος καλείται όταν διαγράφεται η συσχέτιση του fragment με το view του.
- onDestroy
  Αυτή η μέθοδος καλείται πριν το σύστημα σταματήσει το fragment.
- onDetach
  Αυτή η μέθοδος καλείται μόλις το activity σταματήσει να σχετίζεται με το fragment.



Σχήμα 3.11: Fragment lifecycle

3.4.3 Επικοινωνία fragment με το γονικό activity

Για την επικοινωνία ενός fragment με το γονικό του activity, υπάρχει η μέθοδος *getActivity* (Σχήμα 3.11). Αυτή η μέθοδος επιστρέφει ένα αντικείμενο τύπου Activity. Αν θέλουμε να χρησιμοποιήσουμε μια συγκεκριμένη μέθοδο που δεν είναι κοινή σε όλα τα activities, αλλά είμαστε σίγουροι ότι υπάρχει στο γονικό activity, θα πρέπει να κάνουμε cast το αντικείμενο που πήραμε σαν αποτέλεσμα και έπειτα να καλέσουμε τη μέθοδο που επιθυμούμε.

Σχήμα 3.12: Επικοινωνία fragment με γονικό activity

3.5 RESOURCES(& ASSETS)

3.5.1 Resources

Τα android resources είναι πρόσθετα αρχεία, συνήθως στατικό περιεχόμενο που χρησιμοποιούνται από την εφαρμογή. Τα android resources μεταξύ άλλων περιλαμβάνουν:

- Εικόνες
- Ορισμοί layouts
- Κείμενα που χρησιμοποιούνται στις διεπαφές χρήστη
- Κινούμενες εικόνες (animations)

Όλα τα παραπάνω μπορούν (και πρέπει) να συντηρούνται να αναβαθμίζονται ανεξάρτητα από τον κώδικα της εφαρμογής. Επιπρόσθετα, θα πρέπει να παρέχονται εναλλακτικές λύσεις για κάθε τύπο resource, εφόσον απαιτείται για τις διάφορες ιδιαιτερότητες των συσκευών τις οποίες στοχεύει η εφαρμογή.

Κατά τη διάρκεια, στην οποία λειτουργεί η εφαρμογή, επιλέγονται τα κατάλληλα resources ανάλογα με τις διάφορες διαμορφώσεις της συσκευής, όπως η γλώσσα των κείμενων, το μέγεθος των εικόνων και άλλα.

Για να αναφερθούμε και να χρησιμοποιήσουμε τα διάφορα resources, της εφαρμογής στον κώδικα, μπορούμε να χρησιμοποιήσουμε την Android κλάση *R* (Σχήμα 3.14).

Τα διάφορα resources θα πρέπει να ομαδοποιούνται ανάλογα την υποκατηγορία στην οποία ανήκουν. Ο ριζικός κατάλογος για τα resources ονομάζεται res (Σχήμα 3.13).

Σχήμα 3.13: Android res directory structure

Παραπάνω παρουσιάζεται ένας τυπικός κατάλογος που περιέχει android resources. Συγκεκριμένα περιέχονται

- Μια εικόνα (drawable/graphic.png)
- Δύο ορισμοί layouts(layout/main.xml,layout/info.xml)
- Μια εικόνα που χρησιμοποιείται σαν εικονίδιο της εφαρμογής στον Android launcher
- Ένα αρχείο που περιέχει τα διάφορα κείμενα της εφαρμογής στην προκαθορισμένη γλώσσα.

Σχήμα 3.14: Ανάκτηση πληροφοριών από το res directory

Παραπάνω παρουσιάζεται ο τυπικός τρόπος για να αποκτήσουμε πρόσβαση σε ένα android resource.

3.5.2 Assets

Τα android assets μοιάζουν πολύ με τα resources, αλλά διαφέρουν σε κάποια κομβικά σημεία. Μοιάζουν στο ότι και τα δύο περιέχουν στατικό περιεχόμενο που χρησιμοποιείται στον κώδικα της εφαρμογής. Παρόλα αυτά, χρησιμοποιούνται για διαφορετικούς σκοπούς.

Τα android assets τυπικά περιλαμβάνουν αρχεία τα οποία θα πρέπει να χρησιμοποιηθούν σαν ακατέργαστα bytes. Κάποια τέτοια αρχεία μπορεί να είναι textures για κάποιο παιχνίδι (Σχήμα 3.15).

Σχήμα 3.15: Ανάκτηση πληροφοριών από τα android assets

3.5.3 Σύγκριση Android resources & assets

Τα android resources, χρησιμοποιούνται περισσότερο κυρίως λόγω του ευκολότερου API και της ταχύτητας σε σχέση με τα assets. Κατά τη διάρκεια του compile, τα android assets μεταγλωττίζονται μαζί με τον υπόλοιπο κώδικα, οπότε η πρόσβαση σε αυτά είναι αρκετά γρηγορότερη. Αντίθετα, τα android assets, έχουν ένα επιπλέον κόστος στην ταχύτητα λόγω ότι υπολογίζονται κατά τη διάρκεια που τρέχει η εφαρμογή.

3.6 WEBVIEW

Η κλάση *Webview* του Android, μας δίνει τη δυνατότητα να ενσωματώσουμε έναν απλοποιημένο web browser στην εφαρμογή.

Στις περισσότερες περιπτώσεις προτείνεται η χρήση του default web browser της συσκευής. Παρόλα αυτά υπάρχουν καταστάσεις για τις οποίες η χρήση της κλάσης μπορεί να βοηθήσει.

Σχήμα 3.16: Φόρτωση αρχείου στο webview

Στο παραπάνω παράδειγμα (Σχήμα 3.16), φτιάχνουμε ένα αντικείμενο της κλάσης *Webview* και έπειτα καλούμε τη μέθοδο loadUrl ώστε να φορτώσουμε ένα HTML αρχείο. Η τοποθεσία του HTML αρχείου βρίσκεται στον φάκελο android assets και έχει όνομα home.html

3.6.1 Σημαντικές μέθοδοι

- loadUrl
  Φορτώνει ένα URL στο webview.
- addJavascriptInterface
  Συνδέει ένα αντικείμενο Java με το webview. Μέσω αυτού του αντικειμένου, μπορεί το webview να επικοινωνήσει με την εφαρμογή.
- canGoBack
  Μέθοδος που επιστρέφει true, αν υπάρχει τουλάχιστον μια σελίδα στο backstack.
- canGoForward
  Μέθοδος που επιστρέφει true, αν υπάρχει τουλάχιστον μια σελίδα στο forward stack.
- clearHistory
  Καθαρίζει το back/forward stack.
- setWebViewClient
  Ορίζει τον WebViewClient για το webview. Μέσω αυτού του αντικειμένου, υπάρχουν διάφορα callbacks.
- destroy
  Καταστρέφει το webview.

3.6.2 Επικοινωνία webview με την εφαρμογή

Για να μπορέσει η εφαρμογή να αλληλεπιδράσει με την εφαρμογή, θα πρέπει να οριστεί ένα *JavascriptInterface* (Σχήμα 3.17).

```
public class JavascriptInterface {

    protected Context context;
    protected DefaultFragment parent;


    public JavascriptInterface(Context context, DefaultFragment fragment) {
        this.context = context;
        this.parent = fragment;
    }

    @android.webkit.JavascriptInterface
    public void navigateToLink(String link) {
        Intent browserIntent = new Intent("android.intent.action.VIEW", Uri.parse(link));
        parent.startActivity(browserIntent);
    }
}
```

Σχήμα 3.17: Επικοινωνία webview με την android εφαρμογή(½)

Η παραπάνω κλάση προσφέρει στο webview όλες τις μεθόδους της που έχουν το annotation *@android.webkit.JavascriptInterface*.

```
<html>

    <head>
        <script type="text/javascript">
            function navigateToLink(){
                Android.navigateToLink("http://www.google.gr");
            }
        </script>
        <style>

        </style>
    </head>
    <body>

    </body>
</html>
```

Σχήμα 3.18: Επικοινωνία webview με την android εφαρμογή(2/2)

Στο παραπάνω html αρχείο (Σχήμα 3.18) υπάρχει μια Javascript μέθοδος που καλεί τη μέθοδο *navigateToLink* του *JavascriptInterface* που ορίσαμε προηγουμένως. Για να μπορέσει να καλέσει μεθόδους του *JavascriptInterface* πρέπει να χρησιμοποιήσει το namespace *Android*.

3.7 GOOGLE MAP

Η Google προσφέρει ένα πλούσιο SDK, για την ενσωμάτωση χαρτών στις εφαρμογές Android. Το API χειρίζεται τη διαδικασία επικοινωνίας με τους servers που περιέχουν τους χάρτες. Μέσω αυτού του API, ο χρήστης μπορεί να κατεβάσει τον χάρτη, να προσθέσει ένα σημάδι, να κάνει μεγέθυνση και γενικώς να αλληλεπιδράσει πλήρως με τον χάρτη.

Το πρώτο και πιο σημαντικό βήμα για να προσθέσει ο προγραμματιστής το Google Maps SDK στην εφαρμογή του, είναι να δημιουργήσει ένα API κλειδί για την επικοινωνία του με τους servers. Για να εκδώσει ένα τέτοιο κλειδί αρκεί να ακολουθήσει τις οδηγίες στο Google Cloud Platform Console.

```
public class MapsActivity extends AppCompatActivity implements OnMapReadyCallback {
    // ...
    // ...

    @Override
    public void onMapReady(GoogleMap googleMap) {
        this.googleMap = googleMap;

        LatLng IHU = new LatLng(40.655813, 22.806293);
        this.googleMap.addMarker(new MarkerOptions()
                .position(sydney)
                .title("Marker in IHU of Thessaloniki"));
        this.googleMap.moveCamera(CameraUpdateFactory.newLatLng(IHU));
    }
}
```

Σχήμα 3.19: Χρήση του google maps SDK

Στο παραπάνω παράδειγμα (Σχήμα 3.19), έχουμε ένα activity που υλοποιεί το *OnMapReadyCallback* interface.Μόλις το SDK, επικοινωνήσει επιτυχώς με το API, καλείται αυτόματα η μέθοδος *onMapReady*. Αυτή η μέθοδος μας δίνει σαν παράμετρο το *GoogleMap*. Έπειτα, φτιάχνουμε ένα αντικείμενο τύπο *LatLng* και του δίνουμε την τοποθεσία του Διεθνές Πανεπιστημίου Θεσσαλονίκης. Στη συνέχεια φτιάχνουμε ένα σημάδι σε αυτές τις συντεταγμένες και δίνουμε οδηγία στον χάρτη να μετακινηθεί εκεί.

3.7.1 Σημαντικές μέθοδοι

- addMarker
  Προσθέτει ένα σημάδι στον χάρτη.
- animateCamera
  Μετακινεί ‘ομαλά’ τον χάρτη από την τρέχουσα τοποθεσία σε μια καινούρια
- clear
  Αφαιρεί όλα τα πρόσθετα γραφικά στοιχεία από τον χάρτη.
- getCameraPosition
  Επιστρέφει την τρέχουσα τοποθεσία του χάρτη
- getMyLocation
  Επιστρέφει την τοποθεσία του χρήστη
- getUiSettings
  Επιστρέφει τις ρυθμίσεις για τη διεπαφή του χάρτη.
- moveCamera
  Μετακινεί τον χάρτη από την τρέχουσα τοποθεσία σε μια καινούρια
- snapshot
  Δημιουργεί ένα στιγμιότυπο του χάρτη.

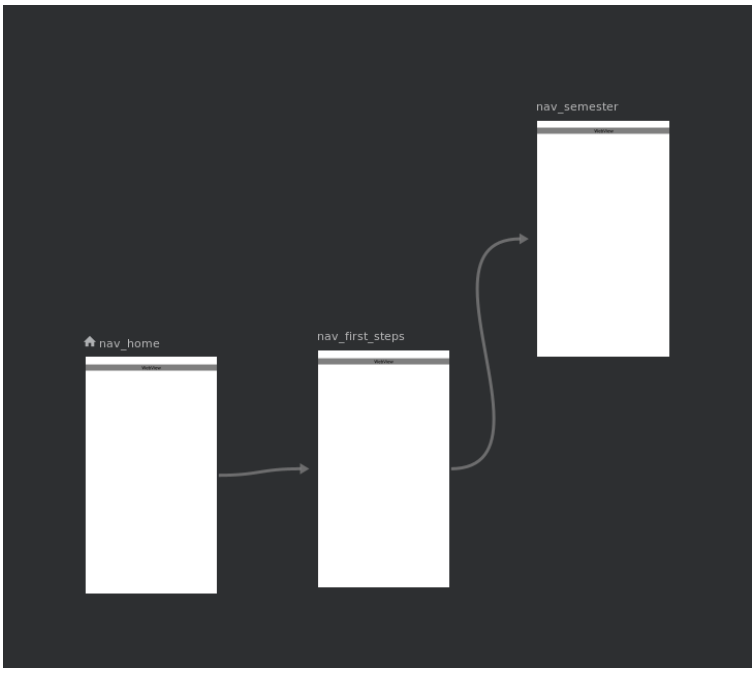
3.8 NAVIGATION

Η πλοήγηση ή navigation αναφέρεται στις διάφορες αλληλεπιδράσεις που κάνει ο χρήστης ώστε να μετακινηθεί στις διάφορες τοποθεσίες της εφαρμογής. Το Android Navigation αποτελείται από τρία κύρια κομμάτια (Σχήμα 3.20):

- Γράφος Πλοήγησης (Navigation Graph): Πρόκειται για ένα XML αρχείο που περιέχει όλη τη σχετική πληροφορία σχετικά με τις διάφορες τοποθεσίες της εφαρμογής και τις πιθανές διαδρομές μεταξύ τους (Σχήμα 3.21).
- NavHost: Πρόκειται για ένα άδειο layout που εμφανίζει τις διαδρομές για τον γράφο πλοήγησης.
- NavController: Πρόκειται για ένα αντικείμενο που ελέγχει την πλοήγηση ανάμεσα στα fragment.

Χρησιμοποιώντας το navigation component, υπάρχουν πολλαπλά οφέλη σε σχέση με τις εναλλακτικές. Κάποια από αυτά είναι:

- Διαχείριση των αλλαγών των fragment.
- Διαχείριση των ‘πάνω’ και ‘πίσω’ ενεργειών με τον σωστό και προτεινόμενο τρόπο.
- Διαχείριση των ‘βαθιών’ συνδέσεων. Βαθιά σύνδεση είναι, η αυτόματη μεταφορά σε μια συγκεκριμένη οθόνη από οποιοδήποτε κατάσταση της συσκευής.



Σχήμα 3.20: Γράφος που δείχνει τις διάφορες δυνατές ενέργειες μεταξύ των fragments της εφαρμογής

Στο παραπάνω παράδειγμα, βλέπουμε έναν τυπικό γράφο πλοήγησης. Παραπάνω φαίνεται σε γραφική μορφή, αλλά στην ουσία είναι ένα XML έγγραφο.

```
protected void onCreate(Bundle savedInstanceState) {
        // ...
        // ...
        NavigationView navigationView = findViewById(R.id.nav_view);
        NavController navController = Navigation.findNavController(this, R.id.nav_host_fragment);
        NavigationUI.setupWithNavController(navigationView, navController);
}
```

Σχήμα 3.21: Γράφος πλοήγησης σε XML

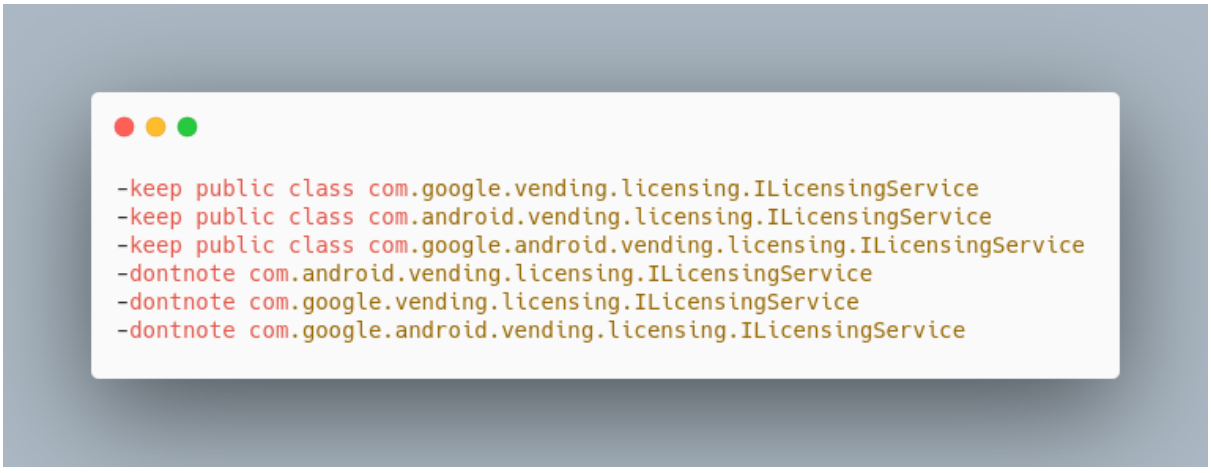
Στο παραπάνω παράδειγμα βλέπουμε πώς συνδέουμε τον γράφο με ένα activity.

3.9 PROGUARD

Το proguard είναι ένας από τους πιο δημοφιλής βελτιστοποιητές για Java Bytecode. Χρησιμοποιώντας το proguard, μπορούμε να κάνουμε την εφαρμογή μας πιο μικρή σε μέγεθος και πιο γρήγορη. Χρησιμοποιώντας το proguard, έχουμε το επιπρόσθετο πλεονέκτημα να κάνουμε την εφαρμογή πιο δυνατή σε επιθέσεις που χρησιμοποιούν reverse engineering.

3.9.1 Πιθανά μειονεκτήματα

- Η πιθανή εσφαλμένη διαμόρφωση μπορεί να προκαλέσει crash στην εφαρμογή.
- Χρειάζεται επιπλέον τεστς.
- Τα μηνύματα λάθους που προκαλούνται από Exceptions είναι δύσκολο να διαβαστούν αφού τα όνομα των κλάσεων, μεθόδων και μεταβλητών έχουν ελαχιστοποιηθεί.



Σχήμα 3.22: Αρχείο με κανόνες για το proguard

Στο παραπάνω παράδειγμα (Σχήμα 3.22) υπάρχουν κάποια proguard rules για το ποιές κλάσεις δε θέλουμε να ελαχιστοποιήσει.

ΕΠΙΛΟΓΟΣ

Σε αυτήν την ενότητα αναλύθηκαν οι διάφορες σημαντικές τεχνολογίες που χρησιμοποιούνται στο Android και οι σημαντικότερες κλάσεις που χρησιμοποιήθηκαν στην εφαρμογή

ΚΕΦΑΛΑΙΟ 4. WEB CLIENT

ΕΙΣΑΓΩΓΗ

Σε αυτή την ενότητα θα γίνει μια βασική εισαγωγή στις κύριες τεχνολογίες που σχετίζονται με την εφαρμογή για Web και χρησιμοποιήθηκαν για την εκπόνηση αυτής της εργασίας.

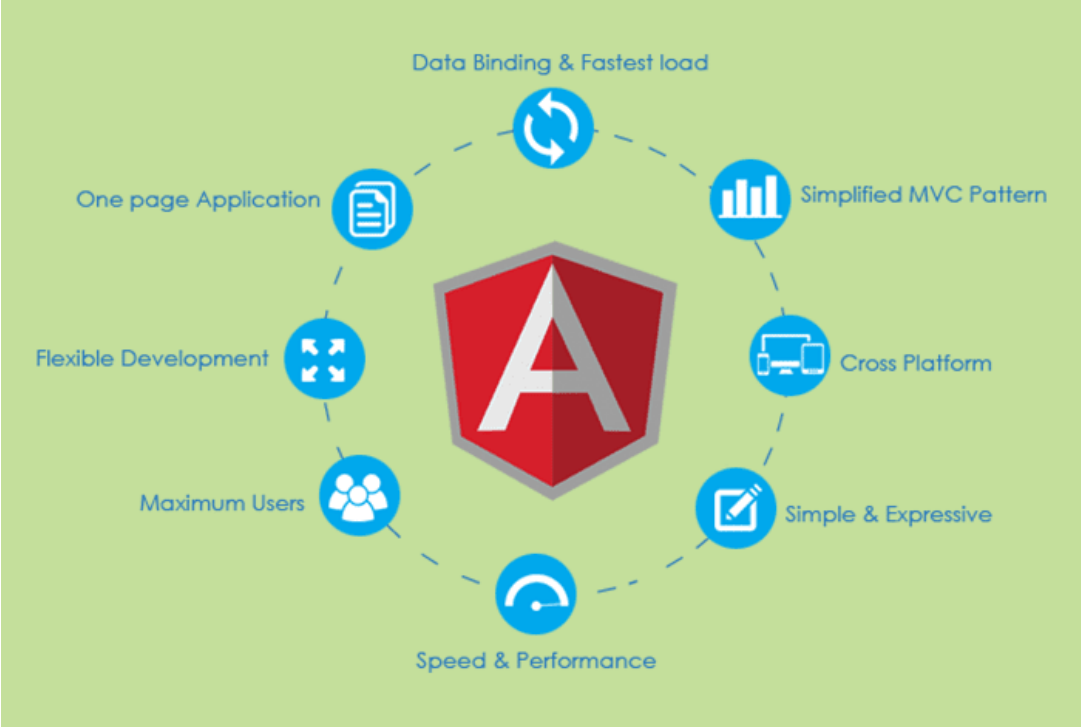
4.1 ANGULAR

Η Angular είναι ένα framework για τη δημιουργία αποδοτικών εφαρμογών. Η Angular δημιουργήθηκε από μια ομάδα της Google η οποία επίσης είχε φτιάξει το framework AngularJS. Η Angular δημιουργήθηκε από την αρχή σαν ξεχωριστό project, και παρουσιάζει πάρα πολλές διαφορές με την AngularJS.

Η πιο πρόσφατη έκδοση της Angular είναι η 11 η οποία δημοσιεύθηκε στις 11 Νοεμβρίου του 2020.

Κάποια από τα σημαντικότερα στοιχεία της Angular είναι (Σχήμα 4.1):

- Δουλεύει σε όλες τις ‘κύριες’ πλατφόρμες:
    - Δημιουργία μοντέρνων διαδικτυακών εφαρμογών χρησιμοποιώντας τις λειτουργίες που προσφέρουν τα web standards
    - Δημιουργία εφαρμογών για κινητά με μετατροπή κώδικα ανάλογα την πλατφόρμα για την καλύτερη αποδοτικότητα.
    - Δημιουργία εφαρμογών για desktop για όλα τα ‘κύρια’ λειτουργικά συστήματα όπως Windows και Mac με ακριβώς τον ίδιο κώδικα.
- Εξαιρετική ταχύτητα και απόδοση
    - Παραγωγή κώδικα για τις διάφορες πλατφόρμες μεταφράζοντας τα Angular templates
    - Σερβίρισμα της εφαρμογής μέσω οποιοδήποτε server.
    - Διαχωρισμός των κομματιών κώδικα ώστε οι χρήστες να φορτώνουν μόνο το υλικό που χρειάζονται.
- Παραγωγικότητα προγραμματιστή
    - Δημιουργία διεπαφών με ευκολία λόγω του συντακτικού
    - Χρησιμοποιώντας το λογισμικό γραμμής εντολών της Angular, είναι εύκολη η δημιουργία, αναβάθμιση και το χτίσιμο εφαρμογών.
    - Υπάρχει πληθώρα από επεκτάσεις για τα IDEs με τα οποία γίνεται εύκολη η συμπλήρωση κώδικα, αναφορά λαθών κ.α.



Σχήμα 4.1: Δυνατότητες της Angular

4.2 ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΜΕ ΤΟ FRAMEWORK ANGULAR

Για την ανάπτυξη της web εφαρμογής χρησιμοποιήθηκε η Angular 11 και οι σύγχρονες βιβλιοθήκες που περιλαμβάνει.

4.2.1 ANGULAR CLI

Το Angular CLI είναι διεπαφή γραμμής εντολών για αρχικοποίηση, παραγωγή και συντήρηση εφαρμογών με το framework Angular. Η εγκατάσταση του Angular CLI γίνεται μέσω του NPM (Σχήμα 4.2).

Σχήμα 4.2: Εγκατάσταση του εργαλείου angular CLI

Οι βασικότερες λειτουργίες του Angular CLI είναι:

- new
  Για τη δημιουργία ενός καινούργιου Angular project
- serve
  Για το χτίσιμο του Angular project και την έκθεση του μέσω HTTP στο τοπικό περιβάλλον
- build
  Για το χτίσιμο του Angular project
- lint
  Για την εκκίνηση των εργαλείων για τη συμμόρφωση του project με κάποιους κανόνες
- test
  Για την εκκίνηση των test για το project
- update
  Για την αναβάθμιση του project σε μια άλλη έκδοση

4.2.2 ANGULAR HTTP CLIENT

Η Angular προσφέρει έναν απλοποιημένο HTTP client για την επικοινωνία με άλλους servers. Ο HTTP client της Angular προσφέρει τα παρακάτω κύρια στοιχεία:

- Η δυνατότητα για HTTP requests χρησιμοποιώντας object με στατικούς τύπους (Σχήμα 4.3).
- Ευέλικτο API για τη διαχείριση λαθών
- Προσφέρει δυνατότητες για εύκολο testing
- Δυνατότητες παρέμβασης σε διάφορα στάδια του HTTP request και response

```
findPost(id:string){

 return this.http
  .post<any>('https://example.com/posts',
    { id: post.id }
  );

}
```

Σχήμα 4.3: Παράδειγμα χρήσης του Angular HTTP client

4.2.3 ANGULAR COMPONENTS

Τα Angular components είναι τα βασικά συστατικά στοιχεία για τη δημιουργία εφαρμογών Angular. Κάθε Angular component αποτελείται από:

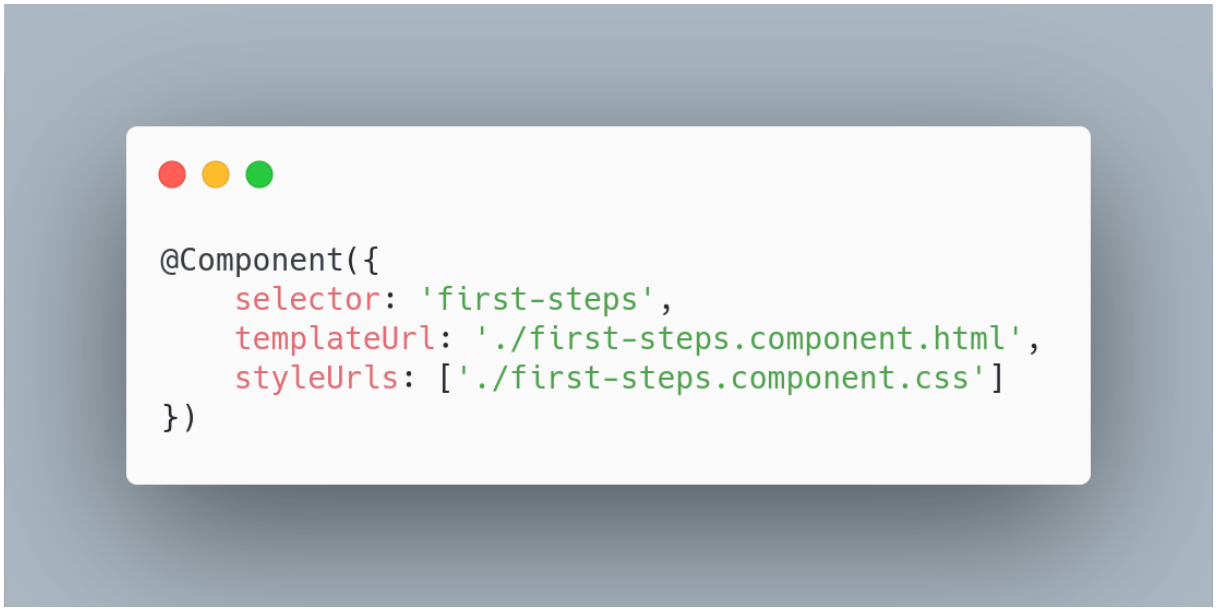
- Ένα πρότυπο HTML που δηλώνει τι θα εμφανιστεί στη σελίδα
- Μια κλάση γραμμένη στη γλώσσα Typescript που καθορίζει τη συμπεριφορά
- Ένα αρχείο CSS που δηλώνει πώς θα εμφανιστούν τα συστατικά HTML

Ο ευκολότερος τρόπος για τη δημιουργία ενός Angular component είναι μέσω του εργαλείου Angular CLI. Για να γίνει αυτό θα πρέπει να εκτελεστεί η εντολή generate και να περαστεί σαν παράμετρος το όνομα του component (Σχήμα 4.4).

```
ng generate component first-steps-of-student
```

Σχήμα 4.4: Δημιουργία Angular component μέσω του εργαλείου Angular CLI

Για τη σύνδεση των διαφόρων συστατικών στοιχείων ενός Angular component χρησιμοποιούνται κάποιοι Angular decorators (Σχήμα 4.5).

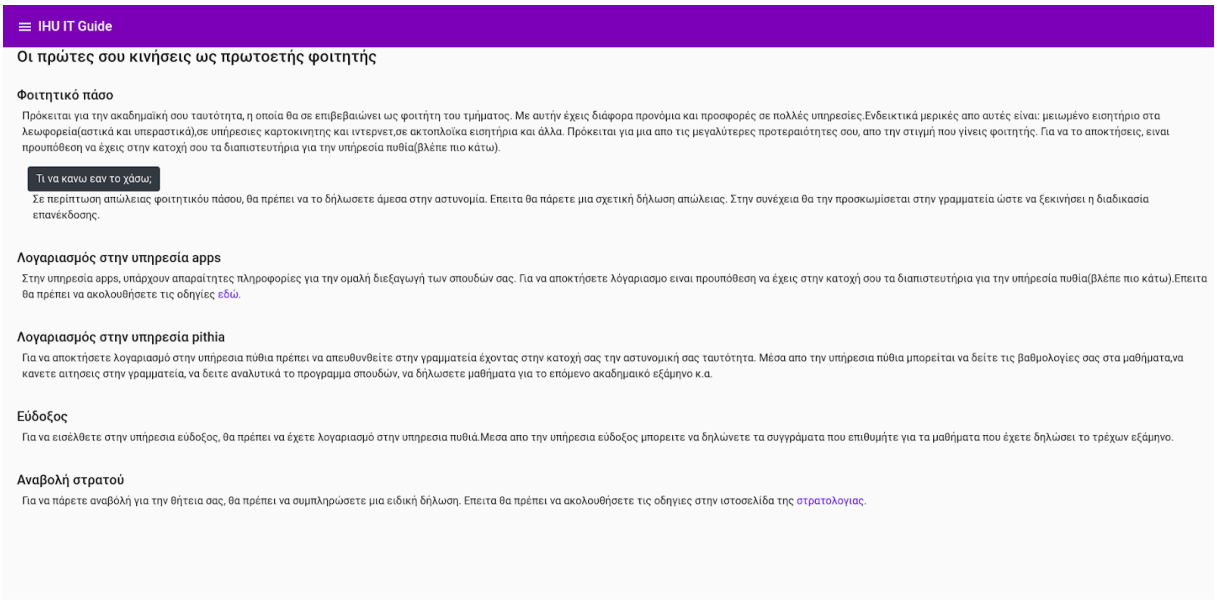


Σχήμα 4.5: Εμπλουτισμός Angular Component με τα απαραίτητα συνοδευτικά αρχεία

4.3 WEB APPLICATION

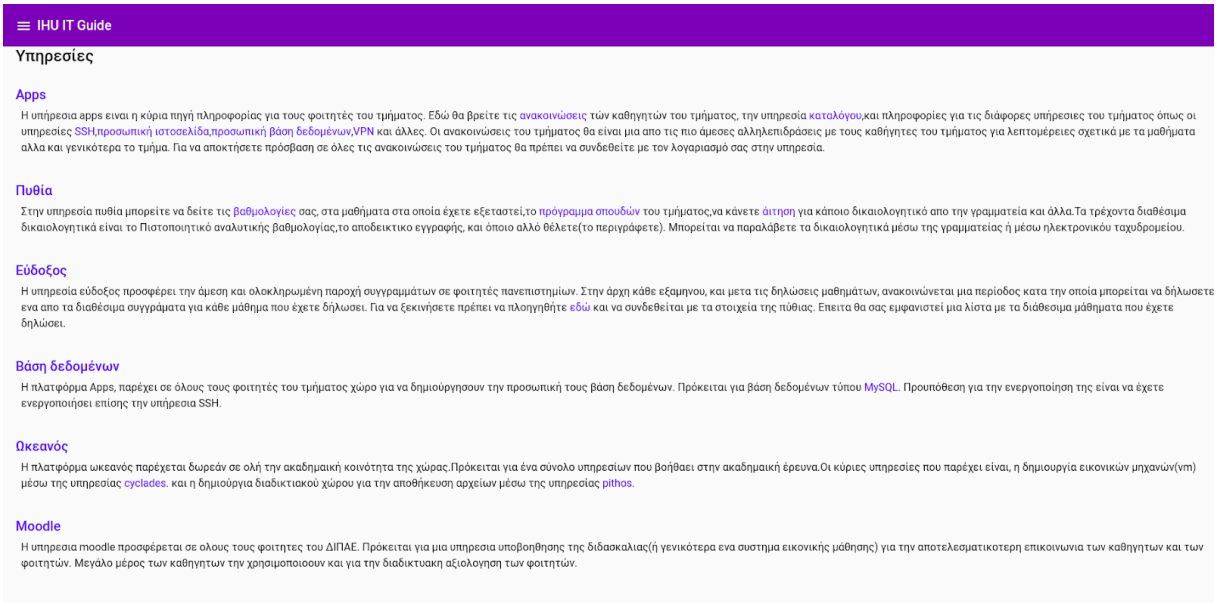
Η εφαρμογή web δημιουργήθηκε μετά την εφαρμογή για Android. Ο τρόπος με τον οποίο αναπτύχθηκαν και οι δύο εφαρμογές έγινε με τέτοιο τρόπο ώστε να είναι εύκολη η μεταγενέστερη υλοποίηση της εφαρμογής σε οποιαδήποτε πλατφόρμα. Γι'αυτόν τον λόγο η εφαρμογή web υλοποιήθηκε σε πολύ μικρότερο χρόνο σχετικά με την εφαρμογή Android.

Παρακάτω θα παρουσιαστούν μερικές οθόνες της εφαρμογής web.



Σχήμα 4.6: Οθόνη πρώτα βήματα της εφαρμογής web

Παραπάνω φαίνεται η οθόνη (Σχήμα 4.6) που περιλαμβάνει τα πρώτα βήματα του φοιτητή. Οι πληροφορίες που περιλαμβάνει είναι ακριβώς οι ίδιες που υπήρχαν στην αντίστοιχη οθόνη του Android client.



Σχήμα 4.7: Οθόνη υπηρεσίες της εφαρμογής web

Στην παραπάνω εικόνα (Σχήμα 4.7) φαίνεται η οθόνη με τις υπηρεσίες του τμήματος.

Όπως παρατηρούμε από τα παραπάνω παραδείγματα, οι οθόνες της εφαρμογής android και web είναι σχεδόν πανομοιότυπες. Στην πραγματικότητα αλλαγές υπάρχουν μόνο όταν χρειάζεται για λόγους ευχρηστίας λόγω διαφορετικής πλατφόρμας. Για παράδειγμα, οι οθόνες των σταθερών υπολογιστών είναι τυπικά μεγαλύτερες από τις οθόνες των κινητών τηλεφώνων οπότε και μπορούν να παρουσιάσουν περισσότερη πληροφορία κάθε στιγμή.

ΕΠΙΛΟΓΟΣ

Σε αυτήν την ενότητα αναλύθηκε το framework Angular και οι διάφορες χρησιμότητες που προσφέρει. Επίσης, αναλύθηκαν κάποια από τα κύρια στοιχεία που προφέρονται στον προγραμματιστή για την αποδοτική ανάπτυξη εφαρμογών.

ΚΕΦΑΛΑΙΟ 5. ΤΕΧΝΙΚΗ MVP

ΕΙΣΑΓΩΓΗ

Το MVP ή Model-View-Presenter είναι μια αρχιτεκτονική για τη σχεδίαση σύγχρονων και ευέλικτων προγραμμάτων. Πρόκειται για μια παραλλαγή του γνωστότερου MVC(ή Model-View-Controller). Σύμφωνα με την αρχιτεκτονική MVP, όλη (ή σχεδόν όλη) η λογική της εφαρμογής βρίσκεται στον presenter. Γενικότερα, όταν ο χρήστης κάνει μια ενέργεια στη διεπαφή, τότε πρέπει να ειδοποιήσει τον presenter για αυτή την ενέργεια. Έπειτα ο presenter, συμβουλευόμενος από το μοντέλο θα πρέπει να υποδείξει στη διεπαφή τι ενέργειες πρέπει να κάνει.

5.1 MODEL

Το μοντέλο μιας εφαρμογής αποτελείται από διάφορες κλάσεις που περιγράφουν τις σχετικές οντότητες.

```java
public class Announcement {
    private String id;
    private String title;
    private String author;
    private String body;
    private String category;
    private String link;

    Announcement(String id, String title, String author, String body, String category, String link) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.body = body;
        this.category = category;
        this.link = link;
    }

    public String id() {
        return id;
    }

    public String title() {
        return title;
    }

    public String author() {
        return author;
    }

    public String body() {
        return body;
    }

    public String category() {
        return category;
    }

    public String link() {
        return link;
    }
}
```

Σχήμα 5.1: Οντότητα ανακοίνωση

Η παραπάνω κλάση περιγράφει μια υποθετική οντότητα μιας Ανακοίνωσης (Σχήμα 5.1). Η ανακοίνωση έχει διάφορες ιδιότητες που την χαρακτηρίζουν όπως ο τίτλος και η κατηγορία.

Γενικότερα, το μοντέλο περιέχει το business logic. Περιγράφει δηλαδή τη ζωή των αντικειμένων του μοντέλου και πως αυτά αλλάζουν κατά την πάροδο του χρόνου.

Το μοντέλο συνήθως σώζεται και σε μια μόνιμη πηγή, όπως μια βάση δεδομένων, ώστε να είναι προσπελάσιμο ακόμα και όταν χαθεί από τη μνήμη της εφαρμογής. Την αποθήκευση και ανάκτηση του μοντέλου έχει ως ευθύνη ο controller.

5.2 VIEW

Το view μιας εφαρμογής αποτελείται από οντότητες που περιγράφουν τις διεπαφές που βλέπει ο χρήστης. Για παράδειγμα, αν προγραμματίζουμε μια διαδικτυακή εφαρμογή, τότε σαν view θεωρούνται τα .html αρχεία, ενώ αν προγραμματίζουμε μια android εφαρμογή, σαν view θεωρούνται τα .xml αρχεία.

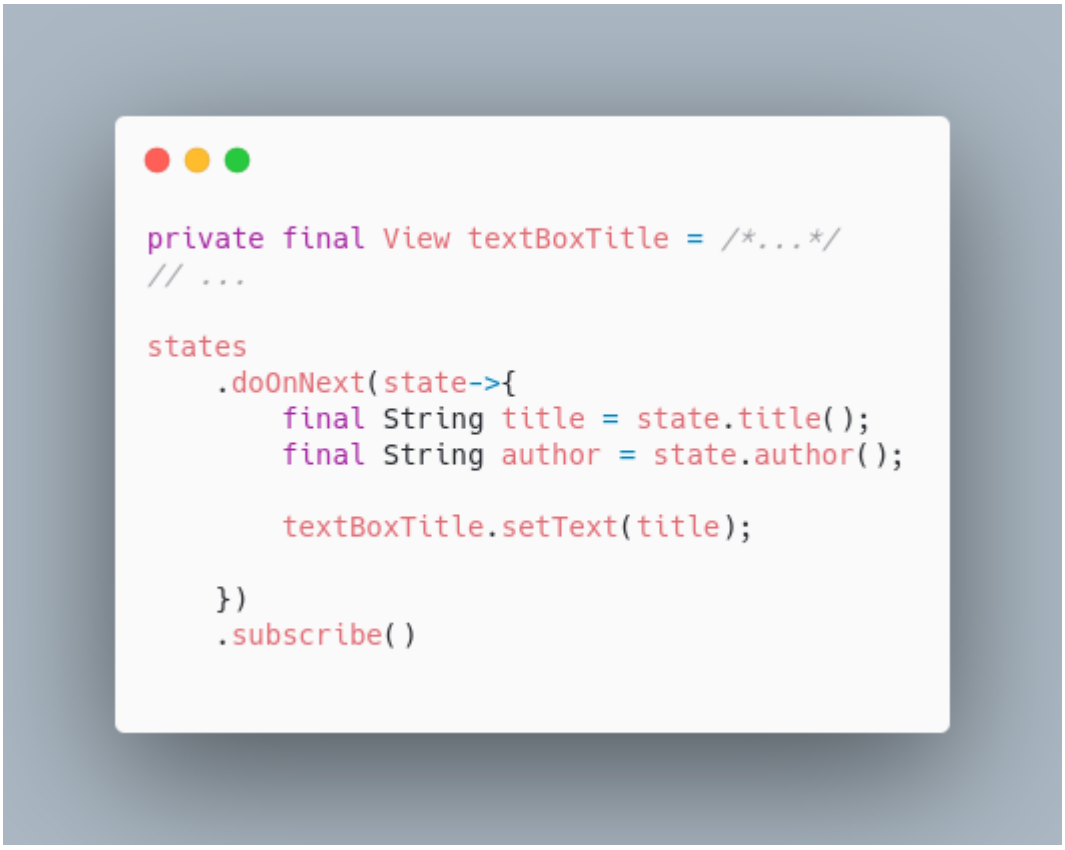
Το view component στο Android, πέρα από τα xml αρχεία που περιέχουν την εμφάνισης της διεπαφής, συνδέεται συνήθως και με ένα fragment ή activity. Στο συνδεδεμένο fragment / activity περιέχεται όλη η διασύνδεση του χρήστη με τη διεπαφή.

Όταν γίνεται μια ενέργεια από τον χρήστη στη διεπαφή, τότε ειδοποιείται το view και στέλνει ένα μήνυμα με τις σχετικές πληροφορίες στον presenter. Εκείνος μετά από διάφορους ελέγχους και υπολογισμούς, απαντά στο view, με τις ενέργειες που πρέπει να κάνει ώστε να αλλάξει καταλήλως η διεπαφή.

```
class AnnouncementsFragment {
 // ...
 @Inject
 AnnouncementsPresenter presenter;


 @Override
 public void onStart(){
  super.onStart();
  final AnnouncementsEvent event = new AnnouncementsInitViewEvent();
  System.out.println("Publishing Initial Event");
  presenter.publishEvent(event);
 }

}
```

Σχήμα 5.2: Επικοινωνία του fragment με τον presenter

Στο παραπάνω παράδειγμα (Σχήμα 5.2), βλέπουμε ένα fragment. Κάνοντας χρήση του dependency injection(DI) αρχικοποιείτε ο αντίστοιχος presenter. Στη μέθοδο onStart, η οποία είναι μέρος του lifecycle του κάθε fragment, δημιουργείται ένα κατάλληλο event, και γίνεται publish στον presenter.

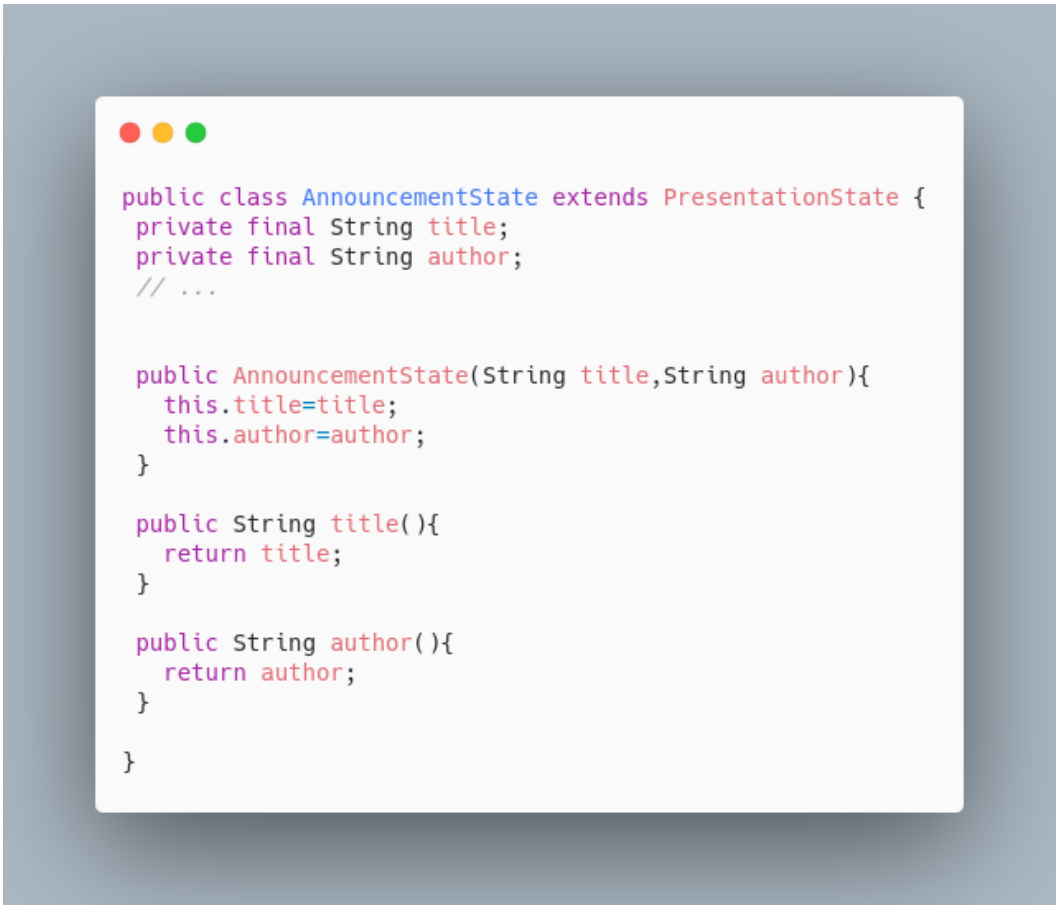


```
private final View textBoxTitle = /*...*/
// ...

states
    .doOnNext(state->{
        final String title = state.title();
        final String author = state.author();

        textBoxTitle.setText(title);

    })
    .subscribe()
```

Σχήμα 5.3: Επικοινωνία του presenter με το fragment(½)

Στο παραπάνω παράδειγμα (Σχήμα 5.3), βλέπουμε τον τρόπο με τον οποίο το view καταλαβαίνει ότι έγινε μια αλλαγή. Μόλις ο presenter πάρει ένα event, αποφασίζει τις ενέργειες που πρέπει να κάνει για να εμφανίσει την κατάλληλη απάντηση στον χρήστη και παράγει ένα *state* το οποίο και προωθεί στο view. Ο κώδικας που περιέχεται μέσα στο block του *doOnNext* καλείται κάθε φορά που ο presenter προωθεί ένα καινούργιο state. Για αυτό το λόγο σε εκείνο το σημείο, γίνεται συνήθως και έλεγχος για το αν το state περιέχει errors.

Γενικότερα, στην περίπτωση που υπάρχει κάποιο σφάλμα με το event που πηγαίνει στον presenter, δεν πετάγεται κάποιο exception. Αντί για αυτό πετάγεται ένα ειδικό state που ονομάζεται failed state, και περιέχει το είδος του λάθους, και σχετικές με αυτό πληροφορίες.



```
public class AnnouncementState extends PresentationState {
 private final String title;
 private final String author;
 // ...


 public AnnouncementState(String title,String author){
   this.title=title;
   this.author=author;
 }

 public String title(){
   return title;
 }

 public String author(){
   return author;
 }

}
```

Σχήμα 5.4: Δομή μηνυμάτων που παράγει ο presenter

Το state περιγράφει την κατάσταση που πρέπει να βρίσκεται το view κάθε στιγμή. Στο παραπάνω παράδειγμα (Σχήμα 5.4), το state περιέχει τις πληροφορίες σχετικά με τον τίτλο και τον συντάκτη της ανακοίνωσης. Το view, με βάση αυτές τις πληροφορίες προσαρμόζει τη διεπαφή ανάλογα.

```
class AnnouncementDoesNotExistFailedState extends AnnouncementFailedState{
  private String announcementId;

  public AnnouncementDoesNotExistFailedState(String announcementId){
   this.announcementId = announcementId;
  }

  public announcementId(){
   return announcementId;
  }
}
```

Σχήμα 5.5: Δομή μηνυμάτων λάθους που παράγει ο presenter.

Το παραπάνω state θα επιστρεφόταν αν δεν υπήρχε ο χρήστης στη βάση δεδομένων της εφαρμογής (Σχήμα 5.5).

Για να χειριστεί το παραπάνω state το view θα έπρεπε να μετατραπεί στο παρακάτω (Σχήμα 5.6).

```
private final View textBoxError = /*...*/
// ...

states
    .doOnNext(state->{
        if (state instanceof AnnouncementFailedState){
            AnnouncementFailedState failedState = (AnnouncementFailedState)state;
            textBoxError.setText("Announcement with id " + failedState.announcementId +
                "wasn't found";
            return;
        }
        // ...
    })
    .subscribe();
```

Σχήμα 5.6: Επικοινωνία του fragment με τον presenter(2/2)

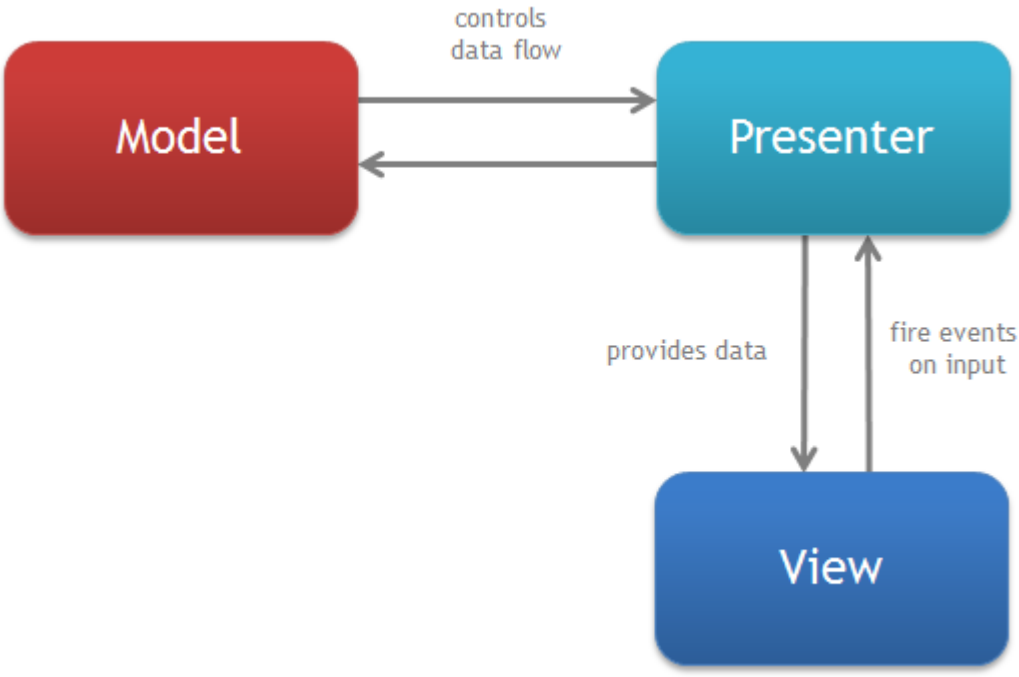
5.3 PRESENTER

Ο presenter μιας εφαρμογής σε ένα σύστημα MVP είναι η οντότητα που δέχεται σαν είσοδο τις ενέργειες του χρήστη και είναι υπεύθυνος να αποφασίσει τις ενέργειες που πρέπει να γίνουν.
Πιο συγκεκριμένα, όταν ο χρήστης κάνει μια ενέργεια στη διεπαφή, ειδοποιείται ο presenter, και έπειτα συμβουλευόμενος το μοντέλο, απαντά στο view με ένα state. Με βάση αυτό το state, το view τροποποιεί τη διεπαφή.

```
class AnnouncementsPresenter extends DefaultPresenter{
 final Database database = /* ... */

 @Override
 public void publishEvent(PresentationEvent event){
  if (event instance of AnnouncementInitViewEvent){
   final String announcementId = event.announcementId();
   final Announcement announcement = database.findById(announcementId);
   if (announcement == null){
    super.nextState(new AnnouncementDoesNotExistFailedState(announcementId));
    return;
   }
   super.nextState(new AnnouncementState(announcement.title(),announcement.author()));
  }

 }

}
```

Σχήμα 5.7: Υλοποίηση του presenter

Στο παραπάνω παράδειγμα (Σχήμα 5.7) βλέπουμε πως υλοποιείται η μέθοδος publishEvent ενός presenter. Σαν παράμετρο δέχεται το γενικό PresentationEvent. Έπειτα ανάλογα με τη συγκεκριμένη υποκλάση του, γίνονται ανάλογες ενέργειες. Εφόσον το event είναι τύπο *UserDetailsInitViewEvent*, ο presenter μέσω ενός instance της βάσης δεδομένων, προσπαθεί να βρει τον χρήστη με το userId που περιέχεται στο event. Αν δεν υπάρχει, τότε δημιουργείται ένα καινούργιο *failedState*. Στην περίπτωση που υπάρχει, δημιουργείται ένα καινούριο *UserDetailsState*. Και στις δύο περιπτώσεις, το event στέλνεται σε ένα subject, το οποίο το προωθεί σε όλους τους subscribers του.



Σχήμα 5.8: Διάγραμμα επικοινωνίας μεταξύ των model, view, presenter

5.4 ANDROID IMPLEMENTATION

Για το android implementation του MVP, θα υπάρχουν 2 ξεχωριστές οντότητες. Η μια οντότητα θα είναι ανεξάρτητη από το android και θα ακολουθεί τις αφηρημένες έννοιες του MVP. Η άλλη οντότητα θα είναι δεμένη με τις τεχνολογίες του android και θα έχει ως σημείο αναφοράς την πρώτη.

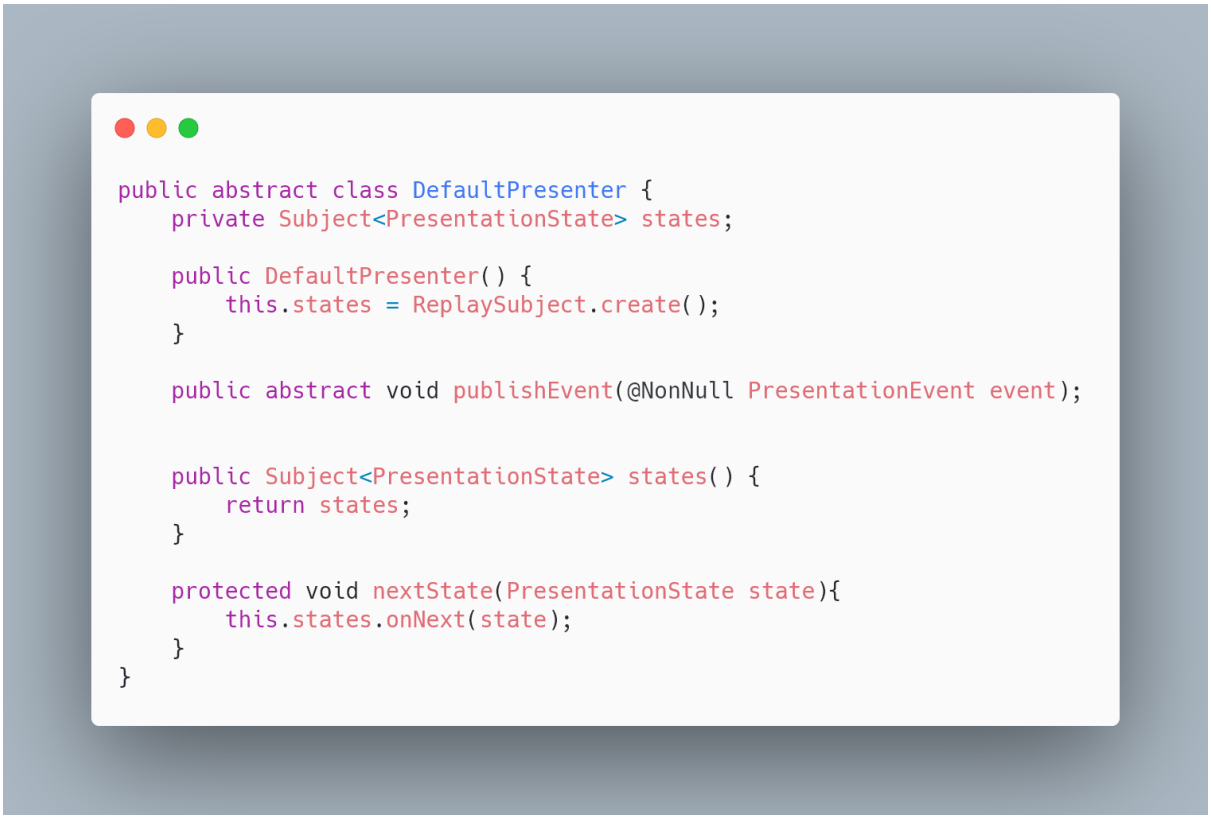
Για να δημιουργηθεί το παραπάνω structure, θα πρέπει να υπάρχει ένα module έξω από το βασικό module που βρίσκεται η android εφαρμογή. Αυτό το module ονομάζεται presentation. Εκεί μέσα βρίσκονται όλες οι αφηρημένες δομές σχετικά με το MVP.

Σχήμα 5.9: Κλάση *PresentationState*

Παραπάνω φαίνεται η αφηρημένη κλάση *PresentationState* (Σχήμα 5.9). Σκοπός αυτής της κλάσης είναι, να περιορίζουμε τον προγραμματιστή στη δημιουργία τέτοιων αντικειμένων (υποκλάσεων της *PresentationState*) και την προώθηση τους στο view.

Σχήμα 5.10: Κλάση PresentationEvent

Παραπάνω φαίνεται η αφηρημένη κλάση *PresentationEvent* (Σχήμα 5.10). Σκοπός αυτής της κλάσης είναι, να περιορίζουμε τον προγραμματιστή στη δημιουργία τέτοιων αντικειμένων (υποκλάσεων της *PresentationEvent*) στο view και τη δημοσίευσή τους στον Presenter.



```
public abstract class DefaultPresenter {
    private Subject<PresentationState> states;

    public DefaultPresenter() {
        this.states = ReplaySubject.create();
    }

    public abstract void publishEvent(@NonNull PresentationEvent event);


    public Subject<PresentationState> states() {
        return states;
    }

    protected void nextState(PresentationState state){
        this.states.onNext(state);
    }
}
```

Σχήμα 5.11: Μέθοδοι ενός presenter

Παραπάνω είναι η αφηρημένη κλάση *DefaultPresenter* (Σχήμα 5.11). Περιέχει ένα subject που περιέχει όλα τα states του view. Η συγκεκριμένη υλοποίηση είναι με *ReplaySubject*. Επίσης, περιέχει την αφηρημένη μέθοδο *publishEvent* που πρέπει να υλοποιήσουν όλες οι κλάσεις που την επεκτείνουν. Κάθε presenter που επεκτείνει τον *DefaultPresenter* υλοποιεί την *publishEvent* διαφορετικά, ανάλογα με τα events τα οποία μπορεί να χειριστεί.

Εκτός από την παραπάνω οντότητα presentation που δε σχετίζεται καθόλου με το android, και μπορεί να χρησιμοποιηθεί πάνω σε οποιαδήποτε τεχνολογία, υπάρχει και ένα άλλο module που ονομάζεται επίσης presentation αλλά βρίσκεται στο ίδιο πακέτο με την εφαρμογή android. Σε αυτό το module, γίνεται η σύνδεση των android components με τα MVP components. Συγκεκριμένα για μια εφαρμογή που περιέχει activities και fragments, αυτό το πακέτο περιέχει τις παρακάτω κλάσεις.

```
public abstract class DefaultActivity extends AppCompatActivity {


    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(layout());


    }

    protected abstract DefaultPresenter getPresenter();


    protected abstract @LayoutRes
    int layout();
}
```

Σχήμα 5.12: Activity που χρησιμοποιεί presenter

Η παραπάνω αφηρημένη κλάση επεκτείνει την *AppCompatActivity* (Σχήμα 5.12). Επίσης, περιέχει την αφηρημένη μέθοδο *getPresenter*. Αυτή η μέθοδος υλοποιείται από όλες τις υποκλάσεις της και επιστρέφει το instance του presenter που περιέχει.

```
public abstract class DefaultFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
Bundle savedInstanceState) {
        return inflater.inflate(layout(), container, false);
    }


    @Override
    public void onStart() {
        super.onStart();
        if (getPresenter() != null) {
            if (initialEvent() != null) {
                System.out.println("Publishing initial event " + initialEvent());
                getPresenter().publishEvent(initialEvent());
            }
        }
    }


    public abstract @LayoutRes
    int layout();


    protected abstract DefaultPresenter getPresenter();

    public abstract PresentationEvent initialEvent();


    public void publishEvent(PresentationEvent event) {
        if (getPresenter() == null){
            return;
        }
        getPresenter().publishEvent(event);
    }

}
```

Σχήμα 5.13: Fragment που χρησιμοποιεί presenter

Παραπάνω φαίνεται η αφηρημένη κλάση *DefaultFragment* (Σχήμα 5.13). Αυτή η κλάση περιέχει την αφηρημένη μέθοδο *initialEvent*. Αυτή η μέθοδος υλοποιείται από τις υποκλάσεις της, και περιέχει το event που πυροδοτεί την αρχικοποίηση των συστατικών της διεπαφής. Επίσης, περιέχει τη μέθοδο *getPresenter* που έχει τον ίδιο ρόλο όπως και στο *DefaultActivity*. Επιπροσθέτως, το *DefaultFragment* υλοποιεί τη μέθοδο *onStart* της κλάσης *Fragment*. Σε αυτή τη μέθοδο στέλνετε το αρχικό event στον presenter.

5.5 ANGULAR IMPLEMENTATION

Η τεχνική MVP υλοποιείται με παρόμοιο τρόπο και στην Angular.

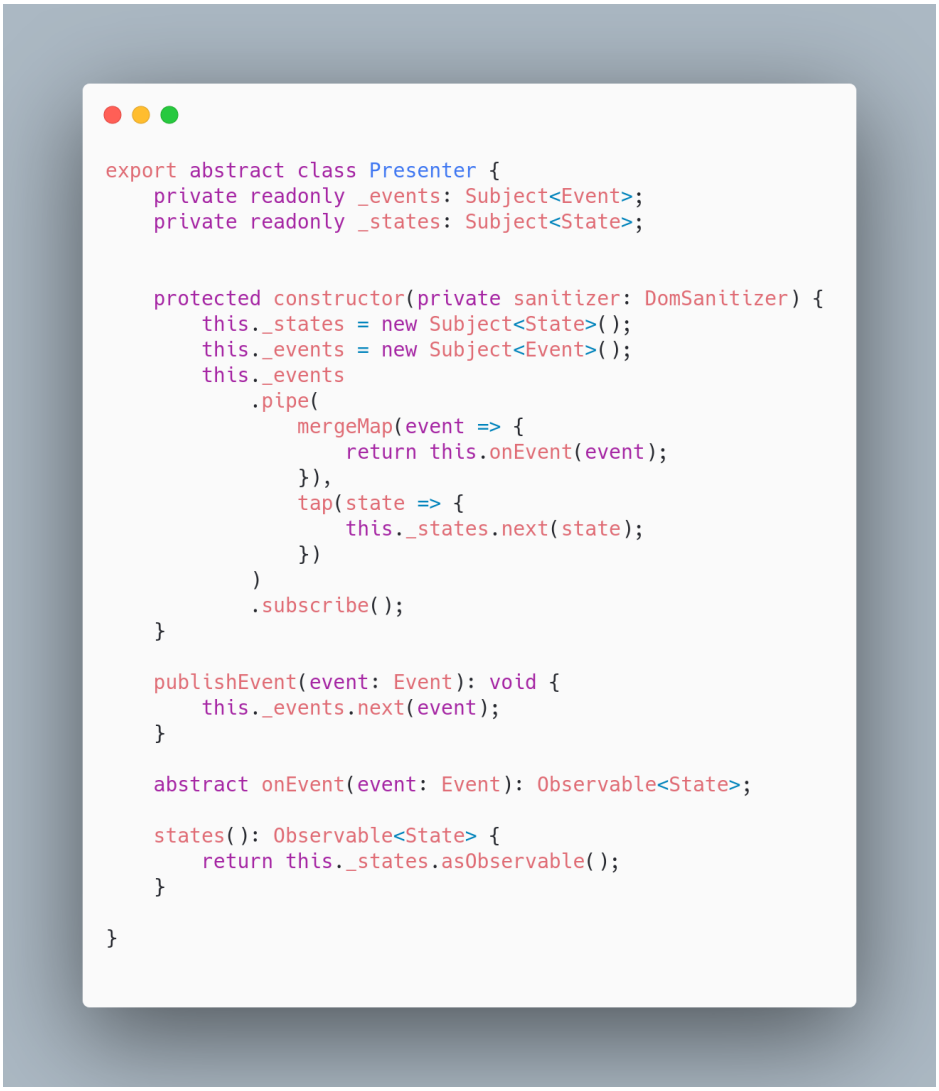
Υπάρχουν οι αφηρημένες κλάσεις *Event* (Σχήμα 5.16), Presenter (Σχήμα 5.17) και *State* (Σχήμα 5.15). Επίσης, αντί για activity και fragment, έννοιες που δεν υπάρχουν στην Angular, υπάρχει το AngularView (Σχήμα 5.14) που είναι η υπερκλάση των component.
Παρακάτω παρουσιάζονται οι βασικές υπερκλάσεις.

```
export abstract class AngularView implements OnInit {
  private readonly statesSubscription: Subscription;

  protected readonly presenter: Presenter;


  protected constructor(presenter: Presenter) {
      this.presenter = presenter;
      this.statesSubscription = new Subscription();
  }

  ngOnInit(): void {
      this.statesSubscription.add(
          this.presenter
              .states()
              .pipe(
                  tap(state => {
                      this.onState(state);
                  })
              )
              .subscribe()
      );
      this.initialEvents()
          .map(event => {
              this.presenter.publishEvent(event);
          })
  }

  abstract onState(state: State): void;

  abstract initialEvents(): Event[];


}
```

Σχήμα 5.14: Υπερκλάση όλων των component

Σχήμα 5.15: Υπερκλάση όλων των states

Σχήμα 5.16: Υπερκλάση όλων των events



```
export abstract class Presenter {
    private readonly _events: Subject<Event>;
    private readonly _states: Subject<State>;


    protected constructor(private sanitizer: DomSanitizer) {
        this._states = new Subject<State>();
        this._events = new Subject<Event>();
        this._events
            .pipe(
                mergeMap(event => {
                    return this.onEvent(event);
                }),
                tap(state => {
                    this._states.next(state);
                })
            )
            .subscribe();
    }

    publishEvent(event: Event): void {
        this._events.next(event);
    }

    abstract onEvent(event: Event): Observable<State>;

    states(): Observable<State> {
        return this._states.asObservable();
    }

}
```

Σχήμα 5.17: Υπερκλάση όλων των presenter

ΕΠΙΛΟΓΟΣ

Σε αυτή την ενότητα αναλύθηκε η αρχιτεκτονική MVP. Αναδείχθηκαν οι διάφορες κλάσεις, οι οποίες απαιτούνται για να υλοποιηθεί και τι ρόλο έχει η κάθε μια. Επίσης, αναδείχθηκε ο τρόπος υλοποίησης σε μια android και web εφαρμογή.

ΚΕΦΑΛΑΙΟ 6. ΒΙΒΛΙΟΘΗΚΕΣ ANDROID

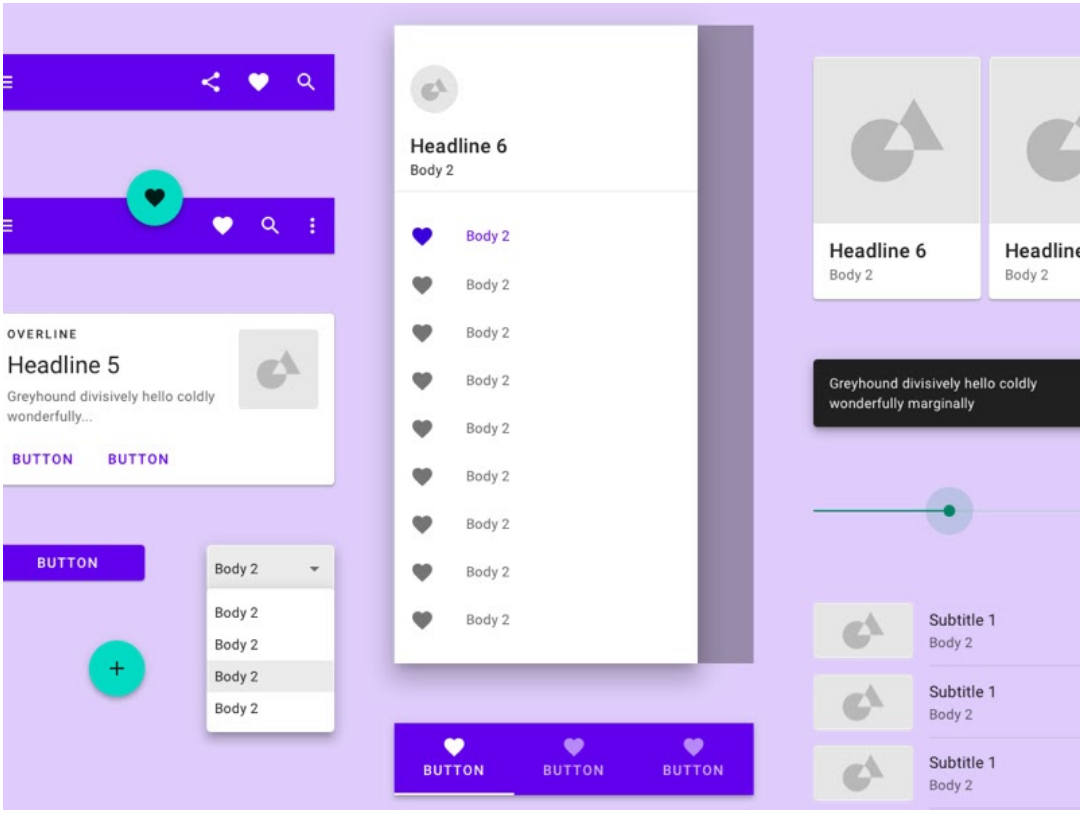
ΕΙΣΑΓΩΓΗ

Σε αυτή την ενότητα θα αναλυθούν εν συντομία οι κύριες βιβλιοθήκες που χρησιμοποιήθηκαν για ανάπτυξη της android εφαρμογής.

6.1 MATERIAL DESIGN

Το material design είναι μια ‘γλώσσα’ για τον σχεδιασμό διεπαφών που αναπτύσσεται από την Google (Σχήμα 6.1). Στοχεύει στην υποστήριξη πολλαπλών περιβαλλόντων διεπαφών. Ένας άλλος κύριος στόχος είναι η μίμηση πραγματικών αντικειμένων ώστε να βελτιστοποιηθεί η εμπειρία των χρηστών.
Το material design ξεκίνησε στα μέσα του 2014 με σκοπό τη συνέπεια σε όλα τα περιβάλλοντα.
Τα 3 κύρια χαρακτηριστικά του material design είναι:

1. Η μεταφορά του υλικού
   Μεταφορά της έννοιας του υλικού και της αφής και η προσφορά χαρακτηριστικών των οποίων οι χρήστες βρίσκουν γνώριμα. Χαρακτηριστικά που καταλαβαίνει κατευθείαν ο χρήστης, με τα οποία όταν αλληλεπιδρά έχει μια πολύ καλή ‘μαντεψιά’ για το τι θα ακολουθήσει.
   Χρήση ενός σχεδιασμού που προσαρμόζεται και είναι σταθερός ανεξάρτητα από την πλατφόρμα.
2. Έντονα και διαλεγμένα με μεγάλη προσοχή στοιχεία, τα οποία είναι ευχάριστα στο μάτι.
   Ο σχεδιασμός είναι έντονος και προσεγμένος ώστε οι χρήστες να μπορούν να εστιάζουν στα κατάλληλα σημεία και να καταλαβαίνουν γρήγορα το μήνυμα. Επίσης, γίνεται προσεγμένη χρήση των χρωμάτων και των κενών ώστε να αξιοποιείται η επιφάνεια με τον καλύτερο δυνατό τρόπο.
3. Η αλληλεπίδραση με τον χρήστη γίνεται μέσω κινήσεων. Το αποτέλεσμα της κάθε κίνησης γίνεται αντιληπτό εύκολα και είναι εμφανές. Επίσης, τα αποτελέσματα είναι απλά και ‘καθαρά’ χωρίς επιπλέον αχρείαστα χαρακτηριστικά.



Σχήμα 6.1: Διάφορα αντικείμενα σε material style

6.2 DAGGER 2

Το dagger είναι μια πλήρως στατική βιβλιοθήκη διαχείρισης εξαρτήσεων που γίνεται στον χρόνο της μεταγλώττισης. Προσφέρεται για Java, Kotlin και Android. Οι πρώτες εκδόσεις του dagger δημιουργήθηκαν από την εταιρία square. Έπειτα από ένα σημείο, ξεκίνησε να το συντηρεί η Google.
Τα κύρια προβλήματα που προσπαθεί να λύσει το dagger περιστρέφονται γύρω από την διερεύνηση προβλημάτων που σχετίζονται με την απόδοση. Συγκεκριμένα, σε αντίστοιχες βιβλιοθήκες υπάρχουν προβλήματα απόδοσης κυρίως λόγω του reflection.

Παράδειγμα χρήσης σε Android.

Για να χρησιμοποιήσουμε το dagger, πρέπει πρώτα να το ενσωματώσουμε στο project μας, μέσω του gradle (Σχήμα 6.2).

```
implementation 'com.google.dagger:dagger:2.27'
implementation 'com.google.dagger:dagger-android:2.15'
annotationProcessor 'com.google.dagger:dagger-android-processor:2.15'
annotationProcessor 'com.google.dagger:dagger-compiler:2.26'
```

Σχήμα 6.2: Αρχείο build.gradle που περιέχει το dagger 2

Έπειτα πρέπει να φτιάξουμε μια κλάση που θα πρέπει να έχει επισημανθεί με το annotation @Module (Σχήμα 6.3).

```
@Module
public class ApplicationModule {

    @Provides
    @Singleton
    MainPresenter provideMainPresenter() {
        return new MainPresenter();
    }
}
```

Σχήμα 6.3: Δημιουργία dagger 2 module

Σε αυτή την κλάση πρέπει να κάνουμε μεθόδους που να επιστρέφουν τους διάφορους τύπους αντικειμένων που χρειαζόμαστε στην εφαρμογή. Οι μέθοδοι θα πρέπει να είναι επισημασμένοι με το annotation @Provides.

Στη συνέχεια πρέπει να φτιάξουμε μια διεπαφή η οποία θα έχει επισημανθεί με το annotation @Component, το οποίο παίρνει σαν παράμετρο την προηγούμενη κλάση (Σχήμα 6.4).

```
@Singleton
@Component(modules = {ApplicationModule.class})
public interface ApplicationComponent {
    void inject(MainActivity mainActivity);
}
```

Σχήμα 6.4: Δημιουργία dagger 2 component

Σε αυτήν την κλάση πρέπει να κάνουμε μεθόδους που θα μας δίνουν τα activities/fragments της εφαρμογής.

Έπειτα θα πρέπει να φτιάξουμε μια κλάση η οποία θα επεκτείνει την *Application* (Σχήμα 6.5).

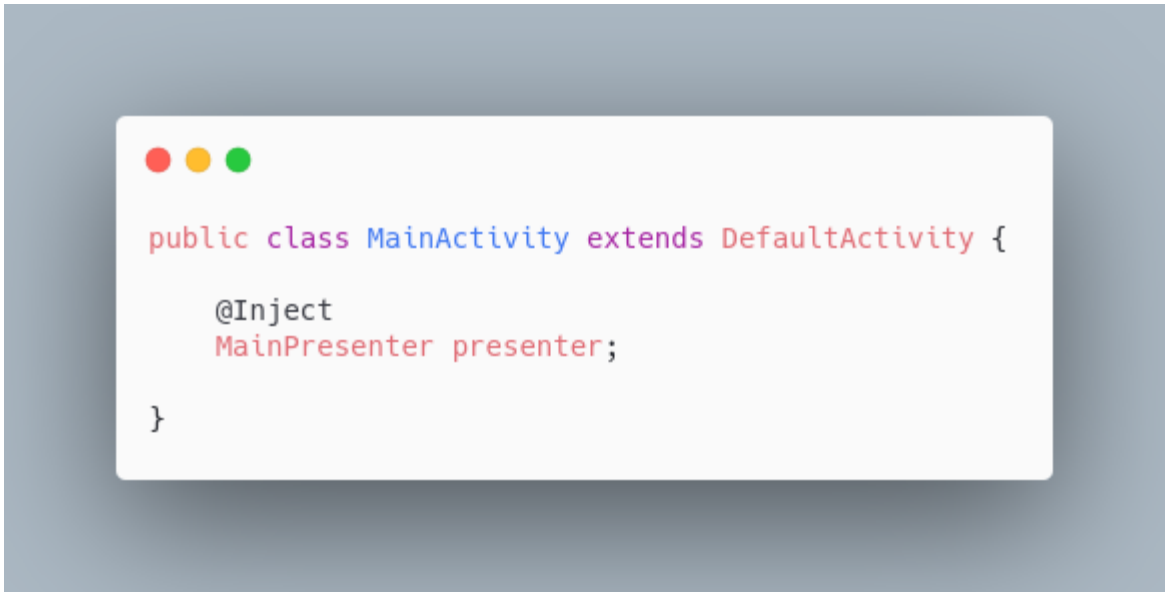
```
public class MyApplication extends Application {
    private static ApplicationComponent applicationComponent;


    @Override
    public void onCreate() {
        super.onCreate();

        applicationComponent = DaggerApplicationComponent.builder()
                .applicationModule(new ApplicationModule())
                .build();
    }

    public static ApplicationComponent applicationComponent() {
        return applicationComponent;
    }
}
```

Σχήμα 6.5: Δημιουργία dagger 2 application

Για να λειτουργήσει η διαχείριση εξαρτήσεων πρέπει επίσης να καλείται το επόμενο κομμάτι κώδικα στη μέθοδο *onCreate* του activity/fragment (Σχήμα 6.6).

Σχήμα 6.6: Ενσωμάτωση dagger 2 application με activity/fragment

Για να αποκτήσουμε πρόσβαση σε αντικείμενο που βρίσκεται στις εξαρτήσεις, θα πρέπει να το κάνουμε annotate με το annotation @Inject (Σχήμα 6.7).

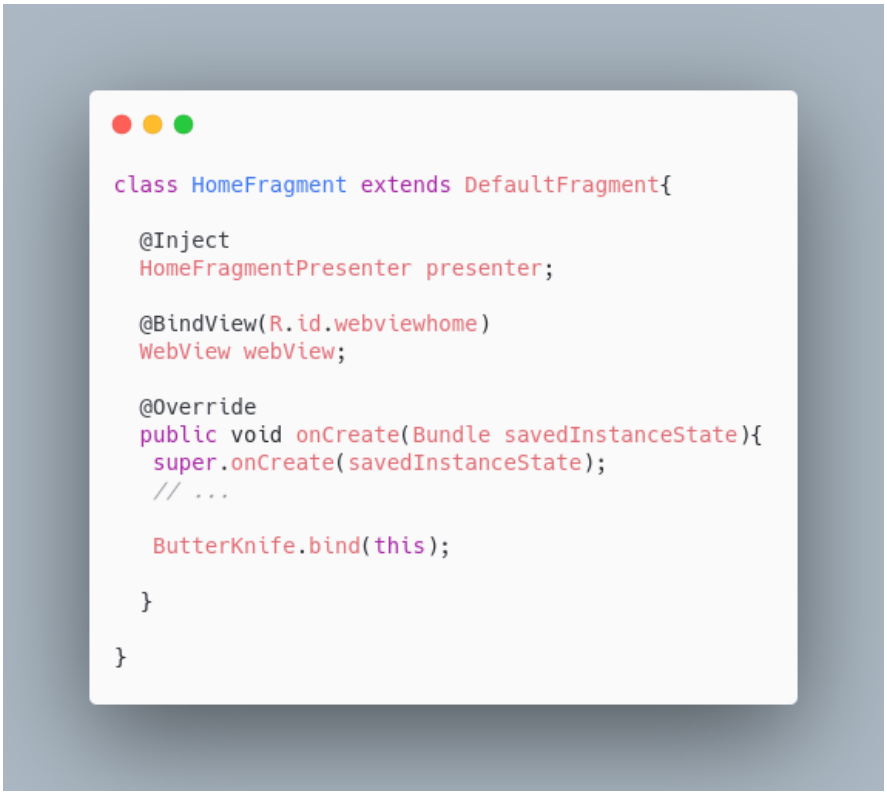


Σχήμα 6.7: Χρήση του dagger 2 για τη διαχείριση εξαρτήσεων

6.3 BUTTERKNIFE

Το butterknife είναι βιβλιοθήκη για τη σύνδεση των android views με τις αντίστοιχες κλάσεις μειώνοντας έτσι τον περιττό κώδικα. Τα κύρια στοιχεία του butterknife είναι:

- Αντικατάσταση όλων των μεθόδων *findViewById* με το annotation *@BindView*
- Η ομαδοποίηση πολλαπλών στοιχείων view
- Η εξάλειψη των ανώνυμων εσωτερικών κλάσεων για τη δημιουργία listeners για γεγονότα



```
class HomeFragment extends DefaultFragment{

  @Inject
  HomeFragmentPresenter presenter;

  @BindView(R.id.webviewhome)
  WebView webView;

  @Override
  public void onCreate(Bundle savedInstanceState){
   super.onCreate(savedInstanceState);
   // ...

   ButterKnife.bind(this);

  }

}
```

Σχήμα 6.8: Χρήση του butterknife για μεταβολή android views

Στο παραπάνω παράδειγμα (Σχήμα 6.8) φαίνεται ο τρόπος με τον οποίο γίνεται η διασύνδεση των στοιχείων του view με το αντίστοιχο activity.

6.4 FIRESTORE

Το firestore είναι μια διαδικτυακή, ευέλικτη και κλιμακούμενη noSQL βάση δεδομένων. Χρησιμοποιείται κυρίως για τον συγχρονισμό εφαρμογών client-side(όπως το android) και του server-side.

Τα κύρια στοιχεία του firestore είναι:

- Ευελιξία: Το firestore υποστηρίζει ευέλικτες και ιεραρχικές δομές δεδομένων. Τα έγγραφα μπορούν να περιέχουν σύνθετα και εμφωλευμένα αντικείμενα.
- Εκφραστικά ερωτήματα: Το firestore υποστηρίζει ερωτήματα για την ανάκτηση εγγράφων. Τα ερωτήματα αυτά μπορούν να περιέχουν πολλαπλά και σύνθετα φίλτρα για την ανάκτηση συγκεκριμένης πληροφορίας.
- Ενημερώσεις πραγματικού χρόνου: Το firestore ενημερώνει τα δεδομένα σε όλες τις συνδεδεμένες συσκευές σε πραγματικό χρόνο.
- Υποστήριξη σε περιπτώσεις που δεν υπάρχει δίκτυο: Το firestore αποθηκεύει τα δεδομένα σε κάθε συσκευή, με τέτοιο τρόπο ώστε να λειτουργεί ακόμα και αν δεν υπάρχει σύνδεση στο δίκτυο.

```
@Inject
FirebaseFirestore database;
// ...

public List<Course> retrieveCourses(){
 database.collection("courses")
   .get()
   .addOnCOmpleteListener(task->{
     List<Course> courses = new ArrayList<>();
     if (task.isSuccessful()){
      for (QueryDocumentSnapshot document: task.getResult()){
       final String title = document.getString("title");
       final String description = document.getString("description");
       courses.add(new Course(title,description));
      }
     }
     return courses;

   })


}
```

Σχήμα 6.9: Ανάκτηση δεδομένων από το firestore

Στο παραπάνω παράδειγμα (Σχήμα 6.9) φαίνεται η ανάκτηση δεδομένων από το firestore.

6.5 CRASHLYTICS

Το firebase crashlytics είναι μια βιβλιοθήκη για την αναφορά λαθών σε πραγματικό χρόνο για εφαρμογές android, iOS και Unity.

Τα κύρια στοιχεία του crashlytics είναι:

- Συμπτυγμένες αναφορές λαθών: Το crashlytics συγκεντρώνει και ομαδοποιεί τα επιμέρους λάθη που αναφέρονται από την εφαρμογή και τα μετατρέπει σε λίστες που κάνουν εμφανή την πηγή των λαθών.
- Λύσεις για τα πιο κοινά λάθη: Το crashlytics προσφέρει το Crash Insights το οποίο περιέχει βοηθητικές πληροφορίες για πληθώρα λαθών.
- Ειδοποιήσεις σε πραγματικό χρόνο: Το firestore στέλνει ειδοποιήσεις για καινούρια και αυξανόμενα προβλήματα σε πραγματικό χρόνο ώστε να ειδοποιείται ο προγραμματιστής εγκαίρως.

Σχήμα 6.10: Αρχείο build.gradle που περιλαμβάνει το firebase-crashlytics

Για να ενεργοποιήσουμε το crashlytics, αρκεί να το αναφέρουμε στο build.gradle αρχείο της εφαρμογής όπως φαίνεται στο παραπάνω παράδειγμα (Σχήμα 6.10).

6.6 GOOGLE PLAY SERVICES

Η βιβλιοθήκη google play services παρέχει διεπαφές για την επικοινωνία με πολλές σημαντικές υπηρεσίες της Google. Χρησιμοποιώντας τη βιβλιοθήκη είναι δυνατόν οι χρήστες της εφαρμογής να πάρουν πρόσβαση στις υπηρεσίες της Google χρησιμοποιώντας τα διαπιστευτήριά τους. Σημαντικό στοιχείο της βιβλιοθήκης επίσης είναι ότι έχει πολύ μικρό αντίκτυπο στη συνολική μνήμη της εφαρμογής.

```
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);

// Get the app's Drive folder
DriveResourceClient client = Drive.getDriveResourceClient(this, account);
client.getAppFolder().
        .addOnCompleteListener(this, new OnCompleteListener<DriveFolder>() {
            @Override
            public void onComplete(@NonNull Task<DriveFolder>() {
                // ...
            }
        });
```

Σχήμα 6.11: Χρήση του google play services σε activity/fragment

Στο παραπάνω παράδειγμα φαίνεται ο τρόπος με τον οποίο γίνεται η πρόσβαση σε ένα αρχείο του google drive χρησιμοποιώντας τη βιβλιοθήκη (Σχήμα 6.11).

6.7 RX JAVA (& RX ANDROID)

Η rxJava είναι μια βιβλιοθήκη για τη δημιουργία σύγχρονων και βασισμένων σε γεγονότα προγραμμάτων. Πρόκειται για μια υλοποίηση της γενικής διεπαφής ReactiveX. Τα βασικότερα στοιχεία της τελευταίας έκδοσης είναι:

- Υποστήριξη για τη Java 8
- Προσαρμοσμένο API για τη Java 8
- Διόρθωση σχεδιαστικών λαθών στο API που υπήρχαν σε προηγούμενες εκδόσεις
- API και για σύγχρονη εκτέλεση
- API για την εύκολη δημιουργία tests

Οι κύριες κλάσεις που προσφέρονται είναι:

- Flowable
- Observable
- Single
- Completable
- Maybe

Η βασική κλάση είναι η *Observable* και γύρω από αυτήν δημιουργήθηκαν οι υπόλοιπες για να διευκολύνουν συγκεκριμένες περιπτώσεις.

Κάποιοι από τους σημαντικότερους τελεστές της κλάσης *Observable* είναι:

- reduce
- cast
- concat
- first
- from
- join
- max
- never
- next

```
public void subscribeToViewModelChanges(View root,Observable<CourseFragmentState> states){

  states
    .doOnNext(state->{
      for (Map.Entry<String, String> entry : state.content().entrySet()) {
          final String key = entry.getKey();
          final String value = entry.getValue();
          view.evaluateContent(key,value);
      }

    })
    .subscribe();

}
```

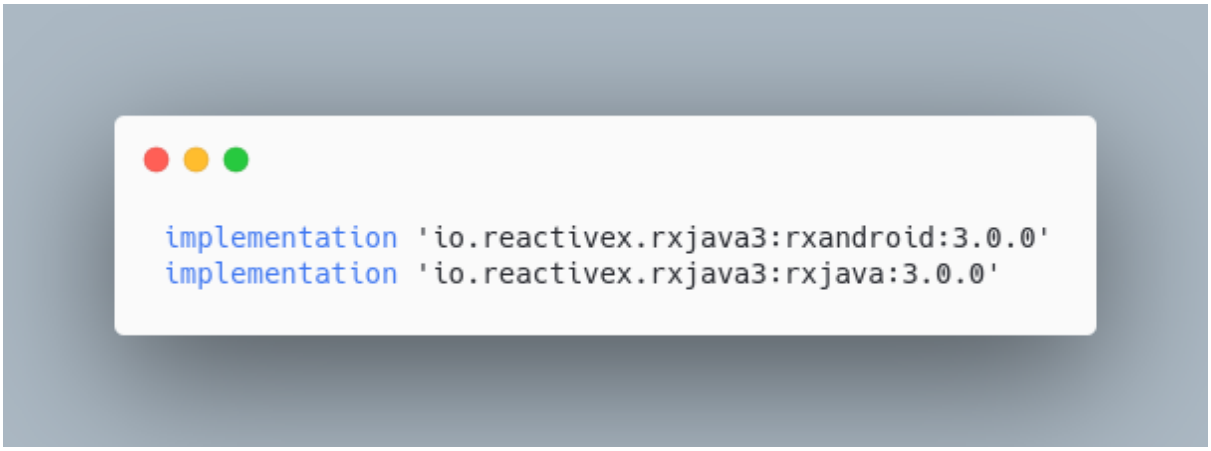
Σχήμα 6.12: Παράδειγμα χρήσης rxjava

Στο παραπάνω παράδειγμα εμφανίζεται μια λίστα από Strings χρησιμοποιώντας τη rxJava (Σχήμα 6.12).

6.7.1 RX ANDROID

Το rxAndroid είναι μια επέκταση της rxJava για την εύκολη σύνδεση με το Android.

Για την ενσωμάτωση του στο android project θα πρέπει να το βάλουμε στο build.gradle αρχείο παράλληλα με τη rxjava (Σχήμα 6.13).



Σχήμα 6.13: Αρχείο build.gradle που περιλαμβάνει το rxandroid

6.8 OK HTTP

Το okHttp είναι μια βιβλιοθήκη για την πιο αποδοτική χρήση των HTTP requests. Τα κύρια χαρακτηριστικά που κάνουν τη βιβλιοθήκη πολύ αποδοτική είναι:

- Το HTTP/2 επιτρέπει όλες τις αιτήσεις προς τον ίδιο δέκτη να μοιράζονται το ίδιο κανάλι επικοινωνίας
- Η ομαδοποίηση των συνδέσεων μειώνει την καθυστέρηση.
- Χρησιμοποιεί συμπίεση για να μειώνει το μέγεθος των download
- Αποθηκεύει τις απαντήσεις των αιτήσεων ώστε να αποφεύγει τις άχρηστες επαναλήψεις.

Εκτός από τα παραπάνω, η βιβλιοθήκη καταλαβαίνει αν το δίκτυο έχει προβλήματα ώστε να επανακάμπτει όταν υπάρχουν κοινά προβλήματα.

Σχήμα 6.14: Χρήση του okHttp για τη δημιουργία ενός GET request

Στο παραπάνω παράδειγμα (Σχήμα 6.14) φαίνεται η διαδικασία για να γίνει ένα απλό GET request.

ΕΠΙΛΟΓΟΣ

Σε αυτή την ενότητα αναλύθηκαν περιληπτικά οι βιβλιοθήκες που χρησιμοποιήθηκαν στην android εφαρμογή καθώς και παρουσιάστηκαν παραδείγματα για κάθε μια.

ΚΕΦΑΛΑΙΟ 7. ΒΙΒΛΙΟΘΗΚΕΣ WEB

ΕΙΣΑΓΩΓΗ

Σε αυτή την ενότητα θα αναλυθούν εν συντομία οι κύριες βιβλιοθήκες που χρησιμοποιήθηκαν για την ανάπτυξη της web εφαρμογής.

7.1 TYPESCRIPT

Η typescript είναι μια γλώσσα προγραμματισμού που είναι βασισμένη στη γλώσσα Javascript. Πρόκειται για μια γλώσσα ανοιχτού κώδικα. Η κύρια διαφορά με τη javascript είναι ότι η typescript χρησιμοποιεί στατικούς τύπους.

Οι στατικοί τύποι προσφέρουν τη δυνατότητα περιγραφής των αντικειμένων, με καλύτερη τεκμηρίωση. Επίσης, επιτρέπουν την typescript να ελέγχει εάν ο κώδικας της εφαρμογής είναι αποδεκτός προτού καν τρέξει.

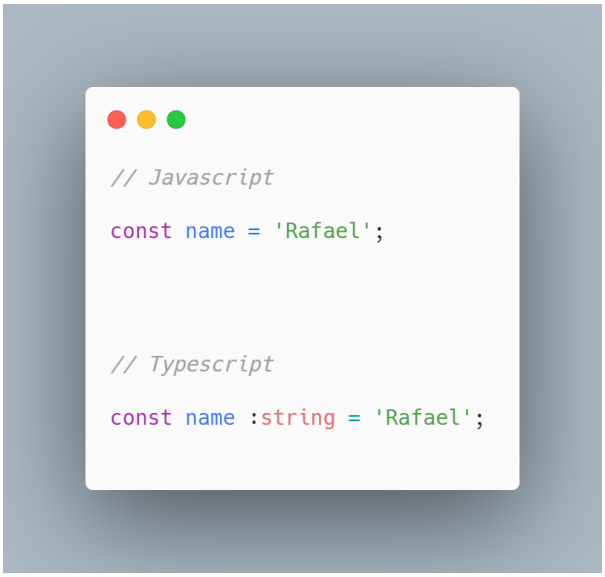
Το να εμπλουτίσεις τον κώδικα με τύπους είναι προεραιτικό. Επίσης, ο compiler αναγνωρίζει σε πολλές περιπτώσεις μόνος του τον τύπο ενός αντικειμένου.

Κάθε συντακτικά σωστός κώδικας σε javascript είναι επίσης σωστός κώδικας typescript. Αυτό σημαίνει ότι η μετατροπή ενός κώδικα εφαρμογής γραμμένη σε javascript σε typescript μπορεί να γίνει σταδιακά και με περισσότερο έλεγχο στον προγραμματιστή.

7.1.1 ΔΙΑΦΟΡΕΣ ΜΕ ΤΗ JAVASCRIPT

Παρακάτω θα μελετήσουμε μερικές διάφορες των δύο γλωσσών σε κάποια βασικά παραδείγματα κώδικα.

1. Ανάθεση τιμής σε μεταβλητή (Σχήμα 7.1)



```
// Javascript

const name = 'Rafael';



// Typescript

const name :string = 'Rafael';
```

Σχήμα 7.1: Διαφορές typescript με javascript (½)

2. Δημιουργία κλάσης αντικειμένων (Σχήμα 7.2)

```
//Javascript
class Course {
  constructor(title,code,description,semester){
   this.title=title;
   this.code=code;
   this.description=description;
   this.semester=semester;
  }
}

//Typescsript
class Course{
    private readonly _title:string;
    private readonly _code:string;
    private readonly _description:string;
    private readonly _semester:string;


    constructor(title: string, code: string, description: string, semester: string) {
        this._title = title;
        this._code = code;
        this._description = description;
        this._semester = semester;
    }


    get title(): string {
        return this._title;
    }

    get code(): string {
        return this._code;
    }

    get description(): string {
        return this._description;
    }

    get semester(): string {
        return this._semester;
    }
}
```

Σχήμα 7.2: Διαφορές typescript με javascript (2/2)

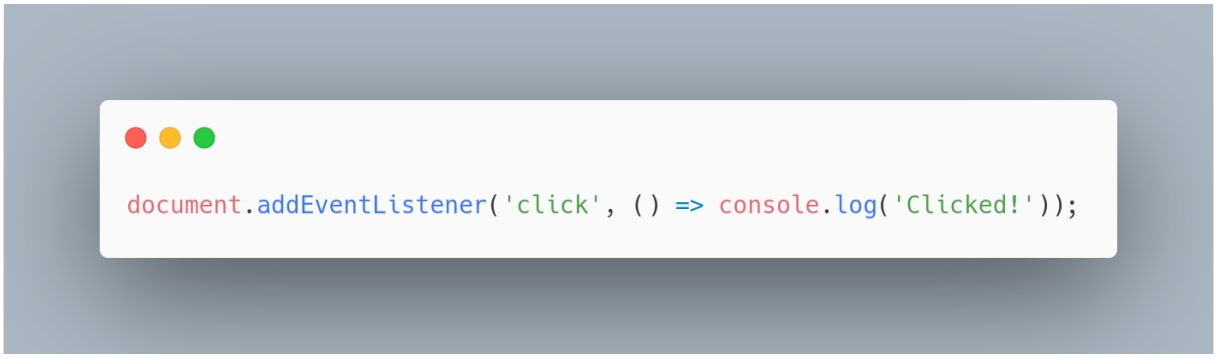
7.2 RXJS

Η rxjs είναι βιβλιοθήκη που είναι υλοποίηση της διεπαφής ReactiveX. Μοιάζει σε πάρα πολλά σημεία με τη rxJava που μελετήσαμε σε προηγούμενο κεφάλαιο. Παρόλο που και η rxjs και η rxjava είναι υλοποιήσεις της ίδιας διεπαφής, παρατηρούνται μερικές διαφορές που σχετίζονται με τα διαφορετικά περιβάλλοντα.

Οι σημαντικότερες έννοιες της rxjs είναι:

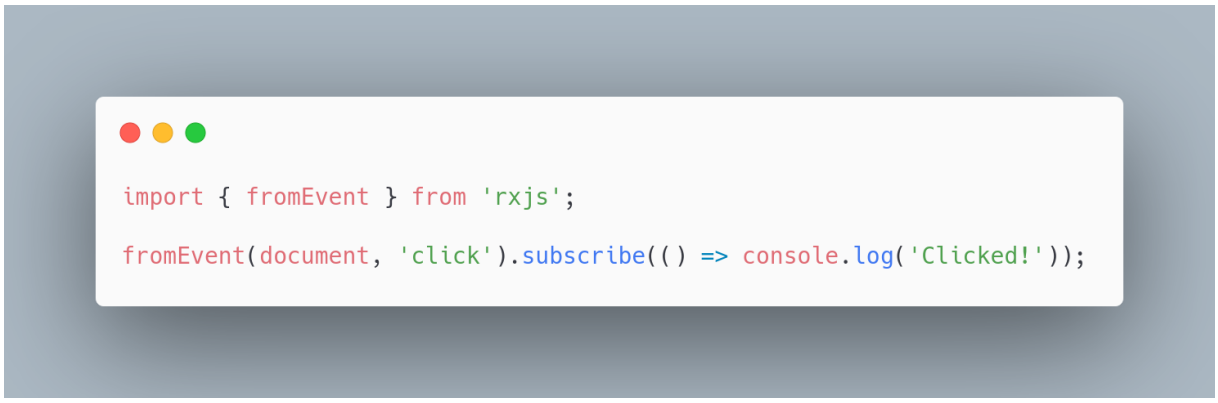
- Observable: αναπαριστά μια συλλογή από τιμές που πιθανόν να αυξηθεί στο μέλλον.
- Observer: είναι μια συλλογή από μεθόδους που πυροδοτούνται όταν φτάνουν καινούργια στοιχεία από το Observable.
- Subscription: αναπαριστά την εκτέλεση του Observable.
- Operators: Είναι μέθοδοι που βοηθούν στη μετατροπή των διάφορων δεδομένων που προέρχονται από το Observable.

Χρησιμοποιώντας τη rxjs υπάρχουν πολλά πλεονεκτήματα σε σχέση με την απλή Javascript. Παρακάτω θα δούμε κάποια παραδείγματα που κάνουν εμφανή αυτή τη διαφορά.



Σχήμα 7.3: Παράδειγμα κώδικα σε javascript (½)

Ο παραπάνω κώδικας σε Javascript μετατρέπεται στο παρακάτω χρησιμοποιώντας rxjs (Σχήμα 7.3).

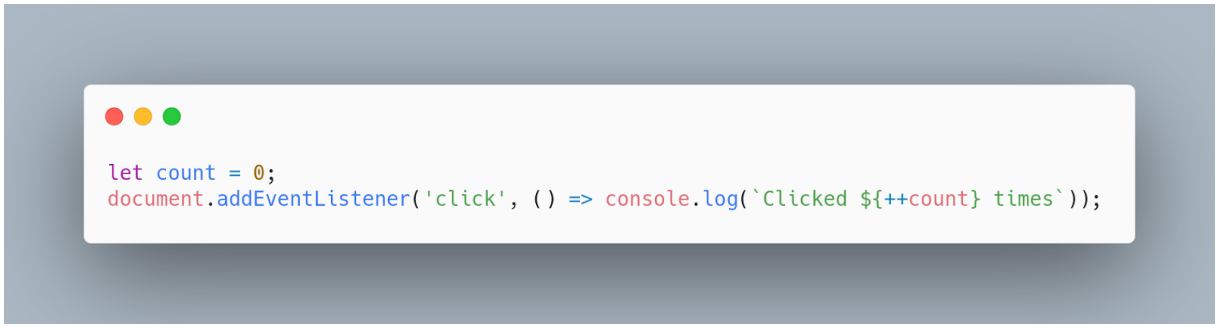


```
import { fromEvent } from 'rxjs';

fromEvent(document, 'click').subscribe(() => console.log('Clicked!'));
```

Σχήμα 7.4: Παράδειγμα κώδικα σε javascript χρησιμοποιώντας rxjs (½)

Σε αυτό το παράδειγμα (Σχήμα 7.4)οι διαφορές είναι ελάχιστες. Παρόλα αυτά όσο μεγαλώνει ένα project τόσο μεγαλύτερη είναι η χρησιμότητα της rxjs.

Στο παρακάτω παράδειγμα θα αναδειχθεί πως χρησιμοποιώντας τη rxjs ο προγραμματιστής ελαττώνει τις πιθανότητες για λάθος (Σχήμα 7.5).



```
let count = 0;
document.addEventListener('click', () => console.log(`Clicked ${++count} times`));
```

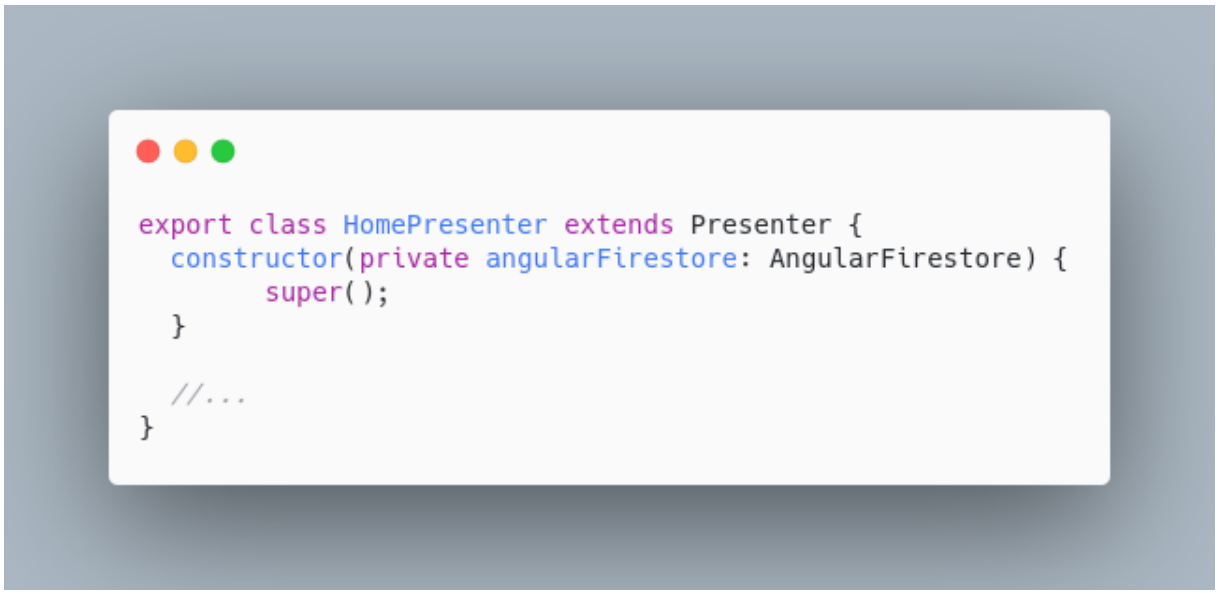
Σχήμα 7.5: Παράδειγμα κώδικα σε javascript (2/2)

Σχήμα 7.6: Παράδειγμα κώδικα σε javascript χρησιμοποιώντας rxjs (2/2)

Η κύρια διαφορά μεταξύ των δύο παραπάνω παραδειγμάτων είναι ότι στην περίπτωση που χρησιμοποιείται η rxjs, δε γίνεται έκθεση της μεταβλητής count και χρησιμοποιείται μόνο εκεί που είναι απαραίτητη (Σχήμα 7.6). Με αυτό τον τρόπο μειώνεται οι πιθανότητες λάθους από απροσεξία.

7.3 ANGULAR DEPENDENCY INJECTION

Η Angular προσφέρει ένα πολύ απλό αλλά παράλληλα δυνατό τρόπο για να γίνεται εύκολη η διαχείριση εξαρτήσεων. Εξαρτήσεις ενός αντικειμένου ονομάζονται τα διάφορα αντικείμενα που απαιτούνται για τη δημιουργία του. Τα εργαλεία διαχείρισης εξαρτήσεων βοηθάνε τον προγραμματιστή να μην επαναλαμβάνεται και να έχει μια καλή δομή στην ιεραρχία του project.
Παρακάτω έχουμε την παρακάτω κλάση που περιγράφει έναν presenter (Σχήμα 7.7). Ο presenter εξαρτάται από ένα αντικείμενο κλάσης *AngularFirestore*.



```
export class HomePresenter extends Presenter {
  constructor(private angularFirestore: AngularFirestore) {
        super();
  }

  //...
}
```

Σχήμα 7.7: Κλάση Presenter με εξαρτήσεις

Για να δημιουργήσουμε έναν presenter θα πρέπει να του προσφέρουμε όλες τις εξαρτήσεις του. Σε αυτήν την περίπτωση θα πρέπει να το προσφέρουμε ένα αντικείμενο τύπου *AngularFirestore*. Αυτή η διαδικασία θα πρέπει να γίνει στο αρχείο app.module της εφαρμογής όπου θα πρέπει να προστεθούν τα παρακάτω.

```
function homePresenterFactory(angularFirestore: AngularFirestore): HomePresenter {
    return new HomePresenter(angularFirestore);
}


// ...

@NgModule({
    providers: [
        {
            provide: HomePresenter,
            useFactory: homePresenterFactory,
            deps: [AngularFirestore],
            multi: false
        }
    ]

})
export class AppModule {
}
```

Σχήμα 7.8: Αρχείο app.module.ts που αναδεικνύει τη διαχείριση εξαρτήσεων

Στο παραπάνω παράδειγμα (Σχήμα 7.8) δημιουργείται μια μέθοδος η οποία είναι υπεύθυνη για τη δημιουργία αντικειμένων τύπου *HomePresenter*. Έτσι όταν οποιοδήποτε αντικείμενο εξαρτάται από ένα αντικείμενο τύπου *HomePresenter*, τότε η εξάρτηση αυτή ολοκληρώνεται αυτόματα.

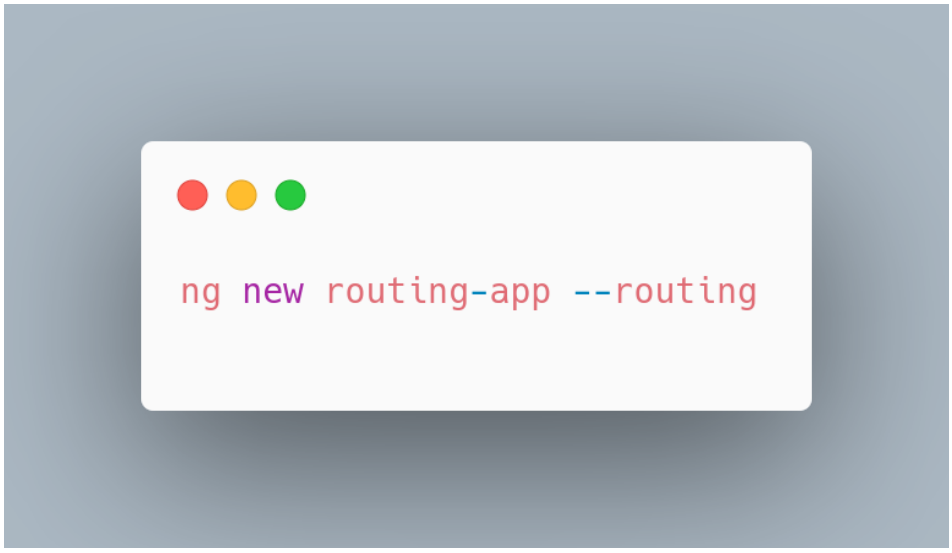
Σε περίπτωση που οι εξαρτήσεις που θέλουμε να ολοκληρώσουμε, εξαρτώνται από άλλα αντικείμενα η παραπάνω διαδικασία γίνεται αναδρομικά.

7.4 ANGULAR ROUTER

Με την Angular δημιουργούνται εφαρμογές τύπου Single Page Applications. Αυτό σημαίνει ότι η εφαρμογή αλλάζει δεδομένα στη σελίδα, ξαναγράφοντας τα καινούρια δεδομένα πάνω στα παλιά αντί να φορτώνει καινούριες σελίδες. Αυτό βοηθά κυρίως σε ταχύτερες εναλλαγές περιεχομένου.

Για να αλλάξει το περιεχόμενο μιας σελίδας τυπικά αλλάζει το τρέχων component που εμφανίζεται. Οπότε αν υπάρχει ένα component που εμφανίζει μια λίστα από μαθητές και ο χρήστης επιλέξει ένα συγκεκριμένο μαθητή, τότε φορτώνει ένα διαφορετικό component που ειδικεύεται στην εμφάνιση ενός μαθητή και παίρνει σαν παράμετρο το στοιχείο ID του συγκεκριμένου μαθητή.

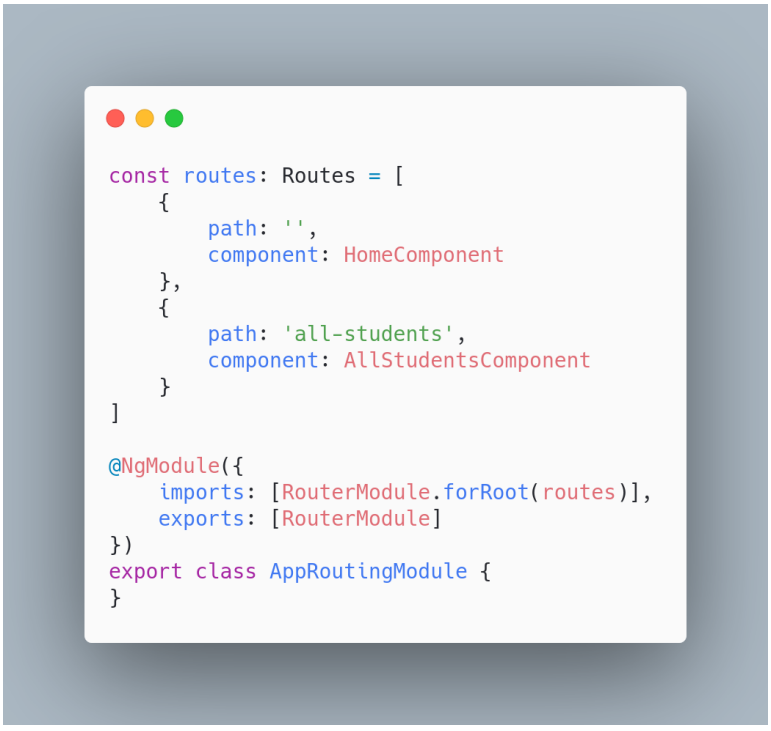
Η Angular προσφέρει εντολές για την εύκολη δημιουργία ενός router που ελέγχει πότε θα πρέπει να εμφανίζεται το κάθε component. Για να δημιουργηθεί ο router θα πρέπει πρωτίστως να εκτελεστεί η παρακάτω εντολή (Σχήμα 7.9).



Σχήμα 7.9: Εντολή εγκατάστασης πακέτου για τον έλεγχο των components

Μετά την εκτέλεση της παραπάνω εντολής δημιουργείται ένα αρχείο με όνομα app-routing.module.ts.

Η προσθήκη καινούργιων routes είναι εύκολη. Το κάθε route αποτελείται από ένα javascript object το οποίο περιέχει κατ ελάχιστον τα properties path και component. Αυτό φαίνεται στο παρακάτω παράδειγμα (Σχήμα 7.10).



```
const routes: Routes = [
    {
        path: '',
        component: HomeComponent
    },
    {
        path: 'all-students',
        component: AllStudentsComponent
    }
]

@NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
})
export class AppRoutingModule {
}
```

Σχήμα 7.10: Αρχείο app.routing.module.ts

7.5 FIREBASE

Η βιβλιοθήκη firebase είναι μια συλλογή από εργαλεία για την ενσωμάτωση υπηρεσιών της Google σε μια javascript ή nodejs εφαρμογή.

Για να μπει σε μια Angular εφαρμογή θα πρέπει να γίνουν 3 απλά βήματα.

1. Εγκατάσταση της βιβλιοθήκης μέσω του npm (Σχήμα 7.11)

Σχήμα 7.11: Εντολή εγκατάστασης βιβλιοθήκης firebase

2. Η προσθήκη στο αρχείο app.module.ts (Σχήμα 7.12)

Σχήμα 7.12:Αρχείο app.module.ts που αναδεικνύει την αρχικοποίηση της βιβλιοθήκης firebase

3. Η προσθήκη των στοιχείων στο αρχείο environment.ts (Σχήμα 7.13)

```
export const environment = {
    production: true,
    firebaseConfig: {
        apiKey: "",
        authDomain: "",
        databaseURL: "",
        projectId: "",
        storageBucket: "",
        messagingSenderId: "",
        appId: "",
        measurementId: ""
    }
};
```

Σχήμα 7.13:Αρχείο environment.ts που αναδεικνύει τα απαραίτητα στοιχεία για τη βιβλιοθήκη firebase

Έπειτα μπορεί να χρησιμοποιηθεί σε οποιοδήποτε component για την επικοινωνία με πολλαπλά google services.

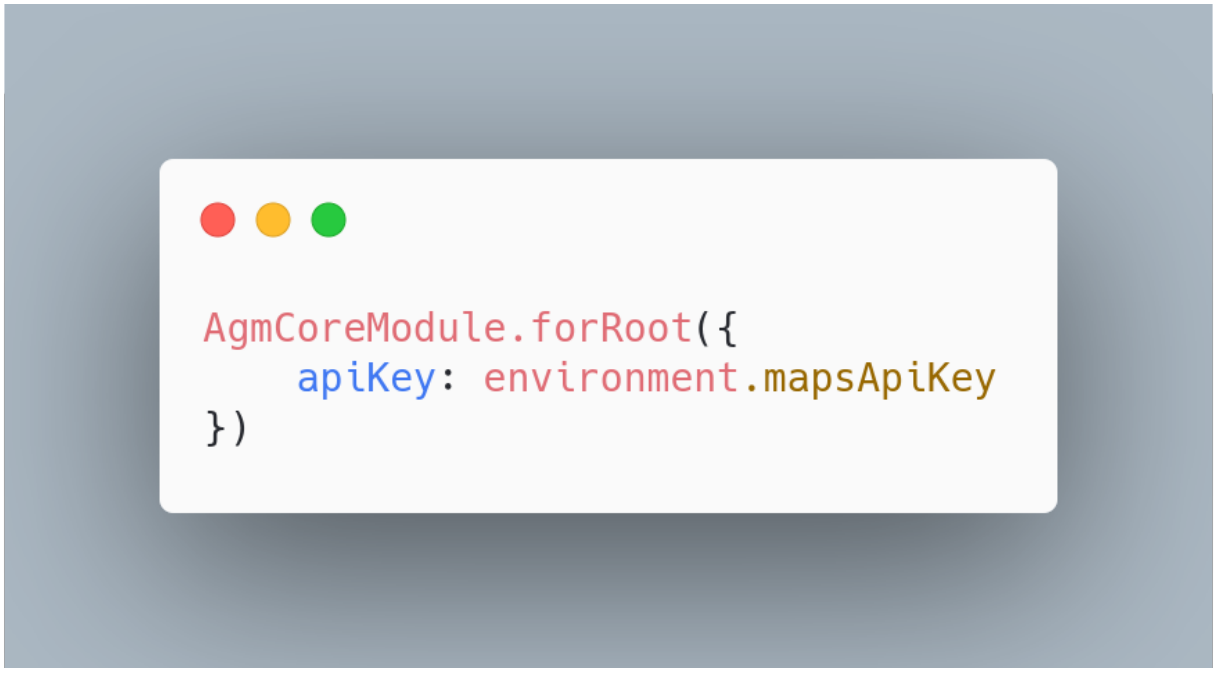
7.6 AGM

Η βιβλιοθήκη Angular Google Maps ή αλλιώς AGM είναι μια βιβλιοθήκη που κάνει εύκολη την ενσωμάτωση χαρτών της Google σε μια Angular εφαρμογή.

Για να ενσωματωθεί σε μια Angular εφαρμογή θα πρέπει πρώτα να εγκατασταθεί μέσω του npm (Σχήμα 7.14).

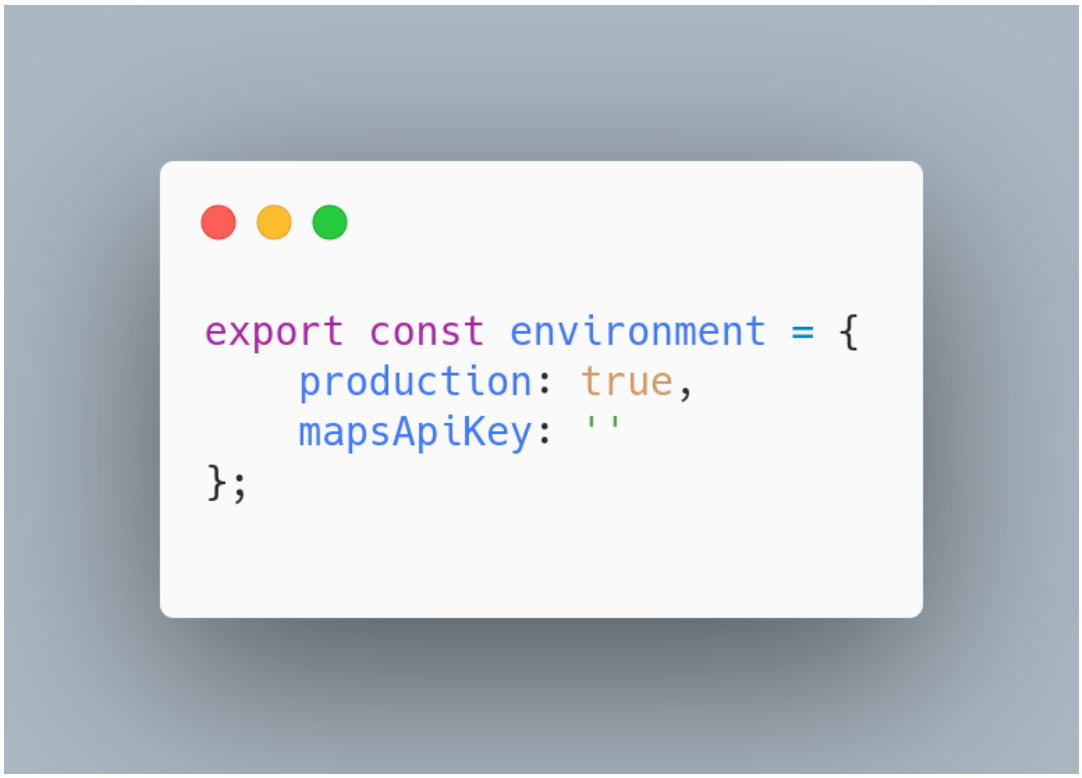
Σχήμα 7.14:Εντολή εγκατάστασης βιβλιοθήκης agm

Έπειτα θα πρέπει να προστεθεί στο app.module.ts (Σχήμα 7.15).



Σχήμα 7.15:Αρχείο app.module.ts που αναδεικνύει την αρχικοποίηση της βιβλιοθήκης agm

Στη συνέχεια θα πρέπει να ενημερωθεί το environment.ts με το κατάλληλο κλειδί για το Google Maps API (Σχήμα 7.16).



Σχήμα 7.16:Αρχείο environment.ts που αναδεικνύει τα απαραίτητα στοιχεία για τη βιβλιοθήκη agm

Τελικά, για να χρησιμοποιηθεί ένας χάρτης Google, το μόνο που πρέπει να γίνει είναι να μπει ο κατάλληλος selector σε ένα component (Σχήμα 7.17).

Σχήμα 7.17: Component που περιλαμβάνει selector της βιβλιοθήκης agm για την εμφάνιση google map

ΕΠΙΛΟΓΟΣ

Σε αυτή την ενότητα αναλύθηκαν περιληπτικά οι βιβλιοθήκες που χρησιμοποιήθηκαν στη web εφαρμογή καθώς και παρουσιάστηκαν παραδείγματα για κάθε μια. Μερικές από τις παραπάνω βιβλιοθήκες προσφέρονται από το framework Angular κάνοντας την ενσωμάτωση τους εύκολη. Κάποιες άλλες χρειάστηκαν κάποιες επιπλέον ρυθμίσεις για την αρχική ρύθμιση τους.

ΚΕΦΑΛΑΙΟ 8. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ ΒΕΛΤΙΩΣΗΣ

Σε αυτήν την ενότητα θα αναλυθούν τα συμπεράσματα που παρήχθησαν κατά τη διάρκεια της εκπόνησης αυτής της εργασίας καθώς και προτάσεις μελλοντικής βελτίωσης της. Το περιεχόμενο αυτής της εργασίας είναι μοιρασμένο σε 6 κεφάλαια τα οποία περιλαμβάνουν την έρευνα και την υλοποίηση των εφαρμογών με αναλυτικό τρόπο. Το δεύτερο και το τρίτο κεφάλαιο της εργασίας αναφέρονται στις διάφορες λειτουργίες της εφαρμογής και ο τρόπος υλοποίησης τους σαν εφαρμογή για κινητά Android. Αναλύθηκαν τα διάφορα συστατικά στοιχεία που το περιλαμβάνουν και πως αλληλεπιδρούν μεταξύ τους. Το περιεχόμενο της εφαρμογής βρίσκεται σε απομακρυσμένη βάση δεδομένων. Όταν ο χρήστης ανοίγει την εφαρμογή γίνεται HTTP request στη βάση δεδομένων για την ανάκτηση των δεδομένων. Έπειτα αυτά τα δεδομένα εμφανίζονται στον χρήστη μέσω των τεχνολογιών του Android όπως τα activities και τα fragments. Για να μην υπάρχει μεγάλη εξάρτηση μεταξύ των δεδομένων και των τεχνολογιών του Android χρησιμοποιήθηκε η τεχνική MVP η οποία αναλύθηκε στο κεφάλαιο 5. Χρησιμοποιώντας αυτή την τεχνική ελαχιστοποιήθηκαν οι εξαρτήσεις από τα διάφορα συστατικά στοιχεία. Εκτός από την εφαρμογή Android, δημιουργήθηκε και μια εφαρμογή Web χρησιμοποιώντας το framework Angular. Η εφαρμογή Web περιέχει ακριβώς το ίδιο περιεχόμενο με την εφαρμογή Android και διαφέρει ελάχιστα στην εμφάνιση των πληροφοριών. Για να δημιουργηθούν οι εφαρμογές Android και Web χρησιμοποιήθηκε πλήθος από βιβλιοθήκες σε κάθε περίπτωση ώστε οι εφαρμογές να χρησιμοποιούν τεχνικές που είναι δοκιμασμένες και αποδίδουν πολύ καλά.

Ως προτάσεις βελτίωσης θα πρέπει να αναφερθεί ότι η εφαρμογή εξαρτάται από την ύπαρξη σύνδεσης διαδικτύου. Μελλοντικά θα μπορούσε να αποθηκεύεται το περιεχόμενο και τοπικά στη συσκευή ώστε να λειτουργεί η εφαρμογή χωρίς σύνδεση διαδικτύου. Βέβαια, η συγκεκριμένη πρόταση εφαρμόζεται μόνο στην εφαρμογή Android. Επίσης, σε περίπτωση αναβάθμισης των πόρων του τμήματος η εφαρμογή θα έπρεπε να τους χρησιμοποιεί για την αποτελεσματική ενημέρωση πεπαλαιωμένων δεδομένων χωρίς την ανάμειξη του διαχειριστή του περιεχομένου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] L. Welling and L. Thomson, Ανάπτυξη Web Εφαρμογών με PHP και MySQL, 4th ed. M. Γκιούρδας 2020.

[2] J. KUROSE and K. ROSS, Computer Networking A Top-Down Approach, 6 th ed. Pearson.

Internet Site

[3] “Build Script Basics”, gradle.org. [Online]. Available
https://docs.gradle.org/current/userguide/tutorial_using_tasks.html

[4] “Build Script Basics” gradle.org. [Online]. Available
https://docs.gradle.org/current/userguide/userguide.html

[5] “Getting Started With Gradle” petrikainulainen.net. [Online]. Available
https://www.petrikainulainen.net/getting-started-with-gradle

[6] “App Manifest Overview” developer.android.com. [Online]. Available
https://developer.android.com/guide/topics/manifest/manifest-intro

[7] “AndroidManifest.xml file in android” javatpoint.com. [Online]. Available
https://www.javatpoint.com/AndroidManifest-xml-file-in-android

[8] “Activity” developer.android.com. [Online]. Available
https://developer.android.com/reference/android/app/Activity

[9] “Android Activity” jenkov.com. [Online]. Available
http://tutorials.jenkov.com/android/activity.html

[10] “Android - Activities” tutorialspoint.com. [Online]. Available
https://www.tutorialspoint.com/android/android_acitivities.htm

[11] “Creating and Using Fragments” codepath.com. [Online]. Available
https://guides.codepath.com/android/creating-and-using-fragments

[12] “Android – Read file from Assets” javacodegeeks.com. [Online]. Available
https://www.javacodegeeks.com/2012/02/android-read-file-from-assets.html

[13] “Resources and Assets” linuxtopia.org. [Online]. Available https://www.linuxtopia.org/online_books/android/devguide/guide/topics/resources/index.html

[14] “Android - WebView” tutorialspoint.com. [Online]. Available https://www.tutorialspoint.com/android/android_webview_layout.htm

[15] “Maps SDK for Android” developers.google.com. [Online]. Available https://developers.google.com/maps/documentation/android-sdk/overview

[16] “Understanding navigation”material.io. [Online]. Available https://material.io/design/navigation/understanding-navigation.html

[17] “Jetpack Navigation” codelabs.developers.google.com. [Online]. Available https://codelabs.developers.google.com/codelabs/android-navigation/

[18] “What is Proguard”perfomatix.com. [Online]. Available https://www.perfomatix.com/proguard-android/

[19] “MVC, MVP and MVVM Design Pattern” medium.com. [Online]. Available https://medium.com/@ankit.sinhal/mvc-mvp-and-mvvm-design-pattern-6e169567bbad

[20] “What is Material Design?” interaction-design.org. [Online]. Available https://www.interaction-design.org/literature/topics/material-design

[21] “Dagger” dagger.dev. [Online]. Available https://dagger.dev/

[22] “Butter Knife” jakewharton.github.io. [Online]. Available http://jakewharton.github.io/butterknife/

[23] “Add Firebase to your Android project” firebase.google.com. [Online]. Available https://firebase.google.com/docs/android/setup

[24] “Firebase crashlytics” firebase.google.com. [Online]. Available https://firebase.google.com/products/crashlytics

[25] “Set Up Google Play Services” developers.google.com. [Online]. Available https://developers.google.com/android/guides/setup

[26] “ReactiveX” reactivex.io. [Online]. Available http://reactivex.io/

[27] “AboutRxJava – Reactive Extensions for the JVM” github.com. [Online]. Available https://github.com/ReactiveX/RxJava

[28] “OkHttp” square.github.io. [Online]. Available https://square.github.io/okhttp/

[29] “Angular Material 10/9 Tutorial: Build Navigation UI with Toolbar and Side Navigation Menu” techiediaries.com. [Online]. Available https://www.techiediaries.com/angular-material-navigation-toolbar-sidenav/

[30] “Angular” angular.io. [Online]. Available https://angular.io/

[31] “In-app navigation: routing to views” angular.io. [Online]. Available https://angular.io/guide/router

[32] “Angular CLI” cli.angular.io. [Online]. Available https://cli.angular.io/

[33] “Communicating with backend services using HTTP” angular.io. [Online]. Available https://angular.io/guide/http

[34] “Angular - HTTP POST Request Examples” jasonwatmore.com. [Online]. Available https://jasonwatmore.com/post/2019/11/21/angular-http-post-request-examples

[35] “Components - Angular Material” material.angular.io. [Online]. Available https://material.angular.io/components/categories

[36] “Introduction to components and templates” angular.io. [Online]. Available https://angular.io/guide/architecture-components

[37] “Typescript: Typed Javascript at any scale” typescriptlang.org. [Online]. Available https://www.typescriptlang.org/

[38] “Typescript vs JavaScript: What's the Difference?” guru99.com. [Online]. Available https://www.guru99.com/typescript-vs-javascript.html

[39] “JavaScript VS TypeScript: Which is better? (2020 Updated)” infinijith.medium.com. [Online]. Available https://infinijith.medium.com/javascript-vs-typescript-which-is-better-2020-updated-871866a3c68c

[40] “TypeScript vs. JavaScript: Understand the differences” infoworld.com. [Online]. Available https://www.infoworld.com/article/3526447/typescript-vs-javascript-understand-the-differences.html

[41] “RxJS Introduction” rxjs-dev.firebaseapp.com. [Online]. Available https://rxjs-dev.firebaseapp.com/

[42] “Dependency injection in Angular” angular.io. [Online]. Available https://angular.io/guide/dependency-injection

[43] “Dependency injection in action” angular.io. [Online]. Available https://angular.io/guide/dependency-injection-in-action

[44] “Firebase” firebase.google.com. [Online]. Available https://firebase.google.com/

[45] “Firebase Javascript SDK” github.com. [Online]. Available https://github.com/firebase/firebase-js-sdk

[46] “Angular 2+ Google Maps Components” github.com. [Online]. Available https://github.com/SebastianM/angular-google-maps

[47] “Angular Google Maps Components” angular-maps.com. [Online]. Available https://angular-maps.com/