ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Android εφαρμογής για την παρουσίαση δεδομένων που αφορούν την εισαγωγή νέων φοιτητών στα ΑΕΙ μέσω των πανελλαδικών εξετάσεων»

Του φοιτητή
Τζελαλή Παναγιώτη
Αρ. Μητρώου: 154546

Επιβλέπων
Στέφανος Ουγιάρογλου
Προσωπικό Ε.ΔΙ.Π.

15 Ιουνίου 2021

Ανάπτυξη Android εφαρμογής για την παρουσίαση δεδομένων που αφορούν την εισαγωγή νέων φοιτητών στα ΑΕΙ μέσω των πανελλαδικών εξετάσεων

21165

Φοιτητής: Τζελαλής Παναγιώτης

Εισηγητής: Στέφανος Ουγιάρογλου

Ημερομηνία ανάληψης Δ.Ε. 05-03-2021

Ημερομηνία περάτωσης Δ.Ε. 15-06-2021

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.

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας είναι η υλοποίηση μίας Android εφαρμογής που θα είναι βοήθημα για τους υποψηφίους των πανελλαδικών εξετάσεων αλλά και γενικότερα για άτομα που ενδιαφέρονται πιο έμμεσα όπως καθηγητές και γονείς μαθητών. Η εφαρμογή θα προσφέρει με συγκροτημένο τρόπο πληροφορίες για τα ενεργά και μη ενεργά τμήματα, τις βάσεις εισαγωγής προηγούμενων ετών, καθώς και ένα σύνολο στατιστικών για τα έτη αυτά. Για την συλλογή των δεδομένων αυτών θα χρησιμοποιηθεί ένα ήδη υπάρχων RESTful API ελεύθερης χρήστης το οποίο συγκεντρώνει τις πληροφορίες από τις επίσημες ανακοινώσεις του Υπουργείου Παιδείας και Θρησκευμάτων και τις προσφέρει με έναν τρόπο οργανωμένο. Θα περιέχει, επίσης, δύο ακόμα κύρια χαρακτηριστικά. Το ένα αφορά την δυνατότητα που προσφέρεται στον χρήστη να αναζητήσει και να προβάλει με γρήγορο τρόπο θέματα εξετάσεων προηγούμενων ετών και το δεύτερο αφορά την ύπαρξη ενός υπολογιστή μορίων για τον ευκολότερο και γρηγορότερο υπολογισμό των τελικών μορίων του υποψηφίου. Ο υπολογισμός αυτός θα πραγματοποιείται με την εισαγωγή των πιθανών βαθμολογιών του για το κάθε μάθημα. Επιπρόσθετα, στα πλαίσια της δημιουργίας μίας ολοκληρωμένης και σύγχρονης εφαρμογής θα δημιουργηθεί ένα δεύτερο βοηθητικό API το οποίο θα προσφέρει πληροφορίες για να κρατάει την εφαρμογή ενημερωμένη με την επικαιρότητα, τόσο στα παλιά θέματα εξετάσεων, όσο και σε πιθανές αλλαγές στον υπολογισμό των τελικών μορίων. Με τον τρόπο αυτόν θα αποφευχθεί η ανάγκη για ενημέρωση της εφαρμογής σε κάθε τέτοιου είδους διαφοροποιήσεις.

«Development of Android application for the presentation of data related to the admission of new students to universities through the national exams»

«Tzelalis Panagiotis»

Abstract

The goal of this thesis is the development of an Android application that will provide help to the candidates of the national exams but also in general for people who are more indirectly interested, like teachers or parents of the students. The application will offer, in a structured way, information about the active and inactive departments, the past year's last succeeded candidate's grade and also a set of statistics for these years. An already existing public RESTful API will be used for the collection of this data, this API gathers information from the official announcements of the Greek Ministry of Education and Religions and provides them in an organized way. The application will also contain two more key features. The first one is the ability of the user to quickly search and view exam questions from previous years. The second feature will be a calculator that offers an easier and quicker calculation of the final grade of the candidate. This final grade will be calculated by entering the expected grades for each course. In addition, as a part of the development of a complete and modern application, a second, helping, API will be created, which will provide information to keep the application up to date, both in the past exam topics and in possible changes in the calculation of the final grade. In this way, the need for update of the application for any of the above matters will be avoided.

Περιεχόμενα

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος

Κεφάλαιο 1ο Εισαγωγή

1.1 Εισαγωγή

Στο κεφάλαιο αυτό, θα αναπτυχθούν πληροφορίες σχετικά με τις Πανελλαδικές Εξετάσεις και τον τρόπο που το Υπουργείου Παιδείας και Θρησκευμάτων (ΥΠΑΙΘ) δημοσιεύει τα αποτελέσματα και τα ανάλογα στατιστικά κάθε χρόνο. Ακόμα, θα αναλυθεί ποιο ήταν το κίνητρο που οδήγησε στην εκπόνηση αυτής της Δ.Ε. και τη συνεισφορά της. Τέλος, θα γίνει αναφορά στην οργάνωση, το χρονικό της και την εκτέλεση της.

1.2 Σύστημα Εξετάσεων

Η εισαγωγής στην τριτοβάθμια εκπαίδευση επιτυγχάνεται κατά κύριο λόγο μέσω των πανελλαδικών εξετάσεων που διοργανώνει κάθε χρόνο το ΥΠΑΙΘ. Η διαδικασία των εξετάσεων αυτών εξαρτάται από την παρούσα νομοθεσία. Επομένως, η ανάλυση που πραγματοποιείται παρακάτω αφορά τη χρονική στιγμή που έγινε η συγγραφή της Δ.Ε. Είναι πιθανών μελλοντικά να υπάρξουν μικρές ή μεγάλες διαφοροποιήσεις, όπως έχει συμβεί και στο παρελθόν.

Οι υποψήφιοι εξετάζονται σε τέσσερα βασικά μαθήματα, ενώ για την εισαγωγή σε συγκεκριμένα τριτοβάθμια τμήματα απαιτείται και η εξέταση σε κάποιο επιπρόσθετο μάθημα, όπως ξένες γλώσσες, σχέδιο, κ.α. Τα εξεταζόμενα μαθήματα ορίζονται από την ομάδα προσανατολισμού που έχουν επιλέξει. Οι μαθητές επιλέγουν ομάδα προσανατολισμού στη Β' Λυκείου αλλά έχουν τη δυνατότητα να την αλλάξουν και αργότερα. Με εξαίρεση το μάθημα της «Νεοελληνικής Γλώσσας και Λογοτεχνίας» το οποίο είναι κοινό σε όλες τις ομάδες τα υπόλοιπα μαθήματα ορίζονται από την επιλεγόμενη ομάδα. Τα μαθήματα της κάθε ομάδας έχουν διαφορετικό συντελεστή βαρύτητας, οπότε και η επίδοση των υποψηφίων στο καθένα επηρεάζει είτε περισσότερο είτε λιγότερο την τελική βαθμολογία του υποψήφιου, η οποία υπολογίζεται πολλαπλασιάζοντας τον βαθμού που έγραψε ο υποψήφιος με τον συντελεστή του αντίστοιχου μαθήματος. Με την δημοσίευση των επιδόσεων των υποψήφιων από το ΥΠΑΙΘ, οι υποψήφιοι συμπληρώνουν το μηχανογραφικό τους δελτίο στο οποίο ορίζουν με σειρά προτίμησης τα τμήματα των τριτοβάθμιων ιδρυμάτων που επιθυμούν να εισαχθούν. Έπειτα, με βάση τις προτιμήσεις και τις βαθμολογίες των υποψηφίων, το ΥΠΑΙΘ ανακοινώνει δημόσια τις βάσεις εισαγωγής για όλες τις κατηγορίες για το συγκεκριμένο έτος εξέτασης, καθώς και διάφορα άλλα επιπλέον στατιστικά, όπως αυτά των επιδόσεων και των προτιμήσεων των υποψήφιων.

Ωστόσο, το σύστημα των πανελληνίων εξετάσεων είναι αρκετά σύνθετο. Εκτός από τον διαχωρισμό των Γενικών και των Επαγγελματικών Λυκείων, υπάρχει και ο διαχωρισμός στις κατηγορίες 90% και 10%. Ο τύπος 90% αφορά τους υποψηφίους οι οποίοι συμμετέχουν στις εξετάσεις το έτος της διενέργησής τους. Αντίθετα, ο τύπος 10% αναφέρεται σε εκείνους που έχουν δώσει τις εξετάσεις αυτές μέσα στα προηγούμενα δύο έτη και επιθυμούν να εισαχθούν σε κάποιο ίδρυμα με βάση τα αποτελέσματα της χρονιάς που εξετάστηκαν. Προφανώς κάθε υποψήφιος ανταγωνίζεται μόνο όσους έκαναν την ανάλογη αίτηση. Παράλληλα, υπάρχουν και οι ειδικές κατηγορίες οι οποίες έχουν παραλλαγές στο πέρασμα του χρόνου. Μερικά παραδείγματα αυτών είναι οι βάσεις εισαγωγής για τους υποψήφιους που ανήκουν σε τρίτεκνη ή πολύτεκνη οικογένεια, για τους μουσουλμάνους και για τους Έλληνες του εξωτερικού. Ακόμη, για συγκεκριμένες σχολές, όπως οι Αστυνομικές, υπάρχουν διαφορετικές κατηγορίες για τους πολίτες και για τους αστυνομικούς. Παρόλα αυτά, είναι σημαντικό να τονιστεί ότι ο μεγαλύτερος όγκος υποψηφίων στις πανελλήνιες προέρχεται από τα Γενικά Λύκεια και την κατηγορία του 90%.

1.3 Κίνητρο

Καθώς ο τρόπος δημοσιοποίησης των αποτελεσμάτων των πανελλαδικών εξετάσεων είναι αρκετά σύνθετος δημιουργήθηκε η ανάγκη μιας πιο απλουστευμένης και εύχρηστης μορφής παρουσίασης η οποία θα είναι προσβάσιμη από το ευρύ κοινό.

Το ΥΠΑΙΘ δημοσιεύει κάθε χρόνο στον χώρο των ανακοινώσεών του μία λίστα από υπολογιστικά φύλλα. Το κάθε υπολογιστικό φύλλο αντιστοιχεί στους διαφορετικούς τύπους και τις διαφορετικές κατηγορίες. Στα υπολογιστικά φύλλα σχετικά με τις βάσεις εισαγωγής περιλαμβάνονται ο κωδικός και ο τίτλος της κάθε σχολής, το πανεπιστήμιο στο οποίο ανήκει, το επιστημονικό πεδίο, οι βάσεις εισαγωγής του πρώτου και του τελευταίου και το είδος της θέσης που αντιστοιχούν αυτές οι βάσεις. Στα αρχεία που αφορούν τα στατιστικά προτιμήσεων περιλαμβάνονται επίσης τα στοιχεία των τμημάτων και ο αριθμός των υποψηφίων που επέλεξαν τη σχολή από την πρώτη έως και την έκτη επιλογή για τους επιτυχόντες και από την πρώτη έως και την τρίτη για το σύνολο όλων των υποψηφίων. Αρχικά, το πρώτο πρόβλημα που εντοπίζεται είναι η μη ανάρτηση αυτών των αρχείων σε έναν συγκεκριμένο και εύκολα προσπελάσιμο χώρο, στον οποίο να βρίσκονται συγκεντρωτικά και οργανωμένα όλα τα αρχεία από όλες τις χρονιές. Αυτό καθιστά δύσκολο, για τον υποψήφιο ή για τον όποιο ενδιαφερόμενο, την συλλογή αυτών των αρχείων. Επιπλέον, τα ίδια τα αρχεία και η δομή τους καθιστούν δύσκολη την αναζήτηση και την ανάλυση των δεδομένων καθώς ο ενδιαφερόμενος θα πρέπει μελετήσει μεγάλο πλήθος διαφορετικών αρχείων για τις κατηγορίες που τον ενδιαφέρει, με σκοπό να βγάλει τα συμπεράσματα που τον ενδιαφέρουν.

Εξαιτίας των συγκεκριμένων προβλημάτων έχει υλοποιηθεί ένα RESTful API, στα πλαίσια μίας άλλης πτυχιακής εργασίας, το οποίο συγκεντρώνει αυτές τις πληροφορίες και τις προσφέρει με έναν συγκεντρωτικό και οργανωμένο τρόπο.

Στόχος αυτής της Δ.Ε. είναι η ανάπτυξη μίας ολοκληρωμένης Android εφαρμογής η οποία θα εκμεταλλεύεται το RESTful API που υλοποιήθηκε και θα παρουσιάζει με έναν εύχρηστο και φιλικό προς τον χρήστη τρόπο όλα τα προαναφερόμενα δεδομένα που παρουσιάστηκαν στην Android συσκευή του.

Επιπρόσθετα, μαζί με τις πληροφορίες που θα προσφέρονται, σχετικά με τις βάσεις και τις σχολές στις οποίες αναφέρονται, ο χρήστης θα μπορεί να χρησιμοποιήσει και ακόμη δύο πολύ σημαντικά βοηθήματα.

1.4 Συνεισφορά

Η εφαρμογή θέτει ως στόχο την υλοποίηση ενός ολοκληρωμένο και εύχρηστου βοηθήματος για τους υποψήφιους των πανελληνίων στην κινητή τους συσκευή. Μαθητές, γονείς, καθηγητές και γενικότερα άτομα που ενδιαφέρονται άμεσα ή έμμεσα για την διαδικασία των πανελληνίων, θα έχουν τη δυνατότητα για εύκολη πρόσβαση σε σημαντικά δεδομένα.

Αξίζει να σημειωθεί ότι λόγω της φύσης και του θέματος της εφαρμογής το μεγαλύτερο ποσοστό των χρηστών εμμένετε να αποτελείται από υποψηφίους πανελλαδικών εξετάσεων. Αρχικά, υπάρχει η δυνατότητα για εύκολη και με φιλικό τρόπο προς τον χρήστη αναζήτηση, ανάλυση και παρατήρηση των διάφορων τμημάτων ανώτατης εκπαίδευσης, των παλιών βάσεών τους και των διάφορων επιπλέων στατιστικών. Εκτός από το βασικό αυτό κομμάτι της εφαρμογής υπάρχουν ακόμη δύο βασικά εργαλεία που θα προσφέρονται. Το πρώτο εκ των δύο αφορά τη συγκρότηση και την παρουσίαση των παλαιών θεμάτων. Με το τρόπο αυτόν, ο χρήστης είναι σε θέση να μπορεί να βρει γρήγορα μέσω του κινητού του οποιοδήποτε παλαιό θέμα επιθυμεί. Το δεύτερο εργαλείο που εμπεριέχεται στην εφαρμογή είναι

μια αριθμομηχανή για τον υπολογισμό των μορίων. Συνεπώς, ο χρήστης μπορεί να υπολογίσει τα τελικά του μόρια πιο εύκολα και γρήγορα, δίχως να χρειάζεται να γνωρίζει τον συντελεστή του κάθε μαθήματος, εισάγοντας μόνο τους βαθμούς των μαθημάτων. Αξίζει να υπωθεί ότι η εφαρμογή διαθέτει μια διαδικασία βελτιστοποίησης ανάλογα με τις προτιμήσεις του εκάστοτε χρήστη. Κάποιες οθόνες καθώς και ο τρόπος εμφάνισής τους προσαρμόζονται ανάλογα με το σχολείο την κατηγορία και άλλα στοιχεία που τον ενδιαφέρουν περισσότερο.

1.5 Χρονικό

Αρχικά, διατάχθηκε μία έρευνα σχετικά με τις ανάγκες και τις προτιμήσεις του κοινού που θα χρησιμοποιεί την εφαρμογή. Στην ανάγκη για μια όσο το δυνατόν πληρέστερη εφαρμογή κρίθηκε απαραίτητό εκτός από την προβολή ιδρυμάτων, βάσεων και διάφορων στατιστικών να προστεθούν και άλλοι δύο βασικοί τομείς. Εκείνος του υπολογιστή μορίων, ο οποίος θα βοηθάει στον εύκολο υπολογισμό μορίων και του τομέα παλαιών θεμάτων, έτσι ώστε ο χρήστης να έχει την δυνατότητα να αναζητήσει και να βρει με εύκολο τρόπο διάφορα παλιά θέματα ανά μάθημα.

Έπειτα, πραγματοποιήθηκε μελέτη σχετικά με τις δυνατότητες και τις πληροφορίες του ήδη υπάρχων API που προσφέρει τις βάσεις. Οι πληροφορίες που παρέχει κρίθηκαν πληρέστατες όσον αφορά τα ιδρύματα, τις βάσεις και τα προαναφερόμενα στατιστικά. Παρόλα αυτά, αποφασίστηκε να κατασκευαστεί ένα επιπλέον API το οποίο θα προσφέρει τα κατάλληλα δεδομένα για την σωστή λειτουργία του υπολογιστή μορίων και των παλαιών θεμάτων. Έτσι, η εφαρμογή θα μπορεί να ενημερώνεται για τυχόν αλλαγές στους συντελεστές βαρύτητας και γενικότερα για τη δομή των μαθημάτων. Επιπρόσθετα, θα υπάρχει η δυνατότητα για προσθήκη περισσότερων παλαιών θεμάτων.

Όταν ολοκληρώθηκε η μελέτη σχετικά με τους τρεις τομείς της εφαρμογής (βάσεις, υπολογιστής μορίων, παλιά θέματα) και τον τρόπο που θα παρέχονται αυτές οι πληροφορίες ξεκίνησε η μελέτη για την ίδια την εφαρμογή. Αποφασίστηκαν, τεχνολογίες, αρχιτεκτονικές και διάφορες άλλες λεπτομέρειες που σχετίζονται με την ανάπτυξη του τόσο του Android App όσο και του βοηθητικού API των οποίων η υλοποίηση ξεκίνησε μετά το πέρας των προηγούμενων διαδικασιών.

1.6 Οργάνωση της Διπλωματικής Εργασίας

Στο συγκεκριμένο κεφάλαιο της Δ.Ε. έγινε αναφορά στο σύστημα εξετάσεων των Πανελλαδικών Εξετάσεων, στη μορφή και τη μέθοδο που δημοσιεύονται τα δεδομένα από το ΥΠΑΙΘ, καθώς και στις ανάγκες που οδήγησαν στην εκπόνηση της Δ.Ε. Περιγράφτηκε εν συντομία το χρονικό και ο τρόπος με τον οποίο έγινε η συνεισφορά για την επίλυση των προβλημάτων.

Στα επόμενα κεφάλαια θα παρουσιαστούν πληροφορίες σχετικά με τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν στο σύνολο τις εργασίας. Θα γίνει σχολιασμός και επεξήγηση σε συγκεκριμένα σημεία του κώδικα τόσο της Android εφαρμογής, όσο και του RESTful API και της βάση που χρειάστηκε να κατασκευαστεί. Τα κεφάλαια 2 και 3 μιλούν για τις διάφορες τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την κατασκευή τη ς Android εφαρμογής και του API αντίστοιχα. Έπειτα στο κεφάλαιο 4 περιγράφεται η ανάλυση και ο σχεδιασμός που υπήρξε για την υλοποίηση ολόκληρου του έργου. Στη συνέχεια, στα κεφάλαια 5 και 6 αναλύονται και επεξηγούνται κομμάτια κώδικα και διαδικασιών που υλοποιήθηκαν τόσο για την Android εφαρμογή όσο και για το API που κατασκευάστηκε. Στο κεφάλαιο 7 θα γίνει παρουσίαση της Android εφαρμογής που θα αναλυθούν οι διάφορες λειτουργίες της και θα δοθούν κάποιες σχετικές οδηγίες. Στο τελευταίο κεφάλαιο θα αναφερθούν κάποια συμπεράσματα και θα αναλυθούν μελλοντικές βελτιστοποιήσεις της εφαρμογής.

Κεφάλαιο 2ο Τεχνολογίες για το Android App

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα πραγματοποιηθεί η ανάλυση τεχνολογιών, εργαλείων, βιβλιοθηκών και αρχιτεκτονικών που χρησιμοποιήθηκαν για την ανάπτυξη της Android εφαρμογής.

2.2 Android

Το Android είναι ένα ανοιχτού κώδικα και βασισμένο στο Linux λειτουργικό σύστημα [1]. Παρουσιάστηκε για πρώτη φορά το 2007 και ενώ αρχικά αναπτύχθηκε από την Android Inc. στη συνέχεια αγοράστηκε από την Google. Γενικά, θεωρείτε ένα λειτουργικό σύστημα για συσκευές με οθόνη αφής όπως κινητά και tablet αλλά δεν περιορίζεται εκεί. Αυτή τη στιγμή χρησιμοποιείται σε διάφορες συσκευές όπως κινητά, tablet, τηλεοράσεις, ρολόγια, αυτοκίνητα κ.α.

2.2.1 Android Studio

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε στις 16 Μαΐου 2013 στο συνέδριο Google I/O και είναι το κύριο IDE της Google για ανάπτυξη εφαρμογών Android[2]. Επίσης, είναι διαθέσιμο ελεύθερα με την άδεια Apache License 2.0. Είναι βασισμένο στο λογισμικό της JetBrains' IntelliJ IDEA και είναι διαθέσιμο για Windows, MacOS και Linux. Επιπρόσθετα, το Android Studio διαθέτει και ειδικό εργαλείο για τον έλεγχο εκδόσεων μέσω Git. Έτσι, η διαδικασία της διαχείρισης εκδόσεων (versioning) πραγματοποιείται με μεγαλύτερη ευκολία.

2.2.2 Android SDK

Το Android SDK είναι ένα σετ ανάπτυξης λογισμικού που αναπτύχθηκε από την Google για την πλατφόρμα Android [1]. Πρόκειται για μια συλλογή από εργαλεία ανάπτυξης λογισμικού και βιβλιοθήκες που απαιτούνται για την ανάπτυξη εφαρμογών Android. Κάθε φορά που η Google κυκλοφορεί μια νέα έκδοση Android ή μια ενημέρωση, κυκλοφορεί επίσης ένα αντίστοιχο SDK, το οποίο πρέπει να κατεβάσουν και να εγκαταστήσουν οι προγραμματιστές. Η τελευταία έκδοση που κυκλοφορεί είναι αυτή του SDK 30 που αντιστοιχεί στο Android 11 [3]. Το Android SDK περιλαμβάνει όλα τα απαραίτητα εργαλεία για τον κώδικα προγραμμάτων από το μηδέν, ακόμη και για τη δοκιμή τους. Τέλος, αξίζει να αναφερθεί ότι είναι συμβατό με Windows, macOS και Linux, ώστε η ανάπτυξη λογισμικού να είναι δυνατή σε οποιαδήποτε από αυτές τις πλατφόρμες.

2.2.3 Kotlin

Η Kotlin είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την ομάδα JetBrains [4]. Είναι πλήρως συμβατή με την Java και τρέχει πάνω στην Εικονική Μηχανή Java (JVM). Επίσης, έχει δεχθεί επιρροές από τις γλώσσες Java, Scala, C#, Groovy.

Η Kotlin στοχεύει κυρίως το JVM, αλλά επίσης μεταγλωττίζεται σε JavaScript (π.χ., για εφαρμογές web frontend που χρησιμοποιούν τη React) ή native code με το LLVM (π.χ., για native iOS εφαρμογές που μοιράζονται επιχειρηματική λογική με εφαρμογές Android).

Από τον Φεβρουάριο του 2012 αναπτύσσεται ως γλώσσα ανοικτού κώδικα. Η πρώτη επίσημη έκδοση 1.0 δημοσιεύτηκε τον Φεβρουάριο του 2016. Τον Μάιο του 2017 η Google ανακοίνωσε ότι η Kotlin θα είναι η επίσημη και προτεινόμενη γλώσσα του Android.

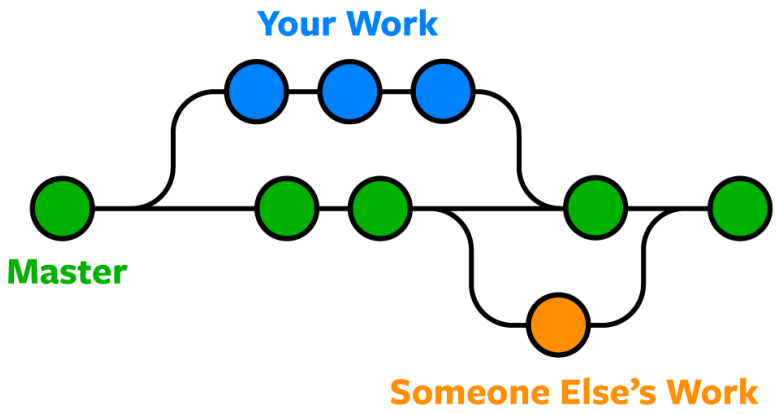
Τέλος, ο κώδικας Java μπορεί να μετατρέπεται σε κώδικα Kotlin, ενώ ένα Android project μπορεί να έχει κώδικα Java και Kotlin ταυτόχρονα.

2.2.4 Android Gradle

Στο Android Studio, το Gradle είναι προεγκατεστημένο και χρησιμοποιείται για την κατασκευή (build) των projects, επομένως παίζει το ρόλο ενός συστήματος κατασκευής (build system) [5]. Ουσιαστικά ο ρόλος του είναι να συγκεντρώνει πόρους, πηγαίο κώδικα, εξωτερικές βιβλιοθήκες κ.α. και να τα συνδυάζει σε ένα τελικό APK. Είναι ένα σύστημα κατασκευής, το οποίο είναι υπεύθυνο για τη συλλογή κώδικα, τη δοκιμή, την ανάπτυξη και τη μετατροπή του κώδικα σε αρχεία .dex και ως εκ τούτου εκτελεί την εφαρμογή στη συσκευή.

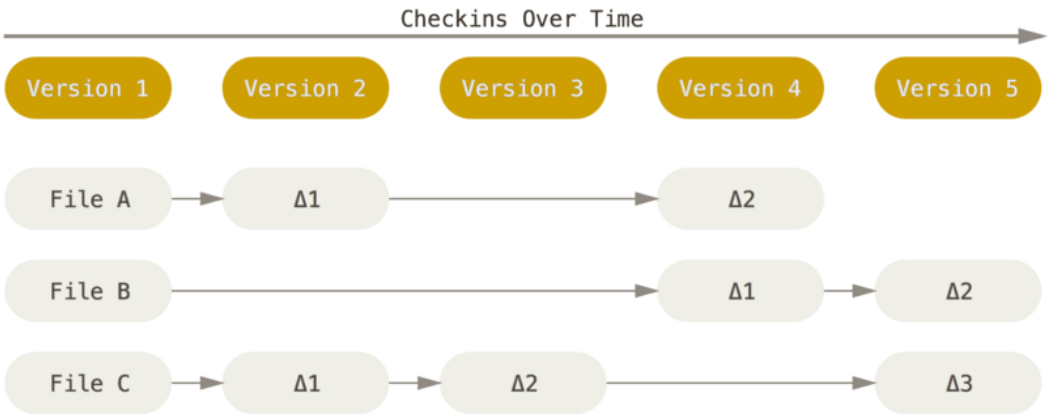
2.3 Git

Το Git είναι το πιο συχνά χρησιμοποιούμενο σύστημα ελέγχου έκδοσης (VCS – Version Control System) [6]. Παρακολουθεί τις αλλαγές που πραγματοποιούνται στα αρχεία, οπότε διαθέτει καταγραφές του τι έχει γίνει και δίνει τη δυνατότητα για επιστροφή σε συγκεκριμένη παλιότερη έκδοση αν αυτό χρειαστεί. Το Git διευκολύνει επίσης τη συνεργασία, επιτρέποντας την συγχώνευση αλλαγών από πολλά άτομα σε ένα κυρίως κλάδο (branch).



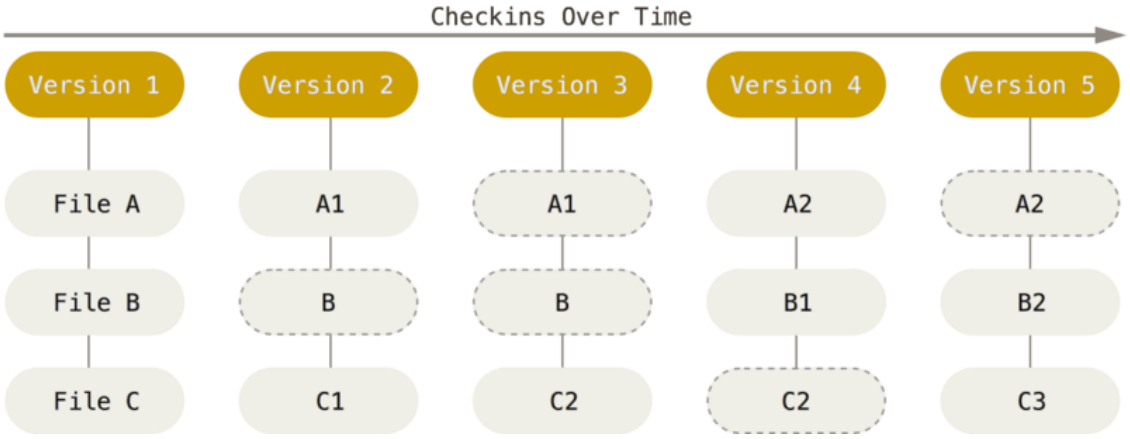
Σχήμα 2.1: Branches - Συγχώνευση αλλαγών

Η κύρια διαφορά μεταξύ του Git και οποιουδήποτε άλλου VCS είναι ο τρόπος που το Git διαχειρίζεται τα δεδομένα του. Εννοιολογικά, τα περισσότερα άλλα συστήματα αποθηκεύουν πληροφορίες ως μια λίστα αλλαγών βάσει αρχείων. Αυτά τα άλλα συστήματα (CVS, Subversion, Perforce, Bazaar κ.α.) διαχειρίζονται τις πληροφορίες που αποθηκεύουν ως ένα σύνολο αρχείων και διαχειρίζονται τις αλλαγές που έγιναν σε κάθε αρχείο με την πάροδο του χρόνου. Αυτός ο τρόπος διαχείρισης είναι γνωστός και ως δέλτα έλεγχος έκδοσης (delta-based version control).



Σχήμα 2.2: Αποθήκευση ως αλλαγές από ένα βασικό/αρχικό αρχείο

Το Git δεν διαχειρίζεται ούτε αποθηκεύει τα δεδομένα του με αυτόν τον τρόπο. Αντ' αυτού, διαχειρίζεται τα δεδομένα του περισσότερο σαν μια σειρά στιγμιότυπων ενός μικροσκοπικού συστήματος αρχείων. Κάθε φορά που δεσμεύεται ή αποθηκεύεται η κατάσταση του έργου, το Git τραβάει μια εικόνα της εμφάνισης όλων των αρχείων εκείνης της στιγμής και αποθηκεύει μια αναφορά σε αυτό το στιγμιότυπο. Για να είναι αποτελεσματικό, εάν τα αρχεία δεν έχουν αλλάξει, το Git δεν αποθηκεύει ξανά το αρχείο, απλώς έναν σύνδεσμο προς το προηγούμενο πανομοιότυπο αρχείο που έχει ήδη αποθηκεύσει. Έτσι τα δεδομένα μοιάζουν περισσότερο σαν μια ροή στιγμιότυπων.



Σχήμα 2.3: Αποθήκευση με στιγμιότυπα

2.3.1 GitHub

Πολλές φορές το GitHub ταυτίζεται λανθασμένα με το Git ενώ στη πραγματικότητα είναι δύο διαφορετικές έννοιες. Πιο συγκεκριμένα, το GitHub είναι μια υπηρεσία φιλοξενίας (hosting service) που βασίζεται σε cloud και επιτρέπει την καλύτερη και ευκολότερη διαχείριση του Git [7]. Προσφέρει τις βασικές υπηρεσίες του δωρεάν. Αν και πιο προηγμένες επαγγελματικές και επιχειρηματικές υπηρεσίες του είναι εμπορικές. Οι δωρεάν λογαριασμοί GitHub χρησιμοποιούνται κατά κόρον για να φιλοξενούν projects ανοιχτού κώδικα.

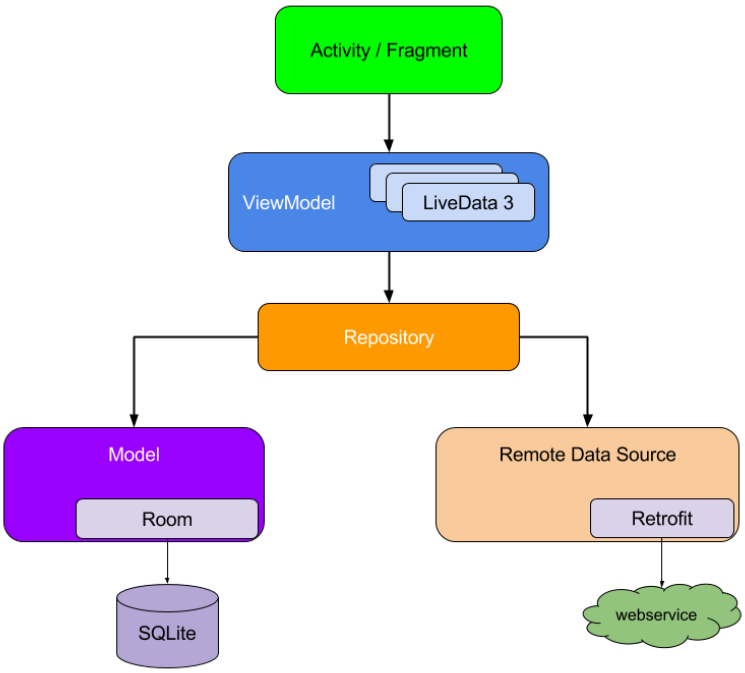
2.4 Αρχιτεκτονικές (Architecture Patterns)

Σε κάθε project είναι επιθυμητό να χρησιμοποιείται κάποια αρχιτεκτονική (architecture partner) έτσι ώστε να οργανώνεται ο κώδικας και να μη παρατηρούνται φαινόμενα με "spaghetti code", δηλαδή μπερδεμένου κώδικα. Φυσικά τα προτερήματα των αρχιτεκτονικών αυτών δεν περιορίζονται μόνο σε

αυτό αλλά και στην γενικότερη ευελιξία που προσφέρουν στο λογισμικό και τον κώδικα κατά την εφαρμογή τους. Το Clean Architecture σε συνδυασμό με το MVVM (Model View ViewModel) είναι ο συνδυασμός των δύο αρχιτεκτονικών που ακολουθήθηκαν για τη υλοποίηση της εφαρμογή.

2.4.1 MVVM

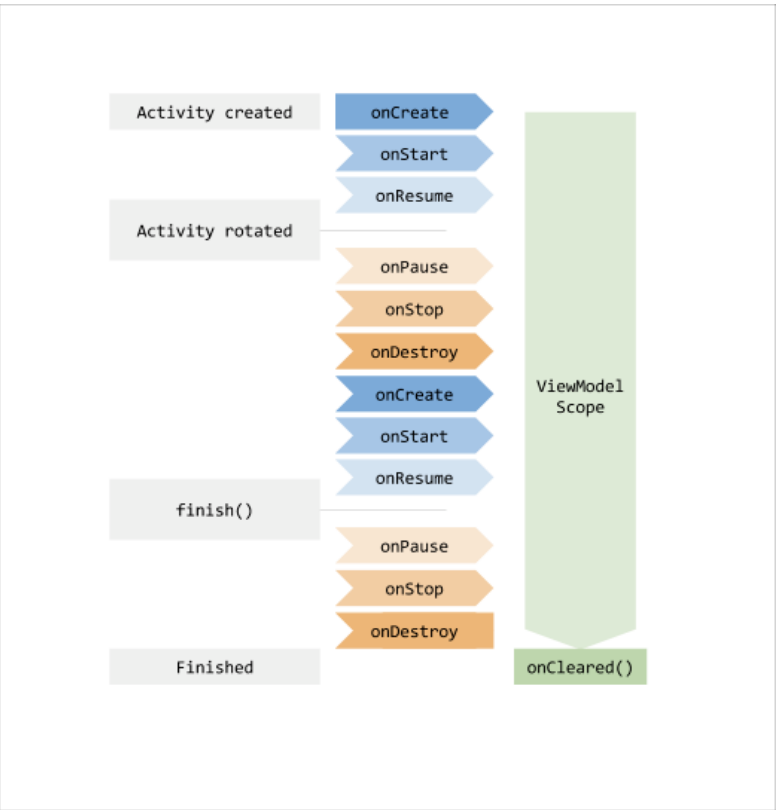
Το MVVM (Model View ViewModel) είναι μία αρχιτεκτονική η οποία είναι ενσωματωμένη στο Android και είναι η προτεινόμενη από την Google [8]. Καθώς και ένα από τα πιο συνηθισμένα λάθη στη δημιουργία Android εφαρμογής είναι η ενσωμάτωση όλου του κώδικα και της λειτουργικότητας στις κλάσεις των Activities και των Fragments, το MVVM δίνει μία οριστική λύση σε αυτό. Στην πραγματικότητα αυτές οι κλάσεις πρέπει να περιέχουν κώδικα που αφορά μόνο το UI και την παρουσίαση των δεδομένων σε αυτό. Οποιαδήποτε άλλη λειτουργία πρέπει να διαφοροποιείτε από αυτές τις κλάσεις και τοποθετείτε σε κλάσεις τύπου ViewModel. Από την Εικόνα 0 καταλαβαίνει κανείς εύκολα ότι ένα Activity/Fragment συνδέεται άμεσα με ένα, ή περισσότερα, ViewModels. Τα ViewModels είναι αυτά τα οποία θα κάνουν τους υπολογισμούς και θα δώσουν τα δεδομένα στο Activity/Fragment.



Σχήμα 2.4: MVVM

Στην αρχιτεκτονική MVVM όμως εισάγεται και μία άλλη έννοια, αυτή του Repository. Το Repository είναι η κλάση η οποία έχει την ευθύνη να επιστρέψει τα δεδομένα που ζητούνται από τη βάση στο ViewModel. Επιπρόσθετα, σε περίπτωση που απαιτείται θα «αποφασίσει» αν θα επιστρέψει τα απομακρυσμένα ή τα τοπικά δεδομένα.

Όπως Φαίνεται και παρακάτω στο Σχήμα 2.5 τα Fragments και τα ViewModels παρόλο που συσχετίζονται έχουν διαφορετική διάρκεια ζωής (lifecycle). Αυτό βοηθάει στην αποφυγή απώλειας δεδομένων κατά τις φάσεις των Activities και των Fragments στις οποίες χάνονται τα αποθηκευμένα και προβαλλόμενα δεδομένα, όπως για παράδειγμα το onPause.



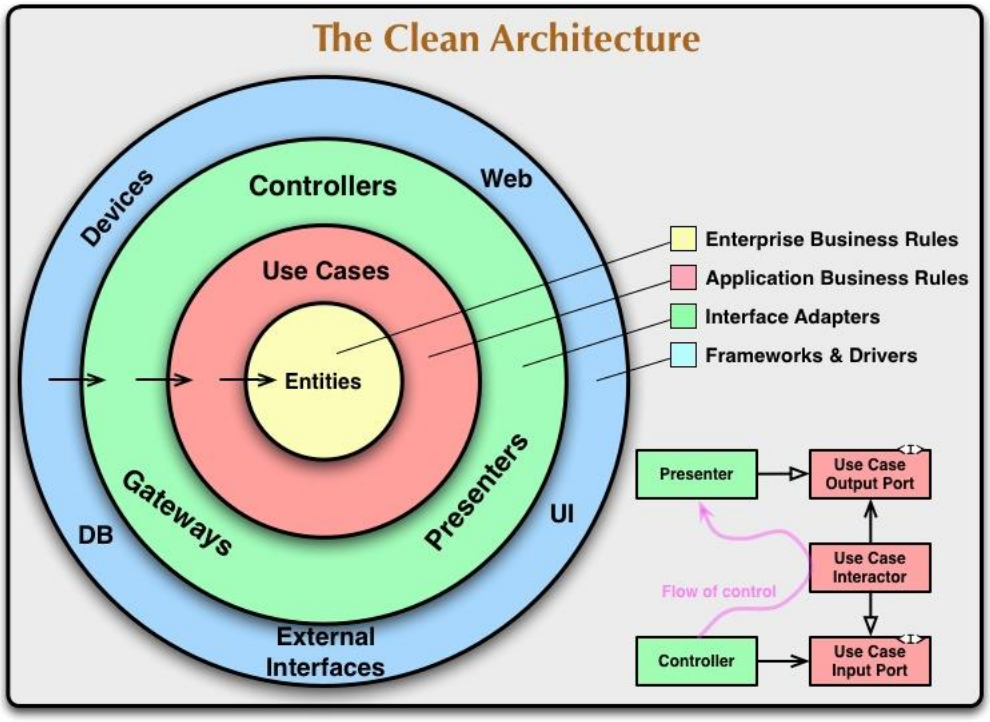
Σχήμα 2.5: MVVM Lifecycle

2.4.2 Clean Architecture

Στόχος του Clean Architecture, όπως και κάθε αρχιτεκτονική, είναι ο διαχωρισμός των εννοιών της εφαρμογής.

Όλες οι αρχιτεκτονικές υπάγονται στις παρακάτω κάποιες βασικές αρχές [9]:

1. Η αρχιτεκτονική δεν εξαρτάται από την ύπαρξη κάποιας βιβλιοθήκης ή λογισμικού. Αυτό επιτρέπει να χρησιμοποιείται χωρίς τους περιορισμούς του εκάστοτε συστήματος ή λογισμικού.
2. Εύκολο testing. Οι επιχειρηματικοί κανόνες μπορούν να δοκιμαστούν χωρίς το UI, τη βάση δεδομένων, τον Web Server ή οποιοδήποτε άλλο εξωτερικό στοιχείο.
3. Ανεξαρτησία UI. Το UI μπορεί να αλλάξει εύκολα, χωρίς να αλλάζει το υπόλοιπο σύστημα και χωρίς να επηρεάσει τους επιχειρηματικούς κανόνες.
4. Ανεξαρτησία από τη βάση δεδομένων. Μπορεί, για παράδειγμα, να πραγματοποιηθεί εύκολα μια αλλαγή Oracle σε SQL Serce, ή για Mongo, BigTable, CouchDB ή κάτι άλλο. Οι επιχειρηματικοί κανόνες δεν δεσμεύουν τη βάση δεδομένων.
5. Ανεξαρτησία από οποιονδήποτε εξωτερικό παράγοντα. Στην πραγματικότητα, οι επιχειρηματικοί κανόνες απλά δεν γνωρίζουν τίποτα για τον «έξω κόσμο».



Σχήμα 2.6: Clean Architecture

Όπως γίνεται εύκολα αντιληπτό από το Σχήμα 2.6 το λογισμικό διαχωρίζεται σε ομόκεντρους κύκλους που αντιπροσωπεύουν διαφορετικούς τομείς του συνολικού κώδικα. Οι εξωτερικοί κύκλοι αφορούν διάφορους μηχανισμούς ενώ οι εσωτερικοί κύκλοι αφορούν πολιτικές και επιχειρηματικούς κανόνες.

Ο βασικός κανόνας είναι η αρχή της εξάρτησης (The Dependency Rule). Σύμφωνα με τον κανόνα αυτό οι εξωτερικοί κύκλοι, δηλαδή οι διάφορες τεχνολογίες όπως για παράδειγμα ο κώδικας που αφορά το UI, μπορούν να αναφέρονται μόνο στους εσωτερικούς κύκλους. Κανένας κύκλος όμως δε μπορεί να γνωρίζει το οτιδήποτε για κύκλο που είναι τοποθετημένος εξωτερικά από τον εαυτό του. Πρακτικά, αυτό σημαίνει ότι οτιδήποτε κώδικας ανήκει σε εσωτερικό κύκλο, δε μπορεί να αναφέρετε σε κώδικα που ανήκει σε πιο εξωτερικό κύκλο. Αυτό συμπεριλαμβάνει μεθόδους, κλάσεις, μεταβλητές ή οτιδήποτε άλλη οντότητα. Το ίδιο ισχύει και με για τα τις μορφές αντικειμένων (data formats) των εξωτερικών κύκλων, τα οποία δεν πρέπει να χρησιμοποιούνται από τους εσωτερικούς κύκλους ως entities.

Πάραυτα , οι κύκλοι είναι σχηματικοί. Υπάρχουν περιπτώσεις που μπορούν να υπάρξουν περισσότεροι ή λιγότεροι από τέσσερις κύκλοι. Δεν υπάρχει κανόνας που να λέει ότι πρέπει να υπάρχουν πάντα αυτοί οι τέσσερις. Ωστόσο, ισχύει πάντα ο κανόνας της εξάρτησης. Οι εξαρτήσεις πηγαίου κώδικα δείχνουν πάντα προς τα μέσα και το λογισμικό γίνεται πιο αφηρημένο στο εσωτερικό του και ενσωματώνει πολιτικές υψηλότερου επιπέδου. Ο εσωτερικός πιο κύκλος είναι ο πιο γενικός. Παρακάτω θα αναλυθούν οι τέσσερις αυτοί κύκλοι και τα χαρακτηριστικά τους.

2.4.2.1 Entities

Μια οντότητα μπορεί να είναι ένα αντικείμενο με μεθόδους ή μπορεί να είναι ένα σύνολο δομών και λειτουργιών δεδομένων. Οι οντότητες (entities) ενσωματώνουν επιχειρηματικούς κανόνες για όλη την επιχείρηση. Σε περίπτωση που η επιχείρηση είναι μόνο μία εφαρμογή τότε οι οντότητες είναι τα επιχειρηματικά αντικείμενα της εφαρμογής. Περιλαμβάνουν τους πιο γενικούς και υψηλού επιπέδου κανόνες. Είναι λιγότερο πιθανό να αλλάξουν όταν αλλάξει κάτι εξωτερικό. Για παράδειγμα, δεν θα

περίμενε κάποιος αυτά τα αντικείμενα να επηρεαστούν από μια αλλαγή στην πλοήγηση της εφαρμογής ή την ασφάλεια. Καμία λειτουργική αλλαγή σε οποιαδήποτε συγκεκριμένη εφαρμογή δεν πρέπει να επηρεάσει το επίπεδο οντοτήτων.

2.4.2.2 Use Cases

Το λογισμικό σε αυτό το επίπεδο περιέχει επιχειρηματικούς κανόνες για συγκεκριμένες εφαρμογές. Περιλαμβάνει και εφαρμόζει όλες τις περιπτώσεις χρήσης (use cases) του συστήματος. Αυτά ενορχηστρώνουν τη ροή δεδομένων από και προς τις οντότητες, και κατευθύνουν τις οντότητες να χρησιμοποιούν τους επιχειρηματικούς κανόνες για την επίτευξη των στόχων των use cases.

Σε αυτό το επίπεδο δεν αναμένουμε οι αλλαγές να επηρεάσουν τις οντότητες. Επίσης, δεν αναμένουμε να επηρεαστεί από αλλαγές σε εξωτερικά στοιχεία, όπως η βάση δεδομένων, το περιβάλλον εργασίας χρήστη κ.α. Ωστόσο, αλλαγές στη λειτουργία της εφαρμογής θα επηρεάσουν τα use cases και επομένως το λογισμικό σε αυτό το επίπεδο. Εάν αλλάξουν οι λεπτομέρειες ενός use case, σίγουρα θα επηρεαστεί κάποιος κωδικός σε αυτό το επίπεδο.

2.4.2.3 Interface Adapters

Το λογισμικό σε αυτό το επίπεδο είναι ένα σύνολο προσαρμογέων (adapters) που μετατρέπουν δεδομένα από τη μορφή που είναι πιο βολική για use cases και entities, στη μορφή που είναι πιο βολική για κάποια εξωτερική λειτουργία όπως η βάση δεδομένων. Αυτό το επίπεδο, για παράδειγμα, θα περιέχει τα datasource τα οποία μετατρέπουν τους τύπους δεδομένων (data format) από αυτό των εξωτερικών λειτουργιών, σε αυτό των use cases και entities.

Τα δεδομένα μετατρέπονται, σε αυτό το επίπεδο, από τη μορφή που είναι πιο βολική για entities και use cases, σε μορφή που είναι πιο βολική ανάλογα την λ, όπως για παράδειγμα για την βάση δεδομένων. Κανένας κωδικός προς τα μέσα αυτού του κύκλου δεν πρέπει να γνωρίζει καθόλου τη βάση δεδομένων. Εάν η βάση δεδομένων είναι μια βάση δεδομένων SQL, τότε όλο το SQL θα πρέπει να περιορίζεται σε αυτό το επίπεδο, και ιδίως στα τμήματα αυτού του επιπέδου που έχουν να κάνουν με τη βάση δεδομένων.

2.4.2.4 Frameworks και Drivers

Το εξωτερικό επίπεδο αποτελείται γενικά από λογισμικό και εργαλεία όπως η βάση δεδομένων, το UI κ.α. Αυτό το επίπεδο είναι όπου πηγαίνουν όλες οι λεπτομέρειες. Το UI είναι μια λεπτομέρεια. Η βάση δεδομένων είναι μια λεπτομέρεια. Κρατάμε αυτά τα πράγματα στο εξωτερικό όπου μπορούν να κάνουν λίγο κακό.

2.4.3 Clean Architecture με MVVM

Ο συνδυασμός των δύο αυτών αρχιτεκτονικών επιλέχθηκε διότι παρά την ενσωματωμένη και προτεινόμενη στο Android αρχιτεκτονική του MVVM υπάρχουν κάποια μειονεκτήματα σε μεγάλα projects. Συγκεκριμένη δεν υπάρχει ξεκάθαρη διάκριση του επιχειρησιακού (business) μέρους του project. Έτσι με την εισαγωγή της λογικής και των στοιχείων του Clean Architecture, όπως τα entities, τα use cases και τα adapters σε συνδυασμό με τα τη λογική του ViewModel επιτυγχάνετε η μέγιστη αποδοτικότητα όσον αφορά την οργάνωση, την επαναχρησιμοποίηση και το testing του κώδικα.

2.5 Αποθήκευση κατάστασης UI

Η διατήρηση και η αποκατάσταση της κατάστασης διεπαφής χρήστη (UI State) μιας οθόνης εγκαίρως και καθ' όλη τη χρονική διάρκεια που απαιτείται η χρήση της είναι ένα κρίσιμο μέρος της εμπειρίας χρήσης [10]. Ο χρήστης αναμένει ότι η κατάσταση της οθόνης θα παραμείνει η ίδια, παρόλο που το σύστημα μπορεί να την καταστρέφει μαζί με οποιαδήποτε κατάσταση είναι αποθηκευμένη σε αυτήν.

Για την γεφύρωση του χάσματος μεταξύ της προσδοκίας του χρήστη και της συμπεριφοράς του συστήματος, χρησιμοποιείται ένας συνδυασμός αντικειμένων ViewModel, της μεθόδου onSaveInstanceState() ή / και ο τοπικός χώρος αποθήκευσης με σκοπό την διατήρηση της κατάστασης του UI σε μεταβάσεις της εφαρμογής κατά τις οποίες η οθόνη καταστρέφεται. Η απόφαση για τον συνδυασμό αυτών των επιλογών εξαρτάται από την πολυπλοκότητα των δεδομένων του UI, τα use cases της εφαρμογής και την ταχύτητας ανάκτησης έναντι της χρήσης μνήμης.

Ανεξάρτητα από την προσέγγιση, πρέπει να διασφαλιστεί ότι η εφαρμογή ανταποκρίνεται στις προσδοκίες των χρηστών σε σχέση με την κατάσταση του UI και παρέχει μια ομαλή και γρήγορη διεπαφή χρήστη. Δηλαδή, πρέπει να αποφεύγεται ο χρόνος καθυστέρησης στη φόρτωση δεδομένων στο περιβάλλον εργασίας του χρήστη, ειδικά μετά από συχνές αλλαγές διαμόρφωσης, όπως περιστροφή) . Στις περισσότερες περιπτώσεις θα πρέπει να χρησιμοποιείται τόσο το ViewModel όσο και το onSaveInstanceState ().

Ανάλογα με την ενέργεια που πραγματοποιεί ένας χρήστης, είτε αναμένει ότι η κατάσταση της δραστηριότητας θα διαγραφεί είτε ότι η κατάσταση θα διατηρηθεί. Σε ορισμένες περιπτώσεις, το σύστημα κάνει αυτόματα αυτό που αναμένεται από τον χρήστη. Σε άλλες περιπτώσεις, όμως, το σύστημα κάνει ακριβώς το αντίθετο. Ο χρήστης αναμένει ότι όταν ξεκινήσει μια δραστηριότητα, η παροδική κατάσταση διεπαφής χρήστη αυτής της δραστηριότητας θα παραμείνει η ίδια έως ότου ο χρήστης απορρίψει εντελώς τη δραστηριότητα. Παραδείγματος χάρη όταν ο χρήστης έχει εισάγει κάποια στοιχεία σε μία οθόνη εγγραφής, και το σύστημα καταστρέψει την οθόνη αυτή τότε τα στοιχεία αυτά δεν πρέπει να χαθούν. Αναμένεται, επίσης, ότι η κατάσταση διεπαφής χρήστη της δραστηριότητάς θα παραμείνει ίδια εάν γίνει προσωρινή αλλαγή σε διαφορετική εφαρμογή και έπειτα επιστροφή στην εφαρμογή αργότερα. Για παράδειγμα, αν πραγματοποιηθεί αναζήτηση στην οθόνη αναζήτησής και, στη συνέχεια, πατηθεί το κουμπί αρχικής σελίδας κατά την επιστροφή στην οθόνη αναζήτησης, αναμένεται να υπάρχει η λέξη-κλειδί αναζήτησης και τα αποτελέσματα, όπως και πριν.

Σε ένα τέτοιο σενάριο, η εφαρμογή τοποθετείται στο παρασκήνιο, το σύστημα κάνει ό,τι μπορεί για να διατηρήσει τη διαδικασία της εφαρμογής στη μνήμη. Ωστόσο, το σύστημα ενδέχεται να καταστρέψει τη διαδικασία εφαρμογής, ενώ ο χρήστης αλληλεπιδρά με άλλες εφαρμογές. Έτσι, η οθόνη καταστρέφεται, μαζί με οποιαδήποτε κατάσταση αποθηκευμένη σε αυτήν. Όταν επανεκκινηθεί η εφαρμογή, η δραστηριότητα θα έχει απροσδόκητα καθαρή κατάσταση μη γνωρίζοντας κανένα από τα δεδομένα που υπήρχαν πριν την καταστροφή της. Η διαδικασία αυτή ονομάζεται process death και πρέπει να λαμβάνεται σοβαρά υπόψιν κατά την ανάπτυξη της εφαρμογής διότι παράλληλα με τα χαμένα δεδομένα μπορούν να δημιουργηθούν και διάφορα σφάλματα που θα σταματήσουν την εφαρμογή.

Στο Σχήμα 2.7 παρουσιάζονται οι τρείς τρόποι αντιμετώπισης των προβλημάτων αυτών. Συνήθως, ο συνδυασμός του ViewModel με το onSaveInstanceState() αρκεί για την πλήρη αποκατάσταση των δεδομένων. Το ViewModel χρησιμοποιεί τη μνήμη RAM του κινητού και δεν υπάρχει καμία εγγραφή δεδομένων στον δίσκο. Έτσι δεν υπάρχει κανένας περιορισμός στην πολυπλοκότητα, ενώ το μέγεθος των δεδομένων περιορίζεται μόνο από την διαθέσιμη μνήμη του κινητού. Η απόκριση στα δεδομένα αυτά κατά την εγγραφή και την ανάγνωση τους είναι πολύ γρήγορη. Σε αντίθεση με ένα Fragment ή

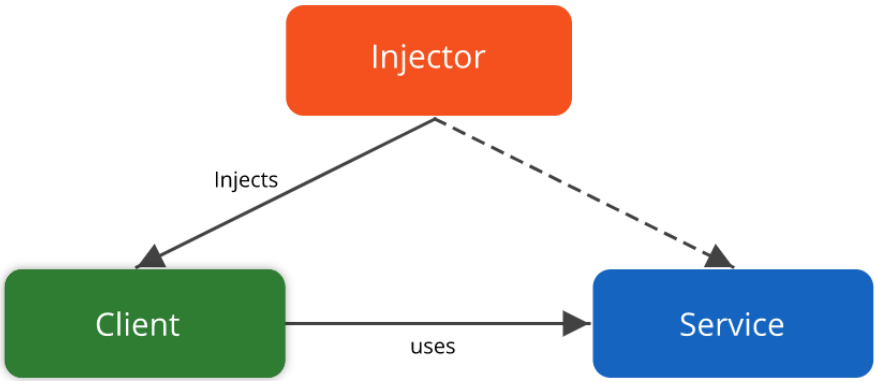
ένα Activity μπορεί να επιβιώσει καταστροφή λόγο περιστροφής του κινητού και άλλες αλλαγές του συστήματος. Παρόλα αυτά δεν μπορεί να επιβιώσει από το process death. Την λύση σε αυτό δίνει το Saved Instance State. Το Saved Instance State σειριοποιεί τα δεδομένα που χρειάζονται στον δίσκο και έτσι επιβιώνουν από οποιαδήποτε καταστροφή της οθόνης. Λόγο της σειριοποίησης στον δίσκο επιτρέπεται η αποθήκευση μόνο μικρών σε μέγεθος δεδομένων, όπως βασικές μεταβλητές (Integer, Float κτλ.) και πολύ μικρά αντικείμενα όπως τα String. Ακόμα και στα String όμως απαιτείται μέγιστη προσοχή στο μέγεθος τους. Τέλος η αποθήκευση στον δίσκο δεν έχει κάποιον περιορισμό σε μέγεθος, εκτός φυσικά από αυτόν της διαθέσιμης χωρητικότητας) όμως η απόκριση σε ανάγνωση και γραφή σε αυτόν είναι πολύ αργή με αποτέλεσμα να αποφεύγεται η χρήση του για τέτοιες περιπτώσεις.

	ViewModel	Saved instance state	Persistent storage
Storage location	in memory	serialized to disk	on disk or network
Survives configuration change	Yes	Yes	Yes
Survives system-initiated process death	No	Yes	Yes
Survives user complete activity dismissal/onFinish()	No	No	Yes
Data limitations	complex objects are fine, but space is limited by available memory	only for primitive types and simple, small objects such as String	only limited by disk space or cost / time of retrieval from the network resource
Read/write time	quick (memory access only)	slow (requires serialization/deserialization and disk access)	slow (requires disk access or network transaction)

Σχήμα 2.7: Αποθήκευση κατάστασης οθόνης (Saving UI States)

2.6 Dependency Injection

Στη μηχανική λογισμικού, το Dependency Injection (DI) είναι μια τεχνική στην οποία ένα αντικείμενο λαμβάνει άλλα αντικείμενα από τα οποία εξαρτάται [11]. Αυτά τα αντικείμενα ονομάζονται εξαρτήσεις (dependencies). Στην τυπική σχέση το αντικείμενο λήψης καλείται πελάτης (client) και το αντικείμενο που έχει περάσει ονομάζεται υπηρεσία (service). Ο κώδικας που μεταβιβάζει το service στον client ονομάζεται “injector”. Αντί ο client να καθορίζει ποια υπηρεσία θα χρησιμοποιήσει, ο injector λέει στον client ποιο service θα χρησιμοποιήσει. Τέλος, το "injection" αναφέρεται στη μετάβαση μιας εξάρτησης δηλαδή ενός service στο αντικείμενο client που θα το χρησιμοποιούσε. Στο σχήμα 2.8 αναπαριστάτε με έναν πιο κατανοητό τρόπο ο παραπάνω ορισμός.



Σχήμα 2.8: Dependency Injection

Πρακτικά η κλάση client "χρησιμοποιεί" κάποιο service το οποίο είναι ένα αντικείμενο. Αυτό το αντικείμενο το ζητάει από τον injector. Ο injector με τη σειρά του δημιουργεί το αντικείμενο "service" και το επιστρέφει έτοιμο προς χρήση στον client. Παρόλα αυτά, η χρήση του DI χωρίς τη βοήθεια κάποιας βιβλιοθήκης είναι κάτι αρκετά χρονοβόρο και δύσκολο καθώς απαιτεί την κατασκευή κάθε κλάσης και των εξαρτήσεων (dependencies) της χειροκίνητα.

2.6.1 Hilt

Το Hilt είναι μια βιβλιοθήκη Dependency Injection για το Android η οποία αυτοματοποιεί αρκετές διαδικασίες του DI [12]. Το Hilt παρέχει έναν τυπικό τρόπο χρήσης του DI στην εφαρμογή σας παρέχοντας containers για κάθε κλάση Android στο project και διαχειρίζοντας αυτόματα τους κύκλους ζωής τους (lifecycle). Επίσης, είναι χτισμένο πάνω από τη δημοφιλή βιβλιοθήκη DI Dagger έτσι ώστε να επωφεληθεί τα διάφορα χαρακτηριστικά του και την υποστήριξη για Android Studio που παρέχει το Dagger. Στο Κεφάλαιο 3 θα αναλυθούν περαιτέρω κάποια σημεία του κώδικα που αφορούν τη βιβλιοθήκη Hilt.

2.7 Android Coroutines

Τα Coroutines είναι ένα design pattern που χρησιμοποιείτε στην Kotlin και κατ' επέκταση στο Android για την εκτέλεση κώδικα ασύγχρονα [13]. Προστέθηκαν στη Kotlin στην έκδοση 1.3 και βασίζονται σε καθιερωμένες έννοιες από άλλες γλώσσες. Ένα από τα μεγαλύτερα πλεονεκτήματα τους είναι η πολύ μεγάλη απλοποίηση του κώδικα που απαιτείται για ασύγχρονες λειτουργίες. Διαχειρίζονται αποτελεσματικά μακροχρόνιες διεργασίες που διαφορετικά θα μπλοκάραν το κύριο νήμα (main thread) και θα έκαναν την εφαρμογή να μην ανταποκρίνεται [14]. Ένας από τους πιο συνηθισμένους λόγους που χρησιμοποιούνται είναι για την λήψη δεδομένων από το διαδίκτυο. Επιπλέον επεξήγηση των Coroutines με παράδειγμα κώδικα θα υπάρξει στο Κεφάλαιο 4.

2.8 ProGuard και R8

Για να γίνει η εφαρμογή όσο το δυνατόν μικρότερη, είναι αναγκαίο να ενεργοποιηθεί η συρρίκνωση (shrinking) για την release έκδοση της εφαρμογής [15]. Η συρρίκνωση έχει την δυνατότητα να καταργεί αχρησιμοποίητο κώδικα και πόρους. Σε παλιότερες εκδόσεις του Gradle η τρόπος συρρίκνωσης μιας εφαρμογής ήταν αποκλειστικά το ProGuard. Πλέον χρησιμοποιείται το R8, το οποίο δουλεύει επίσης με τους ίδιους κανόνες και εντολές με το ProGuard. Επίσης, είναι αποδοτικότερο, καθώς συμπιέζει το αρχείο σε μεγαλύτερο βαθμό, ταχύτερο κατά την εκτέλεση του και πιο συμβατό με την Kotlin. Κάποια

από τα χαρακτηριστικά τόσο του R8 όσο και του ProGuard είναι ότι έχουν την δυνατότητα να αλλάζουν τα ονόματα κλάσεων, μεθόδων και μεταβλητών. Τα όφελος από αυτή την διαδικασία είναι διπλό. Το πρώτο εκ των δύο είναι ότι μειώνεται ο μέγεθος του τελικού αρχείου. Το δεύτερο είναι πως σε περίπτωση που κάποιος κακόβουλος χρήστης επιχειρήσει να αντιστρέψει το τελικό αρχείο (.dex) για να πάρει τον πραγματικό κώδικα της εφαρμογής θα είναι πολύ δύσκολο γι' αυτόν να τον κατανοήσει λόγο τον αλλαγών που έχουν πραγματοποιηθεί σε αυτό. Επίσης, αφαιρείται περιττός κώδικας και αρχεία τα οποία δεν χρησιμοποιούνται. Στις περίπτωση των αρχείων (π.χ. εικόνες) δεν υπάρχει καμία επικινδυνότητα, στο κώδικα που αφαιρείται όμως υπάρχει πιθανότητα να αφαιρεθεί κώδικας ο οποίος εκτελείται αλλά δεν είναι ορατό από το ίδιο το R8. Για τον λόγο αυτό, η χρήση του πρέπει να είναι προσεκτική μαζί με τους κατάλληλους κανόνες από τον προγραμματιστή.

2.9 Firebase

Το Firebase είναι μια πλατφόρμα που αναπτύχθηκε από την Google για τη υποστήριξη της δημιουργίας εφαρμογών για κινητά και ιστούς [16]. Αρχικά ήταν μια ανεξάρτητη εταιρεία που ιδρύθηκε το 2011 από την Envolve. Το 2014, η Google απέκτησε την πλατφόρμα και πλέον είναι ένα από τα μεγαλύτερα, αν όχι η κορυφαία, προσφορά της όσον αφορά την ανάπτυξη εφαρμογών. Η πλατφόρμα του Firebase απαριθμεί 18 από τα οποία θα αναλυθούν μόνο τα δύο. Το Crashlytics και το Analytics τα οποία είναι και τα δύο χαρακτηριστικά της πλατφόρμας που χρησιμοποιεί η εφαρμογή για την λήψη και ανάλυση πληροφορίων. Ιδιαίτερα σημαντικό είναι να τονιστεί ότι όλες οι πληροφορίες που συλλέγονται και χρησιμοποιούνται από αυτές τις υπηρεσίες είναι ανώνυμες και δεν συσχετίζονται με κανέναν τρόπο με τον ίδιο τον χρήστη.

2.9.1 Crashlytics

Η ενσωμάτωση του Crashlytics σε μία εφαρμογή προσφέρει ζωντανή ροή των σφαλμάτων που δημιουργούνται στους χρήστες [17]. Έτσι διευκολύνεται η διαδικασία εύρεσης των σφαλμάτων και εξοικονομείτε σημαντικός χρόνος για την αποκατάσταση τους. Το Crashlytics μπορεί να χρησιμοποιηθεί από εφαρμογές Android, iOS και Unity και προσφέρει διάφορα στοιχεία για τα σφάλματα που προκύπτουν. Κάποια από τα στοιχεία αυτά είναι το σημείο του κώδικα που πραγματοποιήθηκε το σφάλμα, το είδους του σφάλματος, τον αριθμό των χρηστών που είχαν αυτό το σφάλμα, την έκδοση της εφαρμογής, καθώς και πληροφορίες για την συσκευή όπως το μοντέλο και την έκδοση Android που χρησιμοποιεί. Δίνεται, επίσης, η επιλογή για προβολή των σφαλμάτων, είτε ομαδοποιημένα για σφάλματα που έχουν κοινούς παρονομαστές είτε ατομικά για το κάθε σφάλμα ξεχωριστά. Εν κατακλείδι, η διαδικασία της εύρεσης και της επιδιόρθωσης σφαλμάτων διευκολύνεται κατά πολύ μεγάλο βαθμό.

2.9.2 Analytics

Το Google Analytics είναι ένα από τα πιο δυνατά χαρακτηριστικά του Firebase και πρόκειται για ένα εξ ολοκλήρου δωρεάν στοιχείο του [18]. Το Google Analytics ενσωματώνει όλες τις λειτουργίες του Firebase, δίνει πληροφορίες για διάφορες ενέργειες όπως τον χρόνο παραμονής του χρήστη σε κάποια οθόνη. Επιπρόσθετα, είναι εφικτό να προστεθούν συγκεκριμένα συμβάντα από τον προγραμματιστή μέσω του Firebase SDK. Έτσι, μπορούν να υπάρχουν αναλυτικότερες και πιο συγκεκριμένες αναφορές σε σχέση με κάποιο στοιχείο της εφαρμογής που ενδιαφέρει περισσότερο. Οι αναφορές του Google Analytics βοηθούν στην καλύτερη κατανόηση της συμπεριφοράς των χρηστών, κάτι που μπορεί να βελτιώσει κατά πολύ τον μελλοντικό σχεδιασμό της εφαρμογής και του μάρκετινγκ πάνω σε αυτήν. Με την επιπλέον αυτή γνώση, τείνει να γίνεται πολύ πιο εύκολη η λήψη αποφάσεων όπως για παράδειγμα

αποφάσεις που σχετίζονται με τους τομείς της εφαρμογής οι οποίοι πρέπει να αναβαθμιστούν για να βελτιστοποιηθεί όσο το δυνατόν περισσότερο η εμπειρία του χρήστη.

2.10 Βιβλιοθήκες

Παρακάτω θα αναλυθούν οι βιβλιοθήκες που χρησιμοποιήθηκαν για την Android εφαρμογή.

2.10.1 Google Material

Η βιβλιοθήκη Material είναι ένα σύστημα σχεδίασης που δημιουργήθηκε από την Google για να βοηθήσει τους προγραμματιστές να δημιουργήσουν ψηφιακές εμπειρίες υψηλής ποιότητας για Android, iOS, Flutter και το Web [20]. Περιέχει ένα πλήθος από έτοιμα διαδραστικά δομικά στοιχεία (components) για τη δημιουργία του UI, όπως TextViews, Data Pickers, Toolbars κ.α. Εκτός από τα components που προσφέρει η Google στα πλαίσια του συστήματος σχεδίασης προσφέρει και αρκετές σχεδιαστικές οδηγίες (guidelines).

Material is a design system created by Google to help teams build high-quality digital experiences for Android, iOS, Flutter, and the web.

2.10.2 Moshi

Το Moshi είναι μια σύγχρονη βιβλιοθήκη JSON για Android. Διευκολύνει την μετατροπή JSON αρχείων σε αντικείμενα Java και κατ' επέκταση Kotlin και το αντίστροφο [21]. Έχει τη δυνατότητα να εκτελεί αυτόματα την μετατροπή από τη μία μορφή στην άλλη αλλά είναι εφικτό να παραμετροποιηθεί η διαδικασία αυτή με κάποιον custom adapter.

2.10.3 okHttp

Το OkHttp είναι ένας HTTP και HTTP/2 Client για εφαρμογές Android και Java [22]. Διαθέτει προηγμένες λειτουργίες, όπως ομαδοποίηση σύνδεσης (εάν δεν είναι διαθέσιμο το HTTP/2), συμπίεση GZIP και αποθήκευση απόκρισης (response caching) για αποφυγή της χρήσης δικτύου για επαναλαμβανόμενα αιτήματα. Είναι επίσης σε θέση να διαχειριστεί κοινά προβλήματα σύνδεσης και, σε περίπτωση αποτυχίας σύνδεσης, εάν μια υπηρεσία έχει πολλές διευθύνσεις IP, μπορεί να επαναλάβει το αίτημα σε εναλλακτικές διευθύνσεις.

2.10.4 Retrofit

Το Retrofit είναι ένας REST Client για το Android και τη Java [23]. Η βιβλιοθήκη αυτή, είναι από τις πιο διάσημες καθώς κάνει την κύρια δουλειά στην επικοινωνία με ένα REST API.

Στο Retrofit ρυθμίζεται ποιος μετατροπέας χρησιμοποιείται για την σειριοποίηση δεδομένων. Συνήθως, για την σειριοποιήση και την μετατροπή των αντικειμένων από και σε JSON χρησιμοποιείται κάποια βιβλιοθήκη όπως η Gson ή η Moshi. Σε αυτό το project χρησιμοποιήθηκε η βιβλιοθήκη Moshi. Επίσης, είναι δυνατή και η μετατροπή από και σε άλλες μορφές αντί για JSON, όπως η XML, χρησιμοποιώντας έναν αντίστοιχο μετατροπέα.

2.10.5 Glide

Το Glide είναι μια βιβλιοθήκη εικόνων για το Android που αναπτύχθηκε από την bumptech και συνιστάται από την Google [24]. Έχει χρησιμοποιηθεί σε πολλά projects ανοιχτού κώδικα της Google,

συμπεριλαμβανομένης της επίσημης εφαρμογής Google I/O 2014. Παρέχει κινούμενη υποστήριξη GIF και γενικότερα χειρίζεται τη φόρτωση/αποθήκευση των εικόνων.

2.10.6 MPAndroidChart

Το MPAndroidChart είναι μια εύχρηστη και αρκετά διαδεδομένη βιβλιοθήκη σχεδιαγραμμάτων για το Android [25]. Είναι μια open-source βιβλιοθήκη από την PhilJay. Επιτρέπει την δημιουργία πολλών ειδών σχεδιαγραμμάτων. Παράλληλα διαθέτει την ευελιξία για απόλυτο έλεγχο της εμφάνισης των σχεδιαγραμμάτων μέσω ενός πλήθος επιλογών για επεξεργασία.

Κεφάλαιο 3ο Τεχνολογίες του Backend

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα πραγματοποιηθεί η ανάλυση τεχνολογιών και εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη του Backend. Πιο συγκεκριμένα το Backend αναφέρεται στη βάση δεδομένων αλλά και το API που χρειάστηκε να κατασκευαστεί.

3.2 MySQL

Η SQL (Structured Query Language) είναι μία γλώσσα για τη δημιουργία, αποθήκευση και επεξεργασία δεδομένων [26]. Υλοποιεί το σχεσιακό μοντέλο, καθώς τα αποτελέσματα των εντολών που εκτελούνται είναι πίνακες. Το σχεσιακό μοντέλο οργανώνει τα δεδομένα σε πίνακες. Οι πίνακες αποτελούνται από στήλες και γραμμές και μπορούν να είναι αποθηκευμένοι είτε προσωρινά είτε μόνιμα.

Οι στήλες ορίζουν τον τύπο των δεδομένων, ενώ οι γραμμές αποτελούν τα ίδια τα δεδομένα. Οι πίνακες πρέπει να έχουν μία ή ένα σύνολο στηλών οι οποίες θα αποτελούν το κύριο κλειδί και θα είναι μοναδικές τιμές. Προαιρετικά μπορούν να περιέχουν στήλες οι οποίες αποτελούν ξένο κλειδί. Το ξένο κλειδί αποτελεί τιμή που ανήκει σε γραμμή άλλου πίνακα και χρησιμοποιείται για συσχέτιση των δεδομένων διαφορετικών πινάκων.

3.3 MariaDB

Η MariaDB είναι ένα RDBMS (Σχεσιακό Σύστημα Διαχείρισης Βάσης Δεδομένων) το οποίο αξιοποιεί την SQL για τη διαχείριση της βάσης δεδομένων [27]. Αποτελεί ένα ανοιχτού κώδικα RDBMS, το οποίο δημιουργήθηκε το 2009 ως κόμβος της MySQL, ύστερα από την εξαγορά της από την Oracle Corporation. Η MySQL αποτελεί το δημοφιλέστερο RDBMS στον κόσμο. Η MariaDB αρχικά δημιουργήθηκε ως μια εναλλακτική της MySQL που θα ήταν πλήρως συμβατή με αυτή, αλλά με την πάροδο του χρόνου έχουν προστεθεί νέα χαρακτηριστικά, τα οποία είναι μη συμβατά με αυτή. Η SQL και η MariaDB αποτελούν τους πυλώνες του διαδικτυακού API, καθώς η δυνατότητα που προσφέρουν για δομημένη αποθήκευση πληροφορίας, επέτρεψε τον ευκολότερο σχεδιασμό της βάσης δεδομένων.

3.4 JSON

Το JSON ή αλλιώς JavaScript Object Notation είναι ένα πρότυπο παράστασης και οργάνωσης δεδομένων [28]. Η ανάγνωση θεωρείται εύκολη τόσο από ανθρώπους όσο και από μηχανές. Η δομή του θυμίζει αυτή των αντικειμένων της γλώσσας JavaScript, ωστόσο υποστηρίζεται από τις περισσότερες γλώσσες προγραμματισμού.

Αποτελείται από ζεύγη χαρακτηριστικών και τιμών. Τα ονόματα πρέπει να περικλείονται από εισαγωγικά, ενώ οι τιμές μπορούν να είναι αλφαριθμητικά, τύποι δεδομένων αλήθειας (boolean), null ή πίνακες. Επιπρόσθετα, οι τιμές μπορούν να αποτελούν οι ίδιες αντικείμενα. Με τη σειρά τους, τα αντικείμενα αυτά περιέχουν τα δικά τους ζεύγη ονομάτων και τιμών.

Υπάρχουν πολλές βιβλιοθήκες για την αυτόματη μετατροπή των δεδομένων JSON σε αντικείμενα της γλώσσας που επιθυμούμε. Φυσικά η μετατροπή μπορεί να γίνει και χωρίς τη χρήση κάποιας εξωτερικής βιβλιοθήκης αν το επιθυμεί ο προγραμματιστής.

3.5 PHP

Η PHP (αναδρομικό ακρωνύμιο: Hypertext Preprocessor) είναι μια ευρέως χρησιμοποιούμενη γλώσσα ανοιχτού κώδικα γενικής χρήσης που είναι ιδιαίτερα κατάλληλη για ανάπτυξη διαδικτυακών εφαρμογών [29]. Δημιουργήθηκε το 1995 από τον Rasmus Lerdorf και η τελευταία της έκδοση είναι η 8.0.5 στις 29 Απριλίου του 2021. Ένα από τα σημαντικότερα χαρακτηριστικά της, είναι η υποστήριξη ενός μεγάλου πλήθους συστημάτων διαχείρισης βάσεων δεδομένων, από τις πολύ διαδεδομένες (MariaDB, PostgreSQL κ.α.) έως και κάποιες πιο σύγχρονες (MongoDB κ.α.), χωρίς την ανάγκη για εγκατάσταση κάποιου πρόσθετου.

Κάτι ακόμη το οποίο διακρίνει την PHP από μία γλώσσα που εκτελείτε τον πελάτη (client) όπως η JavaScript είναι ότι ο κώδικας εκτελείται στον διακομιστή, δημιουργώντας HTML ή κάποιας άλλης μορφής κείμενο το οποίο στη συνέχεια αποστέλλεται στον client. Ο client θα λάβει τα αποτελέσματα της εκτέλεσης αυτού του σεναρίου, αλλά δεν γνωρίζει ούτε μπορεί να μάθει τον πραγματικό κώδικα που εκτελέστηκε. Με αυτόν τον τρόπο μπορεί να υλοποιηθεί εύκολα και ένα RESTful API. Η PHP τρέχει τον κώδικα στον server μετά από το αντίστοιχο HTTP ερώτημα και επιστρέφει το JSON αποτέλεσμα στον client.

3.5.1 PDO

Η επέκταση PDO (PHP Data Objects) ορίζει μια ελαφριά, συνεπή διεπαφή για πρόσβαση σε βάσεις δεδομένων στην PHP και παρέχεται μαζί της [30]. Κάθε πρόγραμμα οδήγησης (driver) βάσης δεδομένων που εφαρμόζει τη διεπαφή PDO μπορεί να εκτελέσει συγκεκριμένες λειτουργίες της βάσης δεδομένων ως κανονικές λειτουργίες επέκτασης. Παρόλα αυτά, δεν είναι δυνατή η εκτέλεση καμίας λειτουργίας βάσης δεδομένων με την επέκταση PDO από μόνη της. Χρειάζεται η επιπρόσθετη λειτουργεία ενός προγράμματος οδήγησης (driver) PDO για τη συγκεκριμένη βάση για να αποκτηθεί πρόσβαση σε έναν διακομιστή βάσης δεδομένων. Επίσης, ανεξάρτητα από τη βάση δεδομένων που χρησιμοποιείται, χρησιμοποιούντα οι ίδιες λειτουργίες και μέθοδοι για την πραγματοποίηση ερωτημάτων και τη λήψη δεδομένων.

3.6 RESTful API

Το REST (Represantation State Transfer) είναι μια αρχιτεκτονική σχεδίασης λογισμικού για τη δημιουργία διαδικτυακών εφαρμογών [31]. Χρησιμοποιεί ένα υποσύνολο από τις υπάρχουσες μεθόδους του πρωτοκόλλου HTTP. Πρόκειται για τις μεθόδους των GET, PUT, POST και DELETE. Οι τέσσερις αυτές μέθοδοι συνθέτουν την αρχιτεκτονική CRUD. Πιο συγκεκριμένα αναφέρονται στο διάβασμα (reading), την ενημέρωση (updating), τη δημιουργία (creating) και τη διαγραφή (delete) των δεδομένων αντίστοιχα.

Ένα API, για να μπορεί να θεωρηθεί RESTful θα πρέπει να πληροί κάποια συγκεκριμένα κριτήρια που ορίζονται από την αρχιτεκτονική. Συγκεκριμένα θα πρέπει να υλοποιεί την αρχιτεκτονική CRUD, όπως αναφέρθηκε παραπάνω. Όλοι σύνδεσμοι του API, θα πρέπει να βασίζονται σε ένα αρχικό API και να προσφέρεται σε κάθε απόκριση του ο τύπος δεδομένων που επιστρέφεται. Επιπρόσθετα, κατά την επικοινωνία πελάτη-εξυπηρετητή (client-server) η κατάσταση των πόρων αποθηκεύεται στον εξυπηρετητή και είναι διαθέσιμη σε όλους τους πελάτες, ενώ η κατάσταση εφαρμογής αποθηκεύεται στον πελάτη και αφορά αποκλειστικά αυτόν.

Παρόλα αυτά είναι σύνηθες ένα API να υλοποιεί τις μεθόδους του συνδυάζοντας μόνο τις μεθόδους GET και POST ή ανάλογα την περίσταση και της ανάγκες της εφαρμογής η υλοποίηση να πραγματοποιείται με μόνο μία από τις δύο μεθόδους. Αναλυτικότερα, κάποια από τα χαρακτηριστικά

της GET είναι ότι το ερώτημα μπορεί να αποθηκευτεί τοπικά στη cache, υπάρχει περιορισμός στο μέγεθος των πληροφοριών του ερωτήματος, ενώ τα ίδια τα ερωτήματα μπορούν να διαβαστούν και να αποθηκευτούν πολύ εύκολα από τον οποιοδήποτε. Για τους λόγους αυτούς η χρήση της GET πρέπει να είναι μόνο για την ανάγνωση δεδομένων και όχι για την τροποποίηση τους. Απαγορευτική, επίσης, είναι η χρήση της όταν μεταφέρονται ευαίσθητα δεδομένα. Σε τέτοιες περιπτώσεις η χρήση της POST είναι η συνιστώμενη καθώς το ερώτημα δε μπορεί να διαβαστεί ή να αποθηκευτεί από τρίτους. Επίσης, το ερώτημα δεν αποθηκεύετε ποτέ στην cache και ούτε υπάρχει ο περιορισμός μεγέθους στο ερώτημα.

3.7 PHPStorm

Το PHPStorm είναι ένα καινοτόμο, ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που βασίζεται στη Java και έχει σχεδιαστεί από τη JetBrains [32]. Απευθύνεται κυρίως για PHP και web προγραμματισμό. Υποστηρίζει PHP 5.3 / 5.4 / 5.5 / 5.6 / 7.0 / 7.1 / 7.2 / 8.0, παρέχει πρόληψη σφαλμάτων, βέλτιστη αυτόματη συμπλήρωση και αναδιαμόρφωση κώδικα και ένα εκτεταμένο πρόγραμμα επεξεργασίας HTML, CSS και JavaScript. Επίσης, παρέχει έξυπνη ολοκλήρωση κώδικα, επισήμανση σύνταξης, εκτεταμένη διαμόρφωση μορφοποίησης κώδικα, έλεγχο σφαλμάτων και πολλά άλλα.

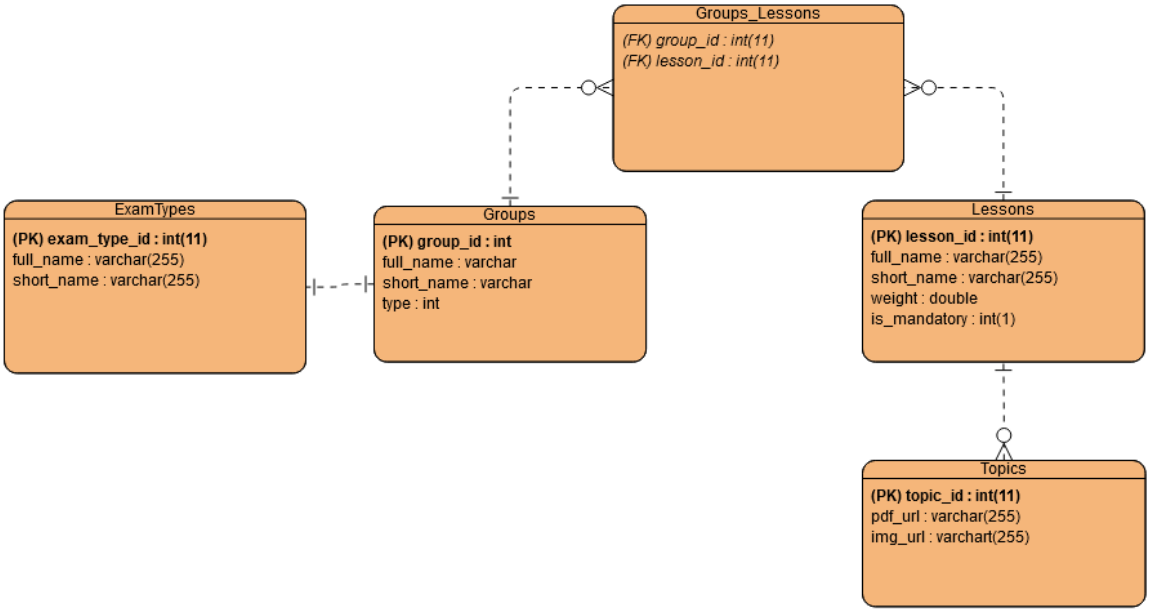
3.8 phpMyAdmin

Το phpMyAdmin είναι ένα εργαλείο ελεύθερου λογισμικού γραμμένο σε PHP, το οποίο προορίζεται για τη διαχείριση της MySQL μέσω του διαδικτύου [33]. Η πρώτη έκδοση βγήκε το 1999, ενώ από το 2015 η ανάπτυξη του βασίζεται εξ ολοκλήρου στο GitHub. Το phpMyAdmin υποστηρίζει ένα ευρύ φάσμα λειτουργιών της MySQL και της MariaDB. Οι λειτουργίες που χρησιμοποιούνται συχνά είναι η διαχείριση βάσεων δεδομένων, πινάκων, στηλών, σχέσεων, ευρετηρίων, χρηστών, δικαιωμάτων κ.α. Όλες αυτές οι λειτουργίες μπορούν να πραγματοποιηθούν μέσω της διεπαφής χρήστη, ενώ παράλληλα υπάρχει και κλασσική δυνατότητα για απευθείας εκτέλεση οποιουδήποτε κώδικα SQL.

Κεφάλαιο 4ο Σχεδιασμός και ανάλυση

4.1 ER Diagram

Το μοντέλο Entity Relationship (ER) είναι μια από τις πιο δημοφιλείς μεθοδολογίες για το σχεδιασμό σχεσιακών βάσεων δεδομένων [34]. Έχουν αναπτυχθεί αρκετά εμπορικά προϊόντα για την υποστήριξη σχεδίασης διαγραμμάτων ER με γραφικό τρόπο. Περιέχει τρία βασικά στοιχεία, σύνολα οντοτήτων, σύνολα σχέσεων και περιορισμούς ακεραιότητας. Το σύνολο οντοτήτων μοντελοποιεί τα αντικείμενα που εμπλέκονται σε ένα έργο όπως τύποι εξετάσεων, πεδία, μαθήματα και ούτω καθεξής. Το σύνολο σχέσεων μοντελοποιεί τις συσχετίσεις μεταξύ αυτών των αντικειμένων. Για παράδειγμα, ένα μάθημα ανήκει σε ένα ή περισσότερα πεδία και το κάθε πεδίο ανήκει σε έναν τύπο εξέτασης. Οι περιορισμοί ακεραιότητας καθορίζουν τους περιορισμούς ορθότητας που εξαρτώνται από την εφαρμογή έναντι οντοτήτων και σχέσεων. Για παράδειγμα, ένας τέτοιος περιορισμός θα ήταν αν το κάθε μάθημα έπρεπε να έχει τουλάχιστον ένα θέμα ανά έτος. Για τις ανάγκες του συγκεκριμένου έργου δημιουργήθηκε το διάγραμμα ER του Σχήματος 4.1.



Σχήμα 4.1: ER Diagram

Η οντότητα του ExamTypes αντιπροσωπεύει του τύπους εξετάσεων. Τέτοιοι τύποι μπορεί να είναι το Γενικό Λύκειο 90%, το Επαγγελματικό Λύκειο 90%, οι αντίστοιχες κατηγορίες για 10% καθώς και πιο ειδικές κατηγορίες όπως για ομογενείς. Η οντότητα Lessons αντιπροσωπεύει τα μαθήματα. Κάθε μάθημα αποτελείται από ένα μοναδικό id, το πλήρες όνομα του και τη συντόμευση του, τον συντελεστή του για τον υπολογισμό μορίων και το αν είναι υποχρεωτικό μάθημα ή επιλογής. Επίσης, τα μαθήματα μπορούν να έχουν παλιά θέματα. Κάθε θέμα, δηλαδή οντότητα Topics, απαρτίζεται από το μοναδικό id του, την διεύθυνση του PDF αρχείου του και μία εικόνα. Τα μαθήματα ανήκουν επίσης σε ένα ή περισσότερα Groups. Η οντότητα Groups είναι αντίστοιχη των ομάδων προσανατολισμού. Κάθε τέτοια ομάδα μπορεί να έχει πολλά μαθήματα καθώς και κάθε μάθημα μπορεί να ανήκει σε πολλά Groups. Η σχέση αυτή διατυπώνεται μέσω της οντότητας Groups_Lessons.

4.2 UI και UX

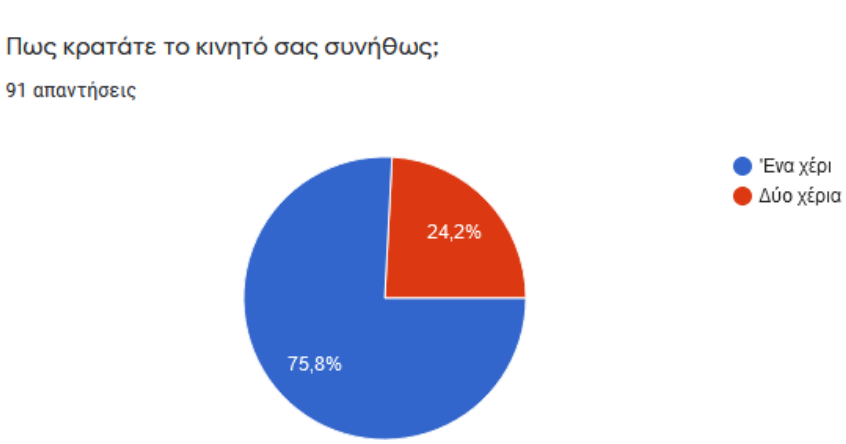
Το UX και UI Design είναι δύο έννοιες πολύ σημαντικές για την ανάπτυξη κάθε εφαρμογής και αποτελούν πρωταγωνιστικό ρόλο στο τελικό αποτέλεσμα. Το UX Design αναφέρεται στον σχεδιασμό της εμπειρίας χρήσης (User Experience), ενώ το UI Design αντιπροσωπεύει το σχεδιασμό διεπαφής χρήστη, δηλαδή το γραφικό στοιχείο της εκάστοτε εφαρμογής. Για την καλύτερη κατανόηση των προτιμήσεων των χρηστών και με σκοπό την ανάπτυξη μιας εφαρμογής που θα έχει όσο το δυνατόν πιο βέλτιστο UX αναπτύχθηκε ένα ερωτηματολόγιο που ερευνά κάποιες βασικές συνήθειες και προτιμήσεις που έχουν οι χρήστες στην αλληλεπίδραση τους με ένα κινητό τηλέφωνο. Αντίστοιχα, για την επιλογή του κατάλληλου UI για την εφαρμογή κατασκευάστηκαν κάποιες μακέτες. Στις παρακάτω παραγράφους θα αναλυθούν περαιτέρω οι δύο αυτές διαδικασίες και τα αποτελέσματα τους.

4.3 Ερωτηματολόγιο

Στο ερωτηματολόγιο συμμετείχαν 91 άτομα τα οποία απάντησαν σε γενικές ερωτήσεις σχετικά με τις προτιμήσεις, τις συνήθειες και διάφορες συνθήκες που επικρατούν κατά την αλληλεπίδραση τους με το κινητό τους. Από τα 91 αυτά άτομα συμμετείχαν 27 μαθητές λυκείου. Οι κύριες πτυχές του ερωτηματολόγιου είναι πέντε και αναλύονται παρακάτω.

4.3.1 Προτίμηση κρατήματος του κινητού

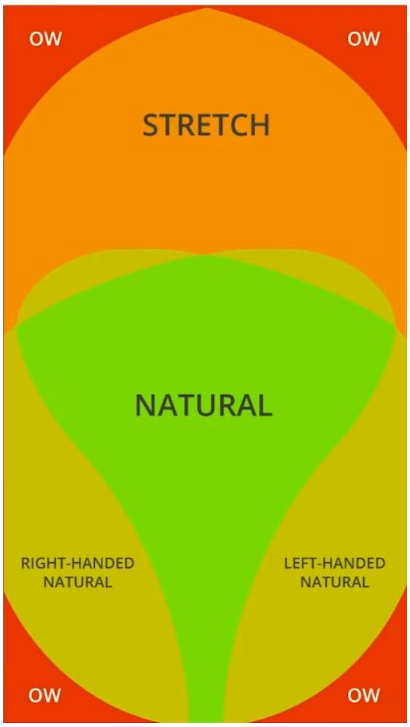
Η ερώτηση για την προτίμηση στο κράτημα του κινητού ζητήθηκε καθώς είναι μία σημαντική πληροφορία η οποία μπορεί να κάνει καλύτερη την εμπειρία χρήσης. Το αποτέλεσμα ήταν συντριπτικά μεγαλύτερο υπέρ του της χρήσης κινητού με το ένα χέρι. Πιο συγκεκριμένα το 75.8% απάντησε ότι κρατάει το κινητό με το ένα χέρι συνήθως ενώ μόλις το 24,2% απάντησε ότι κρατάει πιο συχνά το κινητό με τα δύο χέρια. Καθώς, το αποτέλεσμα ήταν αρκετά ξεκάθαρο η εφαρμογή σχεδιάστηκε έτσι ώστε να είναι όσο το δυνατόν γίνεται πιο φιλική για χρήση με το ένα χέρι.



Σχήμα 4. 2: Ερώτηση 1

Όπως φαίνεται και στο Σχήμα 4.2 κατά την χρήση του κινητού με το ένα χέρι υπάρχουν περιοχές που μπορούν να πατηθούν με μεγαλύτερη ευκολία, έως περιοχές που είναι πολύ δύσκολο να χρησιμοποιηθούν. Αναλυτικότερα η περιοχή με το πράσινο είναι κατά κύριο λόγο η περιοχή που είναι εύκολο και μπορεί να πατηθεί με φυσικό τρόπο, ενώ στις υπόλοιπες περιοχές το χέρι χρειάζεται να

ασκήσει μεγαλύτερη προσπάθεια για να τις χρησιμοποιήσει. Τέλος, οι τέσσερις γωνίες είναι και αυτές που είναι πολύ δύσκολο έως αδύνατον να χρησιμοποιηθούν με την χρήση μόνο τους ενός χεριού.

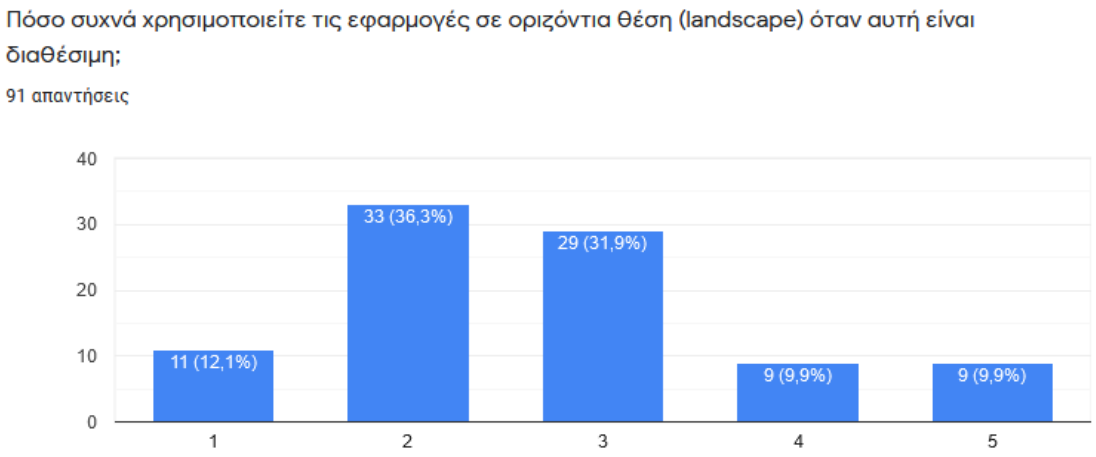


Σχήμα 4. 3: Γράφημα περιοχών πατήματος με χρήση κινητού με το ένα χέρι

Φυσικά δεν είναι δυνατόν όλες οι λειτουργίες να πραγματοποιούνται μέσα στην εύχρηστη (πράσινη) περιοχή αλλά καταβλήθηκε προσπάθεια ώστε ο χρήστης να νιώθει όσο το περισσότερο πιο άνετα γίνεται χρησιμοποιώντας την εφαρμογή με το ένα του χέρι.

4.3.2 Προτίμηση χρήσης κινητού σε οριζόντια θέση

Η δεύτερη ερώτηση που τέθηκε στο ερωτηματολόγιο αφορούσε την προτίμηση της χρήσης του κινητού σε οριζόντια θέση όταν επιτρέπεται αυτό από την εφαρμογή. Πιο συγκεκριμένα στις Android συσκευές υπάρχει η δυνατότητα για προβολή της εφαρμογής είτε σε κάθετη θέση (portrait), είτε σε οριζόντια θέση (landscape). Είναι εφικτό μια εφαρμογή να σχεδιαστεί και για τις δύο περιπτώσεις ή μόνο για μία από τις δύο. Στην ερώτηση που τέθηκε ο χρήστης έπρεπε να εκφράσει το πόσο συχνά χρησιμοποιεί την οριζόντια θέση στο κινητό με έναν αριθμό από το 1 έως το 5, όπου το ένα αντιστοιχεί στο καθόλου και το 5 στο πολύ συχνά. Συλλέγοντας τα αποτελέσματα, παρατηρήθηκε ότι το μεγαλύτερο ποσοστό τείνει να μη χρησιμοποιεί πολύ συχνά αυτό το στατιστικό. Ειδικότερα, μόνο το 20% ανήκει σε κατηγορία που χρησιμοποιεί τις εφαρμογές του συχνά σε οριζόντια θέση.



Σχήμα 4. 4: Ερώτηση 2

Τα δεδομένα που παρουσιάστηκαν παραπάνω και φαίνονται στο Σχήμα 4.4 αναλύθηκαν και μελετήθηκαν πριν την ανάπτυξη της εφαρμογής. Τελικά, αποφασίστηκε η ανάπτυξή της εφαρμογής να είναι μόνο για οριζόντια θέση (portrait).

4.3.3 Γλώσσα προτίμησης στις εφαρμογές

Ένα ακόμα σημαντικό ζήτημα που τίθεται κατά τον σχεδιασμό των εφαρμογών είναι η επιλογή γλώσσας. Στο ερώτημα που τέθηκε για την προτίμηση της γλώσσας ανάμεσα σε ελληνικά και αγγλικά τα ελληνικά υπερίσχυσαν με 62,6%. Επειδή όμως το αποτέλεσμα είναι σχετικά κοντά και εξαιτίας της ευκολίας που προσφέρει το Android στην προσθήκη περισσότερων γλωσσών στην εφαρμογή αποφασίστηκε η ανάπτυξη της εφαρμογής και στις δύο γλώσσες. Έτσι ο χρήστης θα έχει την ευελιξία για να επιλέξει στην επιθυμητή γλώσσα. Τέλος, μέσω αυτού του αποτελέσματος αποφασίστηκε η εφαρμογή να έχει ως προεπιλεγμένη γλώσσα τα ελληνικά.

Σχήμα 4. 5: Ερώτηση 3

4.3.4 Προτίμηση θέματος στις εφαρμογές

Στην ερώτηση για την προτίμηση του θέματος δόθηκαν δύο επιλογές, η επιλογή του φωτεινού θέματος και του σκοτεινού θέματος. Το αποτέλεσμα ήταν υπέρ του σκοτεινού θέματος με ποσοστό 68,1% για το σκοτεινό και 31,9% για το φωτεινό. Και πάλι αποφασίστηκε να υπάρχει η ευελιξία επιλογής του

επιθυμητού θέματος. Αυτή τη φορά όμως σε αντίθεση με την προηγούμενη περίπτωση η προεπιλογή του θέματος θα είναι αυτή του συστήματος και όχι μία από αυτές τις δύο επιλογές. Έτσι ανάλογα με το αν ο χρήστης έχει ενεργοποιημένη ή απενεργοποιημένη την σκοτεινή λειτουργία στο κινητό του η εφαρμογή θα προσαρμόζεται.

Σχήμα 4. 6: Ερώτηση 4

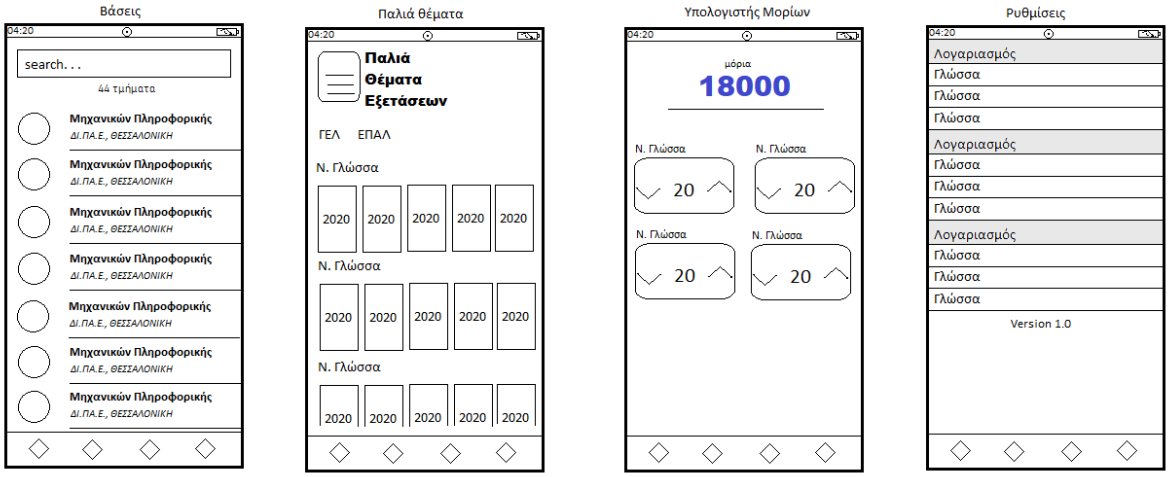
4.3.5 Διαθεσιμότητα ίντερνετ

Τελευταίο, ερώτημα που ζητήθηκε να απαντήσουν οι χρήστες ήταν περί της διαθεσιμότητας τους στο διαδίκτυο. Πιο συγκεκριμένα, ζητήθηκε να απαντήσουν πόσο συχνά δεν έχουν διαθέσιμη πρόσβαση στο διαδίκτυο από το κινητό τους για τον οποιοδήποτε λόγο. Σύμφωνα με την έρευνα τα αποτελέσματα ήταν ότι το 19,8% των χρηστών δεν τους τυχαίνει ποτέ κάτι τέτοιο, το 47,3% τους συμβαίνει πολύ σπάνια, το 7,7% τους συμβαίνει μόλις μερικές φορές τον μήνα, το 9,9% τους συμβαίνει κάθε εβδομάδα, ενώ το 15,4% ήταν το ποσοστό που καθημερινά τυχαίνει να μην έχει πρόσβαση στο διαδίκτυο. Κατά την μελέτη αυτών τον δεδομένων οι πέντε αυτές διαφορετικές ομάδες χωρίστηκαν σε δύο πιο γενικές, στη κατηγορία που αντιμετωπίζει πρόβλημα σύνδεσης στο διαδίκτυο αρκετά συχνά και στη κατηγορία που αντιμετωπίζει αρκετά σπάνια τέτοιο πρόβλημα. Η πρώτη από τις δύο αποτελείται από τις υποκατηγορίες που δεν έχουν πρόσβαση στο διαδίκτυο καθημερινά ή κάθε εβδομάδα, ενώ η δεύτερη κατηγορία αποτελείται από τις υποκατηγορίες που έχουν τέτοιου είδους πρόβλημα κάθε μήνα, σπάνια ή ποτέ. Το αποτέλεσμα των δύο ομάδων είναι σε πολύ μεγάλο βαθμό υπέρ της ομάδας που δεν αντιμετωπίζει συχνά τέτοιας μορφής προβλήματα καθώς το 74,8% των ατόμων που απάντησαν ανήκουν σε αυτήν, ενώ μόλις το 25,2% ανήκει στην δεύτερη κατηγορία. Το ποσοστό αυτό αλλάζει ελάχιστα στους μαθητές καθώς το 30% αυτών ανήκουν στην κατηγορία που δεν έχουν συχνά πρόσβαση στο διαδίκτυο.

Σχήμα 4. 7: Ερώτηση 5

4.4 Επιλογή UI

Όπως προαναφέρθηκε, για τον σχεδιασμό του UI προηγήθηκε ο σχεδιασμός κάποιων πρόχειρων μακετών. Οι μακέτες αυτές, ήταν κάποια προσχέδια για τις τέσσερις κύριες οθόνες της εφαρμογής, των βάσεων, των παλιών θεμάτων, της αριθμομηχανής μορίων και των ρυθμίσεων. Μετά από μελέτη και συζήτηση και σε συνδυασμό με τα αποτελέσματα του ερωτηματολογίου αποφασίστηκε η εφαρμογή να υλοποιηθεί βασισμένη στη μακέτα του Σχήματος 4.8.



Σχήμα 4. 8: Τελική μακέτα

Κεφάλαιο 5ο Ανάλυση Κώδικα της Android εφαρμογής

5.1 Gradle

Στο Gradle πραγματοποιούνται διάφορες ρυθμίσεις σχετικά με το build του προγράμματος όπως η ελάχιστη, μέγιστη και στοχευμένη έκδοση SDK στην οποία θα τρέχει η εφαρμογή [5]. Ακόμη μπορούμε να ρυθμίσουμε την έκδοση της JAVA. Ένας ακόμη από τους πιο συνηθισμένους λόγους επεξεργασίας του Gradle είναι για τη προσθήκη κάποιας εξωτερικής βιβλιοθήκης.

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.activity:activity-ktx:1.2.2'

    testImplementation 'junit:junit:4.13.1'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    //navigation component
    def nav_version = '2.3.2'
    implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
    implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
    androidTestImplementation "androidx.navigation:navigation-testing:$nav_version"

    //material
    def material_version = '1.3.0-beta01'
    implementation "com.google.android.material:material:$material_version"
```

Σχήμα 5.1: Gradle, προσθήκη βιβλιοθήκης

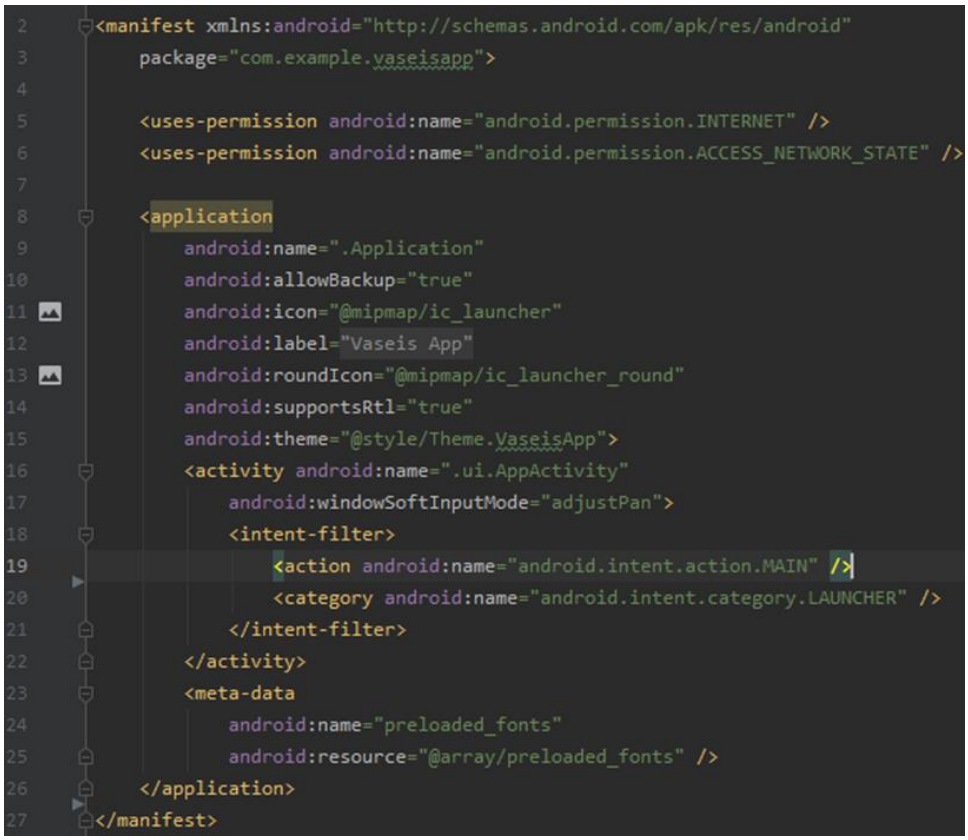
Όπως φαίνεται και στο Σχήμα 5.1 η προσθήκη εξωτερικής βιβλιοθήκης είναι αρκετά εύκολη. Αρκεί να προστεθεί η εντολή implementation ακολουθημένη από το πακέτο της βιβλιοθήκης και την επιθυμητή έκδοση. Μέσω του Gradle ορίζεται, επίσης και το αρχείο του ProGuard το οποίο θα χρησιμοποιηθεί για την release έκδοση της εφαρμογής.

```
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}
```

Σχήμα 5.2: ProGuard

5.2 AndroidManifest

Το AndroidManifest είναι ένα αρχείο που χρειάζεται κάθε Android εφαρμογή [35]. Περιέχει σημαντικές πληροφορίες για την εφαρμογή όπως το όνομα και το εικονίδιο της. Περιέχει ονομαστικά όλα τα activities του project και ορίζει ποιο είναι το αρχικό activity όταν ανοίγει η εφαρμογή. Άλλες σημαντικές πληροφορίες που περιέχει το αρχείο αυτό είναι οι άδειες που χρειάζεται η εφαρμογή.



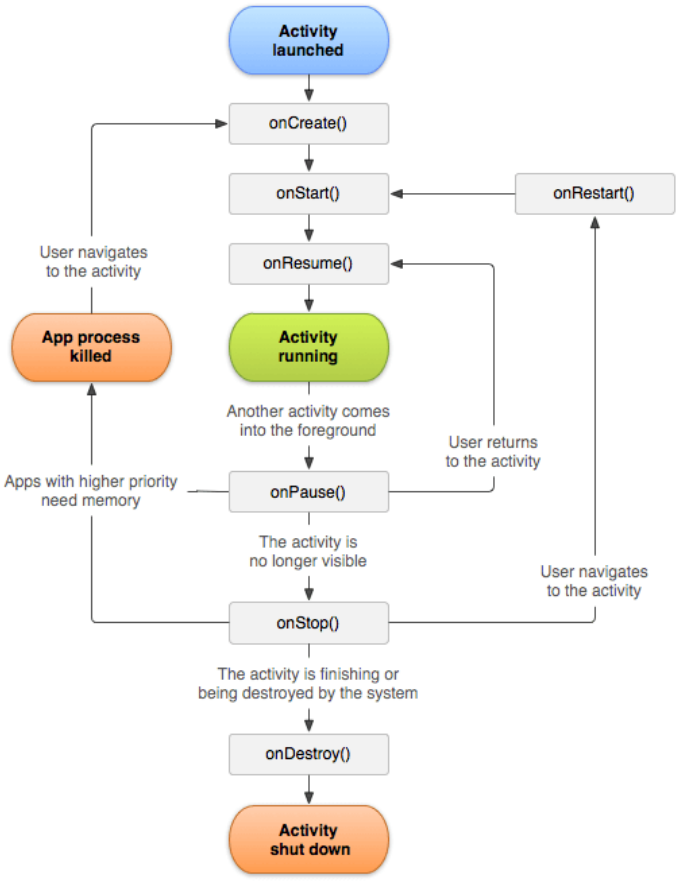
Σχήμα 5.3: AndroidManifest

Το Android για λόγους ασφαλείας απαιτεί συγκεκριμένες άδειες για τη χρησιμοποίηση διάφορων δυνατοτήτων του. Τέτοιες άδειες αφορούν, τη χρήση διαδικτύου, κάμερας, μικροφώνου κ.α. Στη συγκεκριμένη εφαρμογή χρειάστηκαν οι άδειες μόνο για τη χρήστη του διαδικτύου οι οποίες φαίνονται στο Σχήμα 5.3 στις σειρές 5 και 6.

5.3 Activity

Η κλάση Activity είναι ένα κρίσιμο συστατικό μιας εφαρμογής Android και ο τρόπος με τον οποίο ξεκινούν και συνδυάζονται τα Activities αποτελούν θεμελιώδες μέρος του μοντέλου εφαρμογής της πλατφόρμας [36].

Σε αντίθεση με άλλα προγραμματιστικά παραδείγματα στα οποία οι εφαρμογές ξεκινούν με μια κύρια μέθοδο, μία εφαρμογή Android ξεκινά τον κώδικα με ένα Activity. Σε μια εφαρμογή μπορούν να υπάρχουν πολλά Activities, το Activity το οποίο θα είναι το αρχικό ορίζεται από το AndroidManifest και εξηγήθηκε σε προηγούμενο κεφάλαιο. Παρόλα αυτά στη συγκεκριμένη εφαρμογή υπάρχει μόνο ένα, το κυρίως, Activity το οποίο ονομάζεται AppActivity. Το σύστημα του Android ξεκινάει την κλάση Activity καλώντας συγκεκριμένες μεθόδους που αντιστοιχούν σε συγκεκριμένα στάδια της ζωής (lifecycle) του [37]. Το lifecycle αυτό φαίνεται παρακάτω στο Σχήμα 5.6.



Σχήμα 5.6: Activity Lifecycle

5.4 Fragments

Το Fragment αντιπροσωπεύει ένα επαναχρησιμοποιήσιμο τμήμα της διεπαφής χρήστη (UI) της εφαρμογής [38]. Επιπλέον, καθορίζει και διαχειρίζεται το δικό του UI, έχει τον δικό του κύκλο ζωής (lifecycle) [39] και μπορεί να χειριστεί δικά του συμβάντα εισόδου. Τα Fragments δεν μπορούν να υπάρξουν μόνα τους - πρέπει να «φιλοξενούνται» από ένα Activity ή από ένα άλλο Fragment.

Πιο συγκεκριμένα, στην εφαρμογή, το UI του AppActivity περιέχει μόνο το FragmentContainerView, του οποίου η δουλεία είναι να φιλοξενεί τα διάφορα Fragments της εφαρμογής.

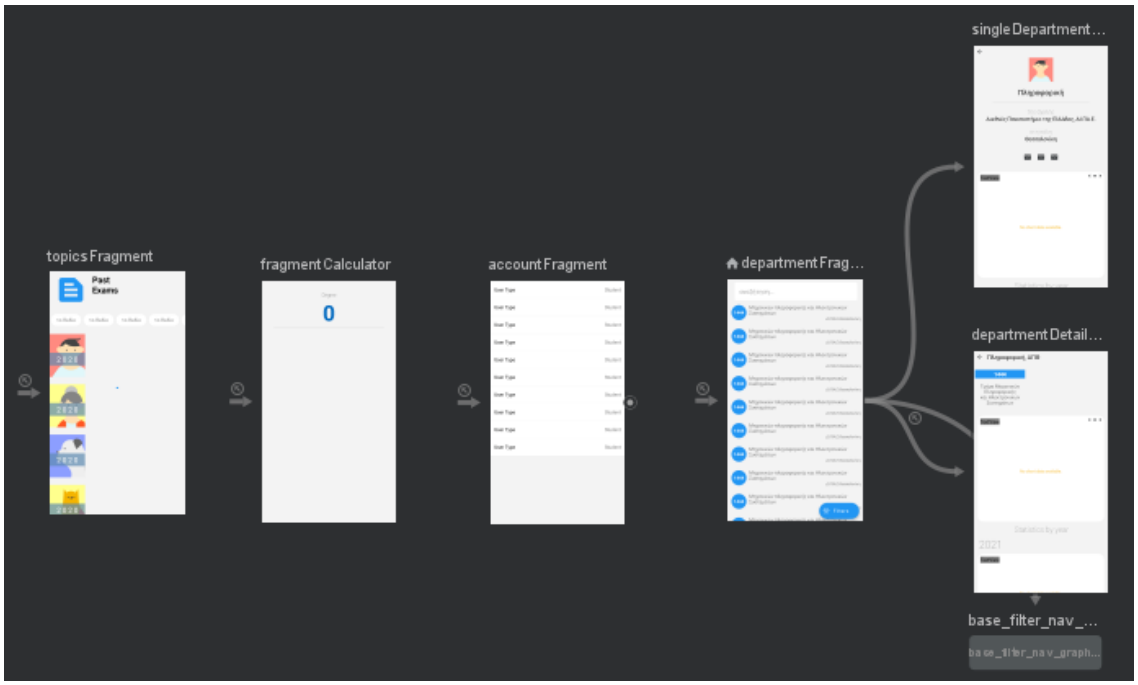
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_scrollFlags="scroll|enterAlways"
    tools:context=".ui.AppActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/app_nav_graph" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Σχήμα 5.7: AppActivity UI

5.5 Navigation Component

Το Navigation Component αναφέρεται στις αλληλεπιδράσεις που επιτρέπουν στους χρήστες να πλοηγούνται σε διάφορα σημεία περιεχομένου, από το ένα Fragment στο άλλο, μιας εφαρμογής [40]. Είναι ένα σημαντικό εργαλείο για την εύκολη διασύνδεση των Fragments. Επίσης, διασφαλίζει μια συνεπή και προβλέψιμη εμπειρία χρήσης ακολουθώντας ένα καθιερωμένο σύνολο κανόνων. Υλοποιείτε από τρία βασικά μέρη, το Navigation Graph, το NavHost και το NavController.

Το Navigation Graph, είναι ένα XML αρχείο που περιέχει όλες τις πληροφορίες που σχετίζονται με την πλοήγηση. Περιλαμβάνει Fragments τα οποία ονομάζονται προορισμοί (destinations), καθώς και τις πιθανές διασυνδέσεις μεταξύ τους (actions). Βέβαια εκτός από Fragments καθιστάτε δυνατή και η προσθήκη άλλων εμφωλευμένων Nav Graphs. Είναι αναγκαίο, επίσης, να οριστεί αρχικός προορισμός. Μέσω του Navigation Graph είναι εφικτό να προσθέσουμε κι άλλες λεπτομέρειες που αφορούν την πλοήγηση ανάμεσα σε Fragments όπως το επιθυμητό animation κατά την μετάβαση από ένα Fragment σε ένα άλλο. Παράλληλα και με βάση το XML αρχείο το Android Studio προσφέρει ένα drag and drop περιβάλλον για πιο εύχρηστη επεξεργασία. Στο σχήμα 5.8 φαίνεται το βασικό NavGraph της εφαρμογής μέσω αυτού του περιβάλλοντος.



Σχήμα 5.8: Dashboard Navigation Graph

Το NavHost είναι ένα κενό κοντέινερ που εμφανίζει τους προορισμούς από το Navigation Graph. Όπως φαίνεται και στο Σχήμα 5.7 αρκεί ο ορισμός των πεδίων name με το NavHostFragment και navGraph με το επιθυμητό Nav Graph.

Τέλος, το NavController είναι το αντικείμενο το οποίο δίνει τη πρόσβαση του NavGraph προγραμματιστικά. Έτσι είναι εφικτή η εύκολη διαχείριση των μεταβάσεων ανάμεσα στα Fragments. Στο Σχήμα 5.9 φαίνεται χαρακτηριστικά η διαδικασία μετάβασης από το τρέχον Fragment σε ένα άλλο, μέσω μίας διασύνδεσης (action). Η κλάση DepartmentFragmentDirections είναι τύπου NavDirections και είναι μια κλάση που δημιουργείτε αυτόματα για κάθε Fragment ενός NavGraph που περιέχει actions. Έτσι δίνεται πρόσβαση σε όλα τα actions αυτά.

```
val action = DepartmentFragmentDirections.actionDepartmentToSingleDepartmentDetails(code, name)
findNavController().safeNavigate(action, R.id.departmentFragment)
```

Σχήμα 5.9: Μετάβαση σε Fragment με τη χρήση NavController

5.6 View Binding

Το View Binding είναι μια δυνατότητα που επιτρέπει την εύκολη αλληλεπίδραση του κώδικα με τα αντικείμενα του UI (views) μίας διάταξης UI (layout) [41]. Είναι η εξέλιξη και ένας πιο αποτελεσματικός τρόπος εύρεσης των views σε σχέση με το findViewById. Ένα View Binding που αναφέρετε σε ένα layout περιέχει άμεσες αναφορές σε όλα τα views του, που έχουν ένα αναγνωριστικό (id), και στα γνωρίσματα (attributes) τους.

Στα πλαίσια του View Binding, και για την αποφυγή επαναληπτικού κώδικα δημιουργήθηκε μία abstract κλάση BaseFragment η οποία επεκτείνει την κλάση Fragment.

```
15  abstract class BaseFragment<VB : ViewBinding> : Fragment() {
16
17      private var _binding: VB? = null
18      protected val binding get() = _binding!!
19
20      abstract fun getViewBinding(): VB
21
22      override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
23          _binding = getViewBinding()
24          return _binding?.root
25      }
26
27      override fun onDestroyView() {
28          _binding = null
29          super.onDestroyView()
30      }
31  }
```

Σχήμα 5.10: BaseFragment

Όλα τα Fragment αντί να είναι τύπου Fragment είναι τύπου BaseFragment. Έτσι είναι αναγκαία πάντα η συμπλήρωση της μεθόδου getViewBinding() όπως φαίνεται και στο Σχήμα 5.11 στη σειρά 18.

```
13  @AndroidEntryPoint
14  class SplashFragment : BaseFragment<FragmentSplashLayoutBinding>() {
15
16      private val viewModel : SplashViewModel by viewModels()
17
18      override fun getViewBinding(): FragmentSplashLayoutBinding  = FragmentSplashLayoutBinding.inflate(layoutInflater)
19
20      override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
21          super.onViewCreated(view, savedInstanceState)
22
23          setupObservers()
24      }
```

Σχήμα 5.11: Splash Fragment

Έπειτα υπάρχει πρόσβαση σε όλα τα views του layout, μέσω της μεταβλητής binding. Η κλάση FragmentSplashLayoutBinding είναι τύπου ViewBinding, δημιουργήθηκε αυτόματα και βασίζεται στο XML αρχείο του layout. Είναι σημαντικό να τονιστεί ότι στα Fragments υπάρχει η δυνατότητα πρόσβασης των views από τη μέθοδο onViewCreated() και όχι από τη μέθοδο onCreate() που είναι πιο πρώιμη στον κύκλο ζωής του Fragment.

5.7 Bottom Navigation

Το Bottom Navigation αντιπροσωπεύει μια τυπική γραμμή πλοήγησης (toolbar) για μία εφαρμογή στο κάτω μέρος της οθόνης [42]. Το Bottom Navigation διευκολύνει τους χρήστες να εξερευνήσουν και να περιηγηθούν στην εφαρμογή μεταξύ των κυρίων κατηγοριών με ένα μόνο πάτημα. Από τα guidelines της Google προτείνεται να χρησιμοποιούνται όταν μια εφαρμογή έχει τρεις έως πέντε κεντρικούς προορισμούς. Στην εφαρμογή έχουμε τέσσερις τέτοιους προορισμούς. Τις βάσεις για κάθε τμήμα, τα παλιά θέματα, τον υπολογιστή μορίων και τις ρυθμίσεις/λογαριασμός. Το τελικό αποτέλεσμα του Bottom Navigation της εφαρμογής φαίνεται στο Σχήμα 5.12

Σχήμα 5.12: Bottom Navigation UI

Ο κάθε κεντρικός προορισμός της εφαρμογής αντιπροσωπεύεται από το αντίστοιχο αντικείμενο (item). Ο τρόπος προσθήκης items στο Bottom Navigation είναι ο ίδιος με ένα Toolbar. Δημιουργούμε, δηλαδή, ένα μενού το οποίο στη συνέχεια το συσχετίζουμε με το Bottom Navigation. Η συσχέτιση αυτή φαίνεται στο Σχήμα 5.13 στην σειρά 26.

```
19    <com.google.android.material.bottomnavigation.BottomNavigationView
20        android:id="@+id/bottom_navigation"
21        android:layout_width="0dp"
22        android:layout_height="wrap_content"
23        app:layout_constraintBottom_toBottomOf="parent"
24        app:layout_constraintStart_toStartOf="parent"
25        app:layout_constraintEnd_toEndOf="parent"
26        app:menu="@menu/dashboard_bottom_nav_menu" />
```

Σχήμα 5.13: Κώδικας του Bottom Navigation

5.8 Διαδικασία επίτευξης Request

Παρακάτω θα αναλυθεί η τρόπος λήψης δεδομένων από το API με τη βοήθεια ενός παραδείγματος για καλύτερη κατανόηση. Θα παρουσιαστεί ολόκληρη η διαδικασία, με βήματα και τα αντίστοιχα παραδείγματα κώδικα για κάθε βήμα. Παράλληλα θα γίνετε επεξήγηση σημαντικών σημείων του κώδικα.

5.8.1 Fragment

Αρχικά πρέπει να δηλωθεί και να δημιουργηθεί ένα instance για το ViewModels. Λόγο της χρήσης του Hilt αυτό γίνετε πολύ εύκολα με μία γραμμή κώδικα. Υπάρχουν τρεις τρόποι να δημιουργήσεις το instance του View Model.

Ο πρώτος είναι με το by ViewModels() το οποίο δημιουργεί μία παρουσία (instance) μόνο για το ίδιο το Fragment. Ο δεύτερος τρόπος είναι με το by ActivityViewModels(). Σε αυτή τη περίπτωση το instance που έχει δημιουργηθεί είναι το ίδιο όχι μόνο για το Fragment, αλλά και για το Activity και για οποιοδήποτε άλλο Fragment δημιουργείτε από το ίδιο Activity και καλεί το View Model με τον ίδιο τρόπο. Αφενός, το θετικό σε αυτή τη περίπτωση είναι ότι δύο ή περισσότερα Fragments και το Activity μπορούν να ανταλλάξουν πληροφορίες μεταξύ τους καθώς όλα βλέπουν και επεξεργάζονται το ίδιοView Model και κατ’ επέκταση τις ίδιες μεταβλητές. Αφετέρου, το συγκεκριμένο instance δε θα καταστραφεί όταν καταστραφεί το Fragment που το κάλεσε, αλλά μόνο όταν καταστραφεί το Activity, το οποίο πολλές φορές μπορεί να οδηγήσει σε αυξημένη χρήση μνήμης χωρίς αιτία.

```
31      private val viewModel: TopicsViewModel by viewModels()
32      //private val viewModel: TopicsViewModel by activityViewModels()
33      //private val viewModel: TopicsViewModel by navGraphViewModels(R.id.main_nav_graph)
```

Σχήμα 5.14:

Η επικοινωνία του Fragment, αλλά και άλλων στοιχείων της εφαρμογής όπως Activities και υπηρεσίες, με το ViewModel πραγματοποιείται μέσω ενός τύπου παρατηρίσημων (observable) δεδομένων, τα LiveData [43]. Σκοπός των LiveData είναι να ενημερώνουν τους παρατηρητές (observers) οι οποίοι έχουν κάνει εγγραφή πάνω τους όταν υπάρξει κάποια ενημέρωση στα δεδομένα τους. Σε αντίθεση με άλλους τύπους τέτοιον δεδομένων τα LiveData γνωρίζουν και σέβονται τους κύκλους ζωής άλλων στοιχείων της εφαρμογής. Αυτό σημαίνει πως τα LiveData ενημερώνουν μόνο observers οι οποίοι ανήκουν σε στοιχεία της εφαρμογής τα οποία βρίσκονται σε κατάσταση ενεργού κύκλου ζωής (lifecycle).

Το LiveData θεωρεί ότι ένας παρατηρητής βρίσκεται σε ενεργή κατάσταση εάν ο κύκλος ζωής του βρίσκεται στην κατάσταση “STARTED” ή “RESUMED”. Το LiveData ειδοποιεί τους ενεργούς παρατηρητές μόνο για ενημερώσεις. Αντίθετα, οι ανενεργοί observers που έχουν εγγραφεί για να παρακολουθούν αντικείμενα LiveData δεν ειδοποιούνται για αλλαγές. Το πολύ σημαντικό πλεονέκτημα αυτού είναι ότι αποφεύγονται “Memory Leaks” δηλαδή χρήση της μνήμης χωρίς αιτία, καθώς και διάφορα σφάλματα λόγο στοιχείων που είναι ανενεργά.

Επιπρόσθετα, όταν το Fragment ή γενικότερα το στοιχείο που ανήκει ο observer γίνει ξανά ενεργό θα λάβει τις τελευταίες ενημερώσεις του LiveData.

```
topicLesson.observe(viewLifecycleOwner, { lessons ->
    binding.progressCircular.visibility = View.GONE

    fillData(lessons)
})

loadExamTypes()
```

Σχήμα 5.15:

Η σχέση αυτή με τον κύκλο ζωής επιτυγχάνεται μέσω ενός αντικειμένου που αντιπροσωπεύει τον κύκλο αυτό και φαίνεται στο Σχήμα 5.15 από το αντικείμενο viewLifecycleOwner.

Τέλος, αφού έχουν ολοκληρωθεί οι εγγραφές των observers στα επιθυμητά LiveData του ViewModel, απλά καλείτε η κατάλληλη μέθοδος του ViewModel η οποία θα ενημερώσει τα LiveData αυτά με τα κατάλληλα δεδομένα.

5.8.2 ViewModel

Το ViewModel με τη σειρά του θα καλέσει το αντίστοιχο Use Case το οποίο στη συγκεκριμένη περίπτωση είναι το getTopicsByExamType(). Βέβαια, μία μέθοδος που επιστρέφει αποτέλεσμα από το ίντερνετ δεν μπορεί να εκτελεστεί στο κεντρικό νήμα, δηλαδή το UI thread. Έτσι το Use Case καλείτε με τη βοήθεια των Coroutines.

```
28      private var _topicLessons = MutableLiveData<List<TopicLesson>>()
29      val topicLesson: LiveData<List<TopicLesson>> = _topicLessons
30
31      fun loadTopicsByExamType(examTypeId: String)  {
32          viewModelScope.launch(Dispatchers.IO){  this: CoroutineScope
33              val result = getTopicsByExamTypeUseCase(examTypeId)
34              _topicLessons.value = result
35          }
36      }
```

Σχήμα5.16: Κάλεσμα του UseCase από το ViewModel

Σχολιάζοντας τον κώδικα στο Σχήμα 5.16, το viewModelScope είναι ένα προκαθορισμένο CoroutineScope που περιλαμβάνεται στις επεκτάσεις του ViewModel KTX, δηλαδή της βιβλιοθήκης του ViewModel που αφορά την Kotlin [44]. Ένα CoroutineScope διαχειρίζεται ένα ή περισσότερα Coroutines. Το launch είναι μια συνάρτηση που δημιουργεί ένα Coroutine και αποστέλλει την εκτέλεση του σώματος λειτουργίας της στον αντίστοιχο διεκπεραιωτή. Ενώ, το Dispatchers.IO υποδεικνύει ότι το συγκεκριμένο Coroutine πρέπει να εκτελεστεί σε ένα νήμα (thread) που προορίζεται για λειτουργίες εισόδου / εξόδου και όχι στο κεντρικό νήμα του UI.

Αξίζει να σημειωθεί ότι για να πετύχει το Android το επιθυμητό αποτέλεσμα των 60 FPS (frames per second) είναι αναγκαίο να δεσμεύει το UI Thread κάθε 17ms (milliseconds). Ως εκ τούτου, σε περίπτωση που το UI Thread είναι απασχολημένο για παραπάνω από 17 ms τότε το frame, δηλαδή η εικόνα θα χαθεί. Αυτό έχει ως αποτέλεσμα να κάνει την εφαρμογή να δείχνει ότι «κολλάει» καθώς θα υπάρχουν χαμένα frames.

5.8.3 Use Case

Το Use Case, όπως προαναφέρθηκε, δεν έχει κάποιο λειτουργικό κώδικα. Περιγράφει μία ενέργεια του χρήστη. Πρακτικά καλεί την αντίστοιχη μέθοδο του DataSource.

```
7   class GetTopicsByExamType @Inject constructor(private val dataSource: TopicsDataSource) {
8       suspend operator fun invoke(id: String): List<TopicLesson> {
9           return dataSource.getExamTypeTopics(id)
10      }
11  }
```

Σχήμα 5.17:

5.8.4 Data Source

Το DataSource με τη σειρά του καλεί τη αντίστοιχη μέθοδο του Repository. Μία από τις πολύ σημαντικές διαδικασίες που πραγματοποιούνται στο DataSource είναι η μετατροπή των αντικειμένων (mapping) από τα Models του Data, σε εσωτερικά Entities.

```
class TopicsDataSourceImpl @Inject constructor(private val repo : TopicsRepository,
private val topicsMapper : TopicsMapper) : TopicsDataSource{
    override suspend fun getExamTypeTopics(id: String): List<TopicLesson> {
        return topicsMapper(repo.getExamTypeTopics(id))
    }
}
```

Σχήμα 5.18:

5.8.5 Repository

Το Repository κατά πρώτο λόγο καλεί το αντίστοιχο ερώτημα από το API interface. Επίσης, είναι το σημείο του κώδικα στο οποίο μπορεί να τοποθετηθεί μία λογική για το αν θα επιστρέφονται τα δεδομένα από το ίντερνετ ή αν θα επιστρέφονται τα τοπικά δεδομένα. Στην εφαρμογή δεν υπάρχει προς το παρόν μια τέτοια λογική αλλά μπορεί εύκολα να προστεθεί χωρίς να χρειαστούν περαιτέρω αλλαγές στον κώδικα σε άλλα σημεία. Σε αυτό το σημείο φαίνεται και ένα από τα μεγάλα κέρδη της χρήσης του Clean Architecture, αλλά και γενικότερα μίας τέτοιας αρχιτεκτονικής, καθώς ο κώδικας διαχωρίζεται και μπορεί εύκολα να βελτιστοποιηθεί.

```
class TopicsRepositoryImpl @Inject constructor(private val api: TopicsApi) : TopicsRepository {
    override suspend fun getExamTypeTopics(id: String): RemoteTopicResponse {
        return api.getExamTypeTopics(id).handleExternalApiFormat()
    }
}
```

Σχήμα 5.19:

5.8.6 API

Το interface του API είναι ένα απλό interface με μεθόδους και δεν χρειάζεται να υλοποιηθεί κάποια κλάση πάνω σε αυτό. Χρησιμοποιώντας την επισήμανση (tag) @GET ή @POST, αναλόγως με τον τύπο του request που θέλουμε να πραγματοποιήσουμε, και συμπληρώνοντας την διεύθυνση του ερωτήματος (endpoint) πραγματοποιείται το request. Αυτό με τη σειρά του θα επιστρέψει τον τύπο που έχουμε ορίσει καθώς το Retrofit, χρησιμοποιώντας το Moshi θα μετατρέψει αυτόματα το JSON αρχείο στα επιθυμητά αντικείμενα και λίστες.

```
interface TopicsApi {
    @GET( value: "/androidapi/api/single_topic.php")
    suspend fun getExamTypeTopics(@Query( value: "id") id : String) : Response<RemoteTopicResponse>
}
```

Σχήμα 5.20:

5.8.7 Module

Όπως μπορεί κάποιος να παρατήρησε, στα προηγούμενα παραδείγματα κώδικα, στα Σχήματα 5.17, 5.18 και 5.19, υπάρχουν μεταβλητές στους constructors των κλάσεων. Οι μεταβλητές αυτές δε χρειάστηκε να δοθούν προγραμματιστικά. Αυτό συμβαίνει λόγο του Hilt που έχει εφαρμοστεί. Το Hilt, όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο είναι ένας αρκετά εύκολος τρόπος να εφαρμοστεί το Dependency Injection στο Android. Ο μοναδικός κώδικας που χρειάστηκε να γραφτεί για τη συγκεκριμένη περίπτωση είναι αυτός του Σχήματος 5.21.

Πιο αναλυτικά η σειρά 24 με 35 αναφέρετε στη δημιουργία του Retrofit και επιστρέφει το interface API. Επίσης σε αυτό το σημείο ορίζεται και η βασική διεύθυνση του API και αυτός είναι ο λόγος που στις μεθόδους των endpoints δεν χρειάστηκε να συμπληρωθεί ολόκληρο το URL, παρά μόνο το τελικό μέρος του. Το tag "@Singleton" ορίζει ότι το interface αυτό έχει ένα instance για όλη την εφαρμογή. Παρακάτω στις σειρές 37 με 46 βρίσκεται το κομμάτι που αφορά το DataSource και το Repository, ενώ για τα Use Cases δεν είναι αναγκαίο να γραφεί επιπλέον κώδικας καθώς το Hilt τα κάνει Inject μόνο του, αρκεί να έχει συμπληρωθεί το κατάλληλο tag πριν τον constructor (βλ. Σχήμα 5.17).

```
20      @Module
21      @InstallIn(ApplicationComponent::class)
22      object TopicsModule {
23
24          @Singleton
25          @Provides
26          fun providePropertiesApi(@BaseHttpClient okHttpClient: OkHttpClient, moshi: Moshi): TopicsApi {
27              val retrofit = Retrofit.Builder()
28                  .baseUrl( baseUrl: "https://vaseis.iee.ihu.gr/")
29                  .client(okHttpClient)
30                  .addConverterFactory(MoshiConverterFactory.create(moshi))
31                  .build()
32
33              return retrofit.create(TopicsApi::class.java)
34          }
35      }
36
37      @Module
38      @InstallIn(ApplicationComponent::class)
39      interface TopicsBindsModule {
40
41          @Binds
42          fun bindTopicsDataSource(dataSource: TopicsDataSourceImpl): TopicsDataSource
43
44          @Binds
45          fun bindTopicsRepository(repository: TopicsRepositoryImpl): TopicsRepository
46      }
```

Σχήμα 5.21:

5.9 RecyclerView και ListAdapter

Το RecyclerView διευκολύνει την αποτελεσματική εμφάνιση μεγάλων συνόλων δεδομένων [45]. Παρέχοντας του τα δεδομένα και το πως θα φαίνεται κάθε στοιχείο η βιβλιοθήκη RecyclerView δημιουργεί δυναμικά τα στοιχεία όταν είναι απαραίτητα. Όπως υποδηλώνει και το όνομα του, το RecyclerView ανακυκλώνει αυτά τα μεμονωμένα στοιχεία. Όταν ένα στοιχείο μετακινείται εκτός οθόνης, το RecyclerView δεν καταστρέφει την προβολή του. Αντ' αυτού, επαναχρησιμοποιεί την προβολή για τα νέα στοιχεία που έχουν μετακινηθεί στην οθόνη. Αυτή η επαναχρησιμοποίηση βελτιώνει σε πολύ μεγάλο βαθμό την απόδοση, βελτιώνοντας την ανταπόκριση της εφαρμογής και μειώνοντας την κατανάλωση ενέργειας. Εκτός από το όνομα της κλάσης, το RecyclerView είναι επίσης το όνομα της βιβλιοθήκης. Παρακάτω όποτε γίνεται αναφορά στο RecyclerView πάνω σε κώδικα εννοείται πάντα η κλάση της βιβλιοθήκη αυτής.

Ο ορισμός των δεδομένων στον RecyclerView πραγματοποιείται μέσω ενός RecyclerView.Adapter. Ο ListAdapter είναι ένας βοηθητικός adapter πάνω στον κλασσικό RecyclerView.Adapter και είναι αυτός που χρησιμοποιείται στην εφαρμογή [46]. Για την μέγιστη κατανόηση αυτών των πολύ σημαντικών

εργαλείων θα αναλυθεί παρακάτω ένα παράδειγμα που υλοποιεί αυτή την διαδικασία. Πιο συγκεκριμένα θα αναλυθεί ο κώδικας που χρησιμοποιείται για την προβολή των ExamTypes.

Όπως φαίνεται και στο Σχήμα 5.22 στη σειρά 14 το ListAdapter δέχεται έναν συγκεκριμένο τύπο δεδομένων, το οποίο σε αυτή την περίπτωση είναι το ExamTypeItem. Επίσης, δέχεται μία κλάση τύπου RecyclerView.ViewHolder η οποία θα αναλυθεί αργότερα και μία παράμετρο τύπου DiffUtil.

```
13  class ExamTypeListAdapter(private val listener: ExamTypeClickListener) :
14      ListAdapter<ExamTypeItem, ExamTypeListAdapter.CategoryViewHolder>(EXAM_TYPE_ITEM_DIFF_UTIL) {
15
16      private var selectedIndex: Int? = null
17
18      interface ExamTypeClickListener {
19          fun onClickListener(item: ExamTypeItem)
20      }
21
22      override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CategoryViewHolder {
23          val layoutInflater = LayoutInflater.from(parent.context)
24          val binding = ItemPersonalizationCategoryBinding.inflate(layoutInflater, parent,  attachToParent: false)
25          return CategoryViewHolder(binding)
26      }
27
28      override fun onBindViewHolder(holder: CategoryViewHolder, position: Int) {
29          holder.bindTo(getItem(position), position)
30      }
```

Σχήμα 5.22: ListAdapter – Μέρος 1/2

Οι μέθοδοι onCreateViewHolder() και onBindViewHolder() είναι οι μέθοδοι που είναι αναγκαίο να υλοποιηθούν. Στην πρώτη από αυτές επιστρέφεται το ViewHolder αφού έχει δημιουργηθεί πρώτα το επιθυμητό binding, το οποίο περνιέται στο ViewHolder ως παράμετρος. Η δεύτερη μέθοδος καλεί την μέθοδο του ViewHolder bindTo() και περνάει ως παράμετρο το αντικείμενο με τις πληροφορίες.

Με τη βοήθεια της μεταβλητής DiffUtil, που δίνεται ως παράμετρος στον ListAdapter, ο RecyclerView γνωρίζει πότε ένα αντικείμενο το οποίο είναι για ανακύκλωση είναι διαφορετικό από το καινούργιο αντικείμενο. Το DiffUtil που χρησιμοποιήθηκε στη συγκεκριμένη περίπτωση φαίνεται στο Σχήμα 5.23. Στο σχήμα αυτό διακρίνεται ότι τα αντικείμενα συγκρίνονται σε δύο διαφορετικούς τομείς. Ανάλογα με το περιεχόμενο τους και ανάλογα με κάποια μεταβλητή που τους χαρακτηρίζει, όπως για παράδειγμα ένα id.

```
65  val EXAM_TYPE_ITEM_DIFF_UTIL = object : DiffUtil.ItemCallback<ExamTypeItem>() {
66      override fun areItemsTheSame(oldItem: ExamTypeItem, newItem: ExamTypeItem): Boolean = oldItem.examType.id == newItem.examType.id
67
68      override fun areContentsTheSame(oldItem: ExamTypeItem, newItem: ExamTypeItem): Boolean = oldItem == newItem
69  }
```

Σχήμα 5.23: DiffUtil

Επίσης, η κλάση τύπου ViewHolder που έχει δοθεί στο ListAdapter υλοποιείται μέσα στον ίδιο τον adapter με μορφή inner class. Η κλάση αυτή είναι υπεύθυνη για την προβολή της επιθυμητής εμφάνισης του αντικειμένου. Στο παρόν παράδειγμα, αυτό συμβαίνει στο Σχήμα 3.20 στη σειρά 32 όπου αφού δοθεί η παράμετρος του επιθυμητού binding το binding.root περνάει στον ViewHolder. Στη συνέχεια υπάρχει η μέθοδος bindTo() η οποία καλείτε από κάθε αντικείμενο που εμφανίζεται για να συμπληρωθούν τα αντίστοιχα δεδομένα.

Τέλος, η υλοποίηση των διάφορων listener που πρέπει να επικοινωνούν με το Fragment από το οποίο καλέστηκε το adapter πραγματοποιείται μέσω interface το οποίο υλοποιείται στο Fragment και δίνετε

στον ListAdapter όταν δημιουργείτε μέσω παραμέτρου. Έπειτα, όπως φαίνεται και στο Σχήμα 3.20 στη σειρά 53 απλά καλείτε η κατάλληλη μέθοδος του interface όπως και όπου χρειάζεται.

```
32    inner class CategoryViewHolder(private val binding: ItemPersonalizationCategoryBinding) : RecyclerView.ViewHolder(binding.root) {
33        fun bindTo(examTypeItem: ExamTypeItem, position: Int) {
34            with(binding) {  this: ItemPersonalizationCategoryBinding
35                examTypeTitleTextView.text = examTypeItem.examType.shortName
36
37                if (examTypeItem.isSelected) {
38                    root.setBackgroundResource(R.drawable.cardview_background_16_selectable)
39                    examTypeTitleTextView.setTextColor(ContextCompat.getColor(root.context, R.color.white_stable))
40                    selectedIndex = position
41                } else {
42                    root.setBackgroundResource(R.drawable.cardview_background_16_white)
43                    examTypeTitleTextView.setTextColor(ContextCompat.getColor(root.context, R.color.text_dr_grey))
44                }
45
46                root.setOnClickListener {  it: View!
47                    selectedIndex?.let { selectedIndex ->
48                        getItem(selectedIndex).isSelected = false
49                    }
50                    examTypeItem.isSelected = true
51                    notifyDataSetChanged()
52
53                    listener.onClickListener(examTypeItem)
54                }
55            }
56        }
57    }
58}
```

Σχήμα 5.24: ListAdapter - Μέρος 2/2

Ένας ακόμα πολύ χρήσιμος adapter χρησιμοποιείται στην εφαρμογή είναι ο ConcatAdapter. Με τον ConcatAdapter γίνεται η παράθεση πολλαπλών adapter σε έναν Recycler. Στο Σχήμα 5.25 στη σειρά 28 φαίνεται η δήλωση ενός ConcatAdapter με δύο διαφορετικούς adapters. Παρόλο που στη συγκεκριμένη περίπτωση η δήλωση των adapter στον ConcatAdapter γίνεται κατά τη αρχικοποίηση του είναι δυνατή και η προσθήκη καινούργιων adapter δυναμικά με το addAdapter(). Ο ConcatAdapter είναι ένα πολύ χρήσιμο εργαλείο για την δημιουργία ενός Recycler με διαφορετικά στοιχεία.

```
26      private val totalAdapter : DepartmentCountAdapter by lazy {DepartmentCountAdapter()}
27      private val departmentAdapter: DepartmentAdapter by lazy { DepartmentAdapter(listener) }
28      private val concatAdapter : ConcatAdapter by lazy { ConcatAdapter(totalAdapter, departmentAdapter) }
```

Σχήμα 5.25: ConcatAdapter

Στο Σχήμα 5.26 φαίνεται το αποτέλεσμα του ConcatAdapter του Σχήματος 5.25. Ο totalAdapter περιέχει ένα μόνο αντικείμενο το οποίο εμφανίζει τα συνολικά τμήματα που εμφανίζονται ενώ ο departmentAdapter εμφανίζει τα τμήματα.

Σχήμα 5.26: Ανάλυση ConcatAdapter σε RecyclerView

5.10 Lazy και Lateinit

Στη παραπάνω παράγραφο, και πιο στη συγκεκριμένα στο Σχήμα 5.25 βρίσκεται το παράδειγμα της αρχικοποίησης τριών adapter. Η αρχικοποίηση αυτή, όπως φαίνεται, πραγματοποιείται απευθείας στην ίδια την δήλωση της μεταβλητής με τη βοήθεια του lazy [47]. Το Lazy δημιουργεί μια νέα παρουσία (instance) που χρησιμοποιεί τον καθορισμένο αρχικοποιητή συνάρτησης αρχικοποίησης. Συνοψίζοντας, δημιουργεί το instance μόνο του δίχως ν χρειαστεί η δημιουργία του μετέπειτα μέσα από κάποια μέθοδο. Παρόμοια με το Lazy, αλλά με διαφορετική λειτουργεία, υπάρχει και η δήλωση lateinit [48]. Με το lateinit ορίζεται ότι μία μεταβλητή θα αρχικοποιηθεί αργότερα. Για τον λόγο αυτό χρειάζεται μεγάλη προσοχή κατά την χρήση του lateinit καθώς σε περίπτωση που καλεστεί πριν από την αρχικοποίηση του θα δημιουργήσει σφάλμα (NullPointerException). Είναι σημαντικό να τονιστεί ότι με κανέναν από τους δύο τρόπους η μεταβλητή δε μπορεί να πάρει null τιμή. Η σημαντική διαφοροποίηση τους είναι ότι το lazy είναι αναγκαστικό να δηλωθεί ως val μεταβλητή, δηλαδή μεταβλητή που της δίνεται η δυνατότητα να αλλάξει αργότερα. Αντίθετα, το lateinit δηλώνεται μόνο ως var μεταβλητή, δηλαδή μεταβλητή που μπορεί να αλλάξει και να οριστεί διαφορετικά. Στη συγκεκριμένη εφαρμογή η αρχικοποίηση των μεταβλητών πραγματοποιείται κατά κύριο λόγο με τη χρήση του lazy.

5.11 Συναρτήσεις Επέκτασης (Extensions)

Η Kotlin παρέχει τη δυνατότητα να επεκτείνει μια τάξη με νέες λειτουργίες χωρίς να χρειάζεται να κληρονομήσει από την κλάση ή να χρησιμοποιήσει μοτίβα σχεδιασμού όπως το Decorator [49]. Αυτό γίνεται μέσω ειδικών δηλώσεων που ονομάζονται επεκτάσεις (extensions). Για παράδειγμα, δίνεται η δυνατότητα για την δημιουργία νέων λειτουργιών για μια κλάση από μια βιβλιοθήκη τρίτων στην οποία δεν υπάρχει η δυνατότητα τροποποίησης. Τέτοιες λειτουργίες είναι διαθέσιμες για κλήσεις με τον συνηθισμένο τρόπο σαν να ήταν μέθοδοι της αρχικής κλάσης. Οι μέθοδοι αυτοί ονομάζονται συναρτήσεις επέκτασης.

```
10  fun Activity.hideSoftKeyboard() {
11      currentFocus?.let {  it: View
12          val inputMethodManager = ContextCompat.getSystemService( context: this, InputMethodManager::class.java)
13          inputMethodManager?.hideSoftInputFromWindow(it.windowToken,  flags: 0)
14      }
15  }
```

Σχήμα 5.27: Συνάρτηση επέκτασης hideSoftKeyboard()

Ένα χαρακτηριστικό επέκτασης τέτοιας συνάρτησης είναι η hideSoftKeyboard(). Στο Σχήμα 5.27 φαίνεται ο κώδικας της συνάρτησης αυτής ο οποίος, όταν καλεστεί, κλείνει το πληκτρολόγιο σε περίπτωση που είναι ανοιχτό. Η hideSoftKeyboard() μπορεί να καλεστεί οπουδήποτε μέσα στο πρόγραμμα μέσω του Activity. Το Σχήμα 5.28 είναι ένα παράδειγμα στο οποίο καλείτε η συνάρτηση αυτή μέσα από ένα fragment. Ο λόγος που υπάρχει η δυνατότητα να καλεστεί η μέθοδος αυτή από το fragment είναι διότι τα fragments έχουν πρόσβαση στο activity που ανήκουν.

```
activity?.hideSoftKeyboard()
```

Σχήμα 5.28: Κάλεσμα συνάρτησης επέκτασης μέσα από fragment

5.12 SharedPreferences

Για την αποθήκευση των ρυθμίσεων και εξατομικεύσεων της εφαρμογής χρησιμοποιήθηκε το SharedPreferences [50]. Το SharedPreferences είναι μία συλλογή κλειδιών-τιμών [51]. Ένα τέτοιο αντικείμενο δείχνει ένα αρχείο που περιέχει τέτοια ζεύγη κλειδιών-τιμών και παρέχει απλές μεθόδους για την ανάγνωσή τους και την εγγραφή τους. Στο Σχήμα 5.29 φαίνεται το Repository των προτιμήσεων, το οποίο χρησιμοποιεί το SharedPreferences και δύο από τις μεθόδους του. Πιο συγκεκριμένα το putSrting για την εγγραφή και το getString για την ανάγνωση τέτοιον τιμών.

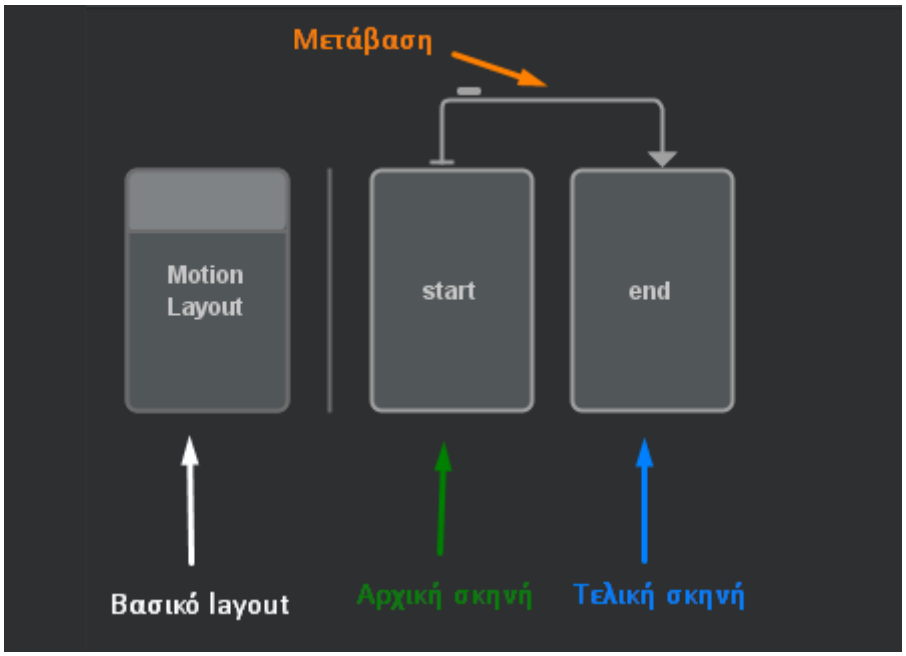
Δύο πολύ σημαντικές προτιμήσεις που αποθηκεύονται με αυτόν τον τρόπο είναι το θέμα της εφαρμογής, και η γλώσσα της. Οι δύο αυτές προτιμήσεις θα αναλυθούν στην επόμενη παράγραφο.

```
class PrefsRepositoryImpl(private val prefs: SharedPreferences) : PrefsRepository {
    override suspend fun setLanguage(lang: String) {
        prefs.edit().putString(LANGUAGE_PREFS, lang).apply()
    }

    override suspend fun getLanguage(): String {
        return prefs.getString(LANGUAGE_PREFS, defValue: "") ?: ""
    }

    override suspend fun setExamType(type: String) {
        prefs.edit().putString(EXAMS_TYPE_PREFS, type).apply()
    }

    override suspend fun getExamType(): String {
        return prefs.getString(EXAMS_TYPE_PREFS, defValue: "") ?: ""
    }

    override suspend fun setGroupType(type: String) {
        prefs.edit().putString(GROUP_TYPE_PREFS, type).apply()
    }

    override suspend fun getGroupType(): String {
        return prefs.getString(GROUP_TYPE_PREFS, defValue: "") ?: ""
    }

    override suspend fun setTheme(theme: String) {
        prefs.edit().putString(THEME_PREFS, theme).apply()
    }

    override suspend fun getTheme(): String {
        return prefs.getString(THEME_PREFS, defValue: "") ?: ""
    }
}
```

Σχήμα 5.29: Shared Preferences

5.13 MotionLayout

Ο κύριος τρόπος που υλοποιούνται οι διάφορες γραφικές κινήσεις στην εφαρμογή πραγματοποιείται με τη βοήθεια του Motion Layout [52]. Το MotionLayout είναι ένας τύπος διάταξης (layout) που βοηθά στην διαχείριση της κίνησης και του γραφικού στοιχείου στην εφαρμογή. Πρόκειται για μια υποκατηγορία του ConstraintLayout και βασίζεται στις δυνατότητες διάταξης του. Ως μέρος της βιβλιοθήκης ConstraintLayout, το MotionLayout διατίθεται ως βιβλιοθήκη υποστήριξης και είναι συμβατό από το API 14 και πάνω. Γεφυρώνει το χάσμα μεταξύ μεταβάσεων διάταξης και σύνθετου χειρισμού κίνησης, προσφέροντας ένα συνδυασμό χαρακτηριστικών μεταξύ του Animation Framework, του TransitionManager και του CoordinatorLayout. Εκτός από την περιγραφή των μεταβάσεων μεταξύ διαφορετικών layout, το MotionLayout επιτρέπει την κίνηση οποιεσδήποτε ιδιότητας. Αναλυτικότερα, υπάρχει η δυνατότητα κινηματογραφικής (animated) αλλαγής της θέσης, του μεγέθους, της ορατότητας, του χρώματος, του άξονα Z (άξονας βάθους), της περιστροφής και πολλών άλλων χαρακτηριστικών οποιουδήποτε view του layout. Με τον συνδυασμό όλων αυτόν των χαρακτηριστικών μπορούν ν κατασκευαστούν από απλές έως πολύπλοκες γραφικές κινήσεις.



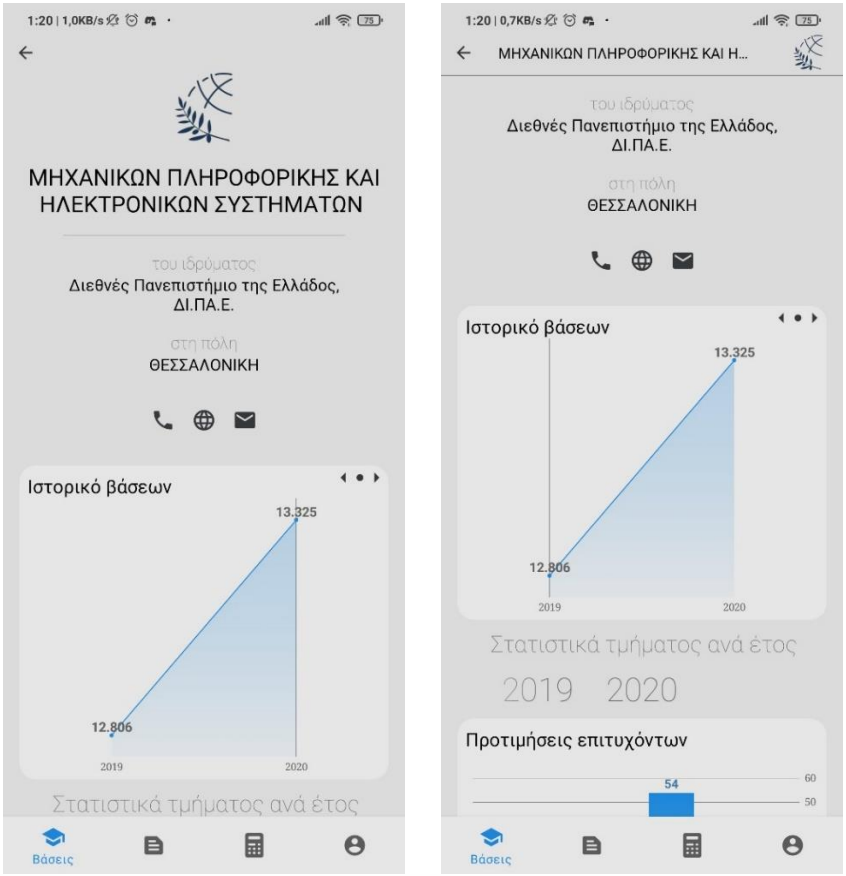
Σχήμα 5.30:MotionLayout

Η λειτουργία του MotionLayout πραγματοποιείται με το βασικό layout και την προσθήκη διάφορων σκηνών και μεταβάσεων. Στο MotionLayout του Σχήματος 5.30 υπάρχον δύο τέτοιες σκηνές. Η αρχική και η τελική. Στις σκηνές αυτές ορίζονται τα διάφορα γνωρίσματα (π.χ. θέση) που θα έχει κάθε view της διάταξης στην αρχή αλλά και στο τέλος της γραφικής κίνησης. Οι σκηνές αυτές συνδέονται μεταξύ τους μέσω μίας μετάβασης (transition). Το Σχήμα 5.31 δείχνει το αρχικό μέρος του κώδικα της μετάβασης. Ο κώδικας αυτός ορίζει την συνολική διάρκεια της μετάβασης, την σκηνή από την οποία θα αρχίσει και την σκηνή στην οποία θα καταλήξει, καθώς και ότι ενεργοποιείται μετακινώντας το scrollView προς τα κάτω.

```
129     <Transition
130         motion:autoTransition="animateToStart"
131         motion:constraintSetEnd="@+id/end"
132         motion:constraintSetStart="@+id/start"
133         motion:duration="1000">
134         <OnSwipe
135             motion:dragDirection="dragUp"
136             motion:touchAnchorId="@+id/single_department_scroll_view"
137             motion:touchAnchorSide="top" />
```

Σχήμα 5.31: MotionLayout – Μετάβαση

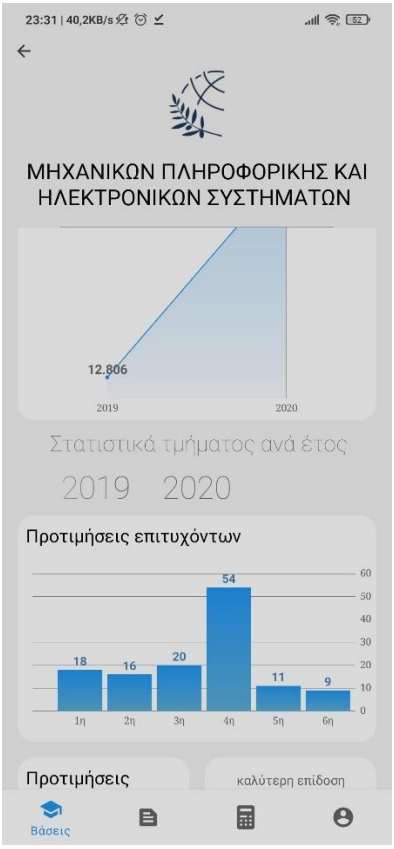
Στο Σχήμα 5.32 φαίνεται η επίδραση του Motion Layout και του κώδικα που αναλύθηκε προηγουμένως. Αριστερά είναι η αρχική σκηνή, καθώς ο χρήστης κάνει scroll προς τα κάτω θα ενεργοποιηθεί η γραφική μετάβαση μέχρις ότου καταλήξει στην σκηνή 2 η οποία είναι το δεξιό μέρος του Σχήματος 5.32. Το αποτέλεσμα εκτός από το αισθητικό κομμάτι του γραφικού στοιχείου βοηθάει και στην καλύτερη εμπειρία χρήστη. Καθώς, ο τίτλος και η εικόνα μεταφέρονται στο επάνω μέρος της οθόνης και πιο συγκεκριμένα στο toolbar απελευθερώνεται ένα μεγάλο μέρος της οθόνης. Παράλληλα, ο χρήστης μπορεί να συνεχίσει να βλέπει ποιο είναι το τμήμα το οποίο παρουσιάζεται ενώ περιηγείται στα παρακάτω στοιχεία.



Σχήμα 5.32: Motion Layout πριν και μετά\

5.14 Saved Instance State

Με τη ενσωμάτωση του MotionLayout στο συγκεκριμένο Fragment δημιουργήθηκε ένα πρόβλημα με το process death [53]. Όπως φαίνεται και στο Σχήμα 5.33 κατά την επιστροφή στην οθόνη έπειτα από ένα process death σε περίπτωση που ο χρήστης είχε κυλίσει την οθόνη προς τα κάτω, κατά την επιστροφή στην εφαρμογή η οθόνη θα παρέμενε με την κύλιση όμως το Motion Layout θα ήταν στην πρώτη σκηνή καθώς δεν θα είχε αποθηκευμένη την προηγούμενη κατάσταση.



Σχήμα 5.33: Πρόβλημα μετά από process dead

Για την επιδιόρθωση του προβλήματος αυτού χρησιμοποιήθηκε το onSaveInstanceState στο οποίο αποθηκεύεται η τιμή του πεδίου progress του Motion Layout. Το πεδίο progress είναι τύπου float και μέσω αυτού το Motion Layout γνωρίζει τι ακριβώς πρέπει να εμφανίσει. Η μέθοδος onSaveInstanceState ενεργοποιείται πριν την καταστροφή του Fragment, πριν δηλαδή χαθούν τα δεδομένα του.

```
71    override fun onSaveInstanceState(outState: Bundle) {
72        super.onSaveInstanceState(outState)
73
74        outState.putFloat(MOTION_PROGRESS, binding.root.progress)
75    }
```

Σχήμα 5.34: onSaveInstanceState()

Η ανάκτηση της τιμής αυτής κατά την δημιουργία του Fragment γίνεται στην μέθοδο onViewCreated όπου και είναι διαθέσιμο το savedInstanceState και τα Views έχουν δημιουργηθεί. Έτσι απλά ορίζεται progress του MotionLayout στην αποθηκευμένη τιμή και η κατάσταση της οθόνης έχει αποκατασταθεί με επιτυχία.

```
77    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
78        super.onViewCreated(view, savedInstanceState)
79
80        if (savedInstanceState != null) {
81            binding.root.progress = savedInstanceState.getFloat(MOTION_PROGRESS,  defaultValue: 0f)
82        }
```

Σχήμα 5.35: Αξιοποίηση του savedInstanceState

5.15 Θέμα και Γλώσσα

Η εφαρμογή δίνει τη δυνατότητα εναλλαγής σε θέμα και γλώσσα. Πιο συγκεκριμένα στην επιλογή του θέματος υπάρχει το φωτεινό, το σκοτεινό καθώς και η επιλογή για να συμβαδίζει με την προεπιλογή του συστήματος. Αντίστοιχα, στις γλώσσες υπάρχει η επιλογή ελληνικών, αγγλικών και η επιλογή συστήματος. Η επιλογή συστήματος επιλέγει τα ελληνικά σε περίπτωση που η γλώσσα του κινητού είναι ρυθμισμένη στα ελληνικά. Σε οποιαδήποτε διαφορετική περίπτωση επιλέγει τα αγγλικά. Όπως τονίστηκε κα στη προηγούμενη παράγραφο οι προτιμήσεις αυτές αποθηκεύονται με τη χρήση του SharedPreferences.

```
<!-- Dashboard Bottom Nav Menu-->
<string name="basesTitle">Bases</string>
<string name="topicTitle">Topics</string>
<string name="calculatorTitle">Calculator</string>
<string name="accountTitle">Account</string>
```

```
<!-- Dashboard Bottom Nav Menu-->
<string name="basesTitle">Βάσεις</string>
<string name="topicTitle">Θέματα</string>
<string name="calculatorTitle">Μόρια</string>
<string name="accountTitle">Λογαριασμός</string>
```

Σχήμα 5.36: Αποσπάσματα από strings.xml και el/strings.xml

Για την δημιουργία δίγλωσσης εφαρμογής αρκεί να δημιουργηθεί ένα δεύτερο αρχείο strings.xml με τις ελληνικές μεταφράσεις όπως στο Σχήμα 5.36 [54]. Με παρόμοιο τρόπο δημιουργείται και μία εφαρμογή με δύο θέματα. Δημιουργόντας, δηλαδή, από ένα αρχείο theme.xml που αντιστοιχεί στο κάθε θέμα.

Ένα άλλο χαρακτηριστικό των Strings είναι τα Quantity Strings (plurals). Πολλές γλώσσες έχουν διαφορετικούς κανόνες για τη γραμματική συμφωνία με την ποσότητα. Στα Ελληνικά, για παράδειγμα, η ποσότητα 1 είναι μια ειδική περίπτωση. Γράφεται "1 αριθμός", αλλά για οποιαδήποτε άλλη ποσότητα η σωστή γραμματική συμφωνία είναι "n αριθμοί". Αυτή η διάκριση μεταξύ ενικού και πληθυντικού είναι πολύ συνηθισμένη. Επειδή κάποιες γλώσσες κάνουν καλύτερες διακρίσεις το Android υποστηρίζει ένα πιο πλήρες σετ που υποστηρίζεται από τις διακεκτριμένες τιμές μηδέν (zero), ένα (one), δύο (two), λίγα (few), πολλά (many) και άλλα (other).

Οι κανόνες για να αποφασιστεί ποια περίπτωση είναι η κατάλληλη για να χρησιμοποιηθεί για μια δεδομένη γλώσσα και ποσότητα μπορεί να είναι πολύ περίπλοκη διαδικασία. Επομένως το Android παρέχει μεθόδους όπως το getQuantityString() για να την καταλληλότερη επιλογή.

```
83      <plurals name="bases_candidate">
84          <item quantity="one">%1$s\nυποψήφιος</item>
85          <item quantity="other">%1$s\nυποψήφιοι</item>
86      </plurals>
```

Σχήμα 5.37: Quantity Strings (plurals)

Στο Σχήμα 5.37 φαίνεται ο τρόπος σύνταξης ενός plural. Στη συγκεκριμένη περίπτωση υπάρχει ο ποσοτικός διαχωρισμός για την ελληνική γλώσσα ανάμεσα στο ένα και σε όλες τις άλλες περιπτώσεις. Παρακάτω στο Σχήμα 5.38 παρουσιάζεται ο τρόπος με τον οποίον χρησιμοποιείται το συγκεκριμένο plural. Ειδικότερα στην γραμμή 556 καλείται μέσω της resources αναφοράς η μέθοδος getQuantityString() η οποία παίρνει 3 παραμέτρους. Η πρώτη παράμετρος είναι το plural που θα χρησιμοποιηθεί, η δεύτερη παράμετρος είναι η επιθυμητή ποσότητα και η τρίτη παράμετρος είναι η μεταβλητή που αντιστοιχεί στη παράμετρο του String. Οι παράμετροι αυτοί μπορούν να οριστούν σε οποιοδήποτε String ή plural με την μορφή “%1$s”. Ο αριθμός που ακολουθείται από το σύμβολο % αντικατοπτρίζει τη θέση της μεταβλητής που θα χρησιμοποιηθεί σε περίπτωση που υπάρξει εισαγωγή περισσότερων από μίας τέτοιας παραμέτρου. Αντίστοιχα, το γράμμα που ακολουθείται από το σύμβολο $ αντικατοπτρίζει τον τύπο της μεταβλητής. Στη συγκεκριμένη περίπτωση το γράμμα είναι το s το οποίο αντιστοιχεί στο String.

```
556     binding.totalCandidatesTitleTxt.text = resources.getQuantityString(
557         R.plurals.bases_candidate,
558         quantity: stat?.candidatePreferences?.getTotal() ?: 1,
559         ...formatArgs: stat?.candidatePreferences?.getTotal()?.toString() ?: "-"
560     )
```

Σχήμα 5.38: getQuantityString()

5.16 Glide

Όλες οι εικόνες που εμφανίζονταί στην εφαργμογή και προέρχονται από το API μέσω URL εμφανίζονται στα ImageViews μέσω της βιβλιοθήκης Glide. Στο Σχήμα 5.39 φαίνεται ο τρόπος και το συντακτικό με το οποίο γίνεται αυτή η ανάθεση. Στο συγκεκριμένο παράδειγμα υπάρχουν 5 ορίσματα. Το with το οποίο δέχετε σαν παράμετρο είτε το context του layout, είτε το context του ιδιου του ImageView που θα χρησιμοποιηθεί. Το load το οποίο παίρνει ως παράμετρο το URL. Το diskCacheStrategy με το οποίο ορίζεται η στρατηγική με την οποία θα αποθηκεύει τις εικόνες το Glide για να μην τις κατεβάζει σε περίπτωση που ζητηθεί η ίδια εικόνα οπουδήποτε στην εφαρμογή. Υπάρχουν πέντε διαφορετικές τέτοιες στρατιγικές. Το ALL με το οποίο τα δεδομένα αποθηκεύονται τοπικά και πριν και μετά την κωδικοποίηση. Το NONE με το οποίο δεν γίνεται καμία αποθήκευση τοπικά. Το DATA με το οποίο αποθηκεύονται τοπικά τα δεδομένα πριν αποκωδικοποιηθούν. Το RESOURCE με το οποίο, αντίθετα με το DATA, αποθηκεύονται τοπικά τα δεδομένα μόνο αφού αποκωδικοποιηθούν. Τέλος, το AUTOMATIC το οποίο προσπαθεί να εφαρμόσει μία έξυπνη στρατηγική. Έπειτα το όρισμα του CenterInside προσαρμόζει την εικόνα για την επιθυμητή εμφάνιση της. Παρόμοια τέτοια ορίσματα που διατίθενται στο Glide είναι τα CenterCrop, CircleCrop, FitCenter και RoundedCorners. Το τελικό όρισμα που χρησιμοποιείται είναι το into() με το οποίο ορίζεται το View στο οποίο θα τοποθετηθεί η εικόνα. Επιπρόσθετα, ένα ακόμα όρισμα που χρησιμοποιείται συχνά αλλά δεν φαίνεται στο παρακάτω σχήμα είναι αυτό του placeholder() με το οποίο ορίζεται μία

συγκεκριμένη εικόνα η οποία εμφανίζεται σε περίπτωση που δεν είναι εφικτή η φόρτωση της εικόνας από το URL.

```
Glide.with(binding.root.context).load(uniLogo).diskCacheStrategy(DiskCacheStrategy.ALL).centerInside().into(binding.logoImageView)
```

Σχήμα 5.39: Glide

5.17 Επικοινωνία με άλλες εφαρμογές

Μια Android εφαρμογή συνήθως έχει πολλά Activities. Κάθε Activity χρησιμοποιεί ένα UI με το οποίο επιτρέπει στο χρήστη να εκτελεί μια συγκεκριμένη εργασία (π.χ. προβολή βάσεων). Για να μεταφερθεί ο χρήστης από το ένα Activity στο άλλο, η εφαρμογή πρέπει να χρησιμοποιήσει ένα Intent για να καθορίσει μια συγκεκριμένη ενέργεια [55]. Όταν μεταβιβάζεται ένα Intent στο σύστημα με μια μέθοδο όπως το startActivity(), το σύστημα το χρησιμοποιεί για να εντοπίσει και να ξεκινήσει το κατάλληλο στοιχείο της εφαρμογής. Ωστόσο, η χρήση ενός Intent εκτός από την δυνατότητα να ξεκινήσει ένα καινούργιο Activity της εφαρμογής, μπορεί να ξεκινήσει και Activities που περιέχονται σε διαφορετικές εφαρμογές [56]. Η δυνατότητα αυτή του Android να στέλνει τον χρήστη σε μία άλλη εφαρμογή με βάση μια "ενέργεια" που θα ήθελε να εκτελέσει είναι ένα πολύ σημαντικό χαρακτηριστικό του.

Καθώς η εφαρμογή υλοποιήθηκε με ένα μοντέλο ενός Activity (Single-Activity), το Intent χρησιμοποιήθηκε για την επικοινωνία με διαφορετικές εφαρμογές. Για παράδειγμα σε περίπτωση που ο χρήστης επιθυμεί να προβάλει PDF ενός παλιού θέματος θα χρησιμοποιήσει την προεγκατεστημένη εφαρμογή που διαθέτει το κινητό για αυτή τη λειτουργία. Το πλεονέκτημα σε αυτή τη διαδικασία είναι ότι δεν είναι απαραίτητο να υλοποιηθεί εξ ολοκλήρου η λειτουργία αυτή από την ίδια την εφαρμογή. Πέρα από αυτό, είναι σχεδόν σίγουρο ότι η εφαρμογή που έχει σχεδιαστεί ειδικά γι’ αυτόν τον σκοπό (στην περίπτωση του παραδείγματος για την προβολή PDF αρχείων) θα έχει υλοποιήσει και βελτιστοποιήσει σε καλύτερο βαθμό τη λειτουργία του. Στο Σχήμα 5.40 φαίνεται ο κώδικας που χρησιμοποιείται για να καλεστεί η εφαρμογή για την προβολή PDF αρχείου.

```
111             try {
112                 val intent = Intent(Intent.ACTION_VIEW)
113                 intent.setDataAndType(Uri.parse(topic.pdfUrl),  type: "application/pdf")
114                 intent.flags = Intent.FLAG_ACTIVITY_NO_HISTORY
115                 startActivity(intent)
116             } catch (e: ActivityNotFoundException) {
117                 Toast.makeText(context, "Unable to find application to open PDF file.", Toast.LENGTH_SHORT).show()
118                 startActivity(Intent(Intent.ACTION_VIEW, Uri.parse(topic.pdfUrl)))
119             }
```

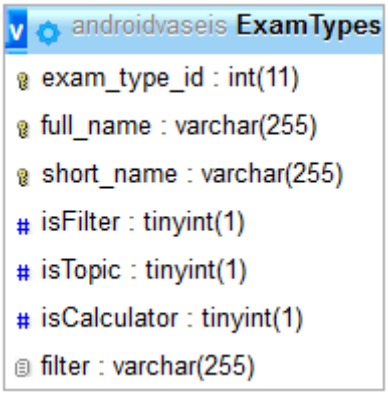
Σχήμα 5. 40: Άνοιγμα PDF αρχείου με εξωτερική εφαρμογή

Πιο συγκεκριμένα στην γραμμή 112 δημιουργείται ένα intent τύπου ACTION_VIEW και στη παρακάτω γραμμή τοποθετούνται τα δεδομένα και ο τύπος τους. Σε αυτή την περίπτωση τα δεδομένα είναι το URL του pdf και ο τύπος το “application/pdf”. Με την χρήση αυτού του τύπου η συσκευή θα αναγνωρίσει το αρχείο ως pdf και θα ψάξει μόνο για εφαρμογές που διαθέτουν την δυνατότητα να το προβάλουν. Είναι σημαντικό να τονιστεί, ότι η ίδια η εφαρμογή δεν έχει καμία γνώση για το ποια είναι και αν υπάρχει τέτοιου είδους εφαρμογή. Για τον λόγο αυτό είναι αναγκαία η χρήση ενός try-catch, καθώς σε περίπτωση που δεν βρεθεί διαθέσιμη εξωτερική εφαρμογή θα δημιουργηθεί το ActivityNotFountException. Στις γραμμές 117,118 διαχειρίζεται αυτή η περίπτωση ξεκινώντας ένα intent δίχως type. Η υλοποίηση αυτή θα στείλει το αρχείο στον browser από τον οποίο ο χρήστης θα έχει την δυνατότητα να κατεβάσει το αρχείο.

Κεφάλαιο 6ο Ανάλυση Κώδικα του Backend

6.1 Βάση Δεδομένων

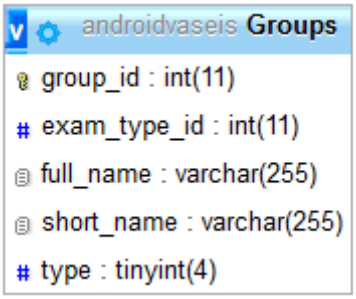
Ο πίνακας ExamTypes περιγράφει τους διάφορους εξεταστικούς τύπους που υπάρχουν όπως το Γενικό Λύκειο και το Επαγγελματικό Λύκειο. Αποτελείται από εφτά πεδία. Το πεδίο exam_type_id είναι το κύριο κλειδί, είναι τύπου int και είναι ένα πεδίο το οποίο η τιμή του παράγεται μόνη της από τη βάση δεδομένων (auto generated). Γενικότερα όποτε γίνεται αναφορά σε κάποιο κύριο κλειδί σε αυτή την βάση θα ισχύουν πάντα οι προηγούμενοι παράμετροι. Έπειτα, υπάρχουν τα πεδία του full_name και του short_name τα οποία είναι τύπου varchar και περιγράφουν το πλήρες όνομα και μία συντομογραφία του τύπου εξετάσεων. Ακολουθούν τα πεδία isFilter, isTopic και isCalculator. Καθώς η εφαρμογή χρειάζεται σε διάφορα σημεία της τους τύπου εξετάσεων (πχ φίλτρα βάσεων, παλιά θέματα κ.α.) δημιουργήθηκε η ανάγκη για περισσότερη ευελιξία στην διαχείριση τους. Για παράδειγμα, ένας τύπος εξέτασης όπως το ΓΕΛ 10% θα είχε νόημα να εμφανίζεται στα φίλτρα βάσεων αλλά όχι στα παλιά θέματα. Με τις τρείς αυτές μεταβλητές γίνεται η διαχείριση αυτή. Όταν το isFilter ισούται με 1 επιστρέφεται ο τύπος βάσης στα φίλτρα, όταν το isTopic ισούται με 1 επιστρέφεται στα παλιά θέματα ενώ όταν το isCalculator ισούται με 1 επιστρέφεται στην αριθμομηχανή μορίων. Φυσικά είναι δυνατός οποιοσδήποτε συνδυασμός αυτών των τιμών. Το τελευταίο πεδίο του πίνακα είναι το filter. Το filter είναι τύπου varchar και περιέχει το αντίστοιχο φίλτρο της τύπου εξέτασης. Αυτό το φίλτρο αναφέρεται στο API των βάσεων. Το πεδίο αυτό είναι σημαντικό καθώς με αυτόν τον τρόπο η εφαρμογή μπορεί να παραμείνει ενημερωμένη , χωρίς να χρειάζεται update παρά την οποιαδήποτε αλλαγή στο API των βάσεων.



Σχήμα 6.1: Πίνακας ExamTypes

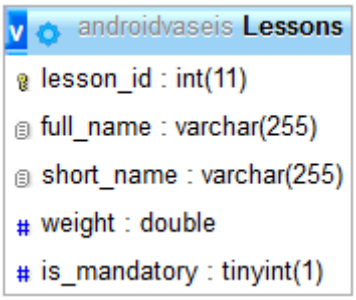
Έπειτα υπάρχει ο πίνακας Groups, ο οποίος αντιπροσωπεύει τις ομάδες προσανατολισμού (ή τις παλιότερες κατευθύνσεις). Αποτελείται από πέντε πεδία, το κύριο κλειδί group_id, το ξένο κλειδί exam_type_id, το full_name και το short_name και το type. Το κύριο κλειδί καθώς και τα πεδία full_name και short_name είναι αντίστοιχα του πίνακα ExamTypes. Το exam_type_id είναι το ξένο κλειδί, αναφέρετε, δηλαδή, σε ένα exam_type_id του πίνακα ExamTypes. Έτσι, οι δύο πίνακες συσχετίζονται και συμπερασματικά φαίνεται ότι το κάθε αντικείμενο τύπου Groups ανήκει σε ένα αντικείμενο ExamTypes. Το έκτο και τελευταίο πεδίο του είναι το type το οποίο είναι τύπου int. Το πεδίο type ενός Group μπορεί να πάρει τέσσερις διαφορετικές τιμές που αντιστοιχούν σε τέσσερις αριθμούς από το 0 έως το 3. Το πεδίο αυτό χαρακτηρίζει τον τύπο του Group και προσφέρει μεγαλύτερη ευελιξία. Η ευελιξία αυτή αναφέρεται στην τρόπο με τον οποίο χρησιμοποιούνται τα Groups. Γενικά, υπάρχουν δύο λόγοι κύριες λειτουργίες που χρησιμοποιούν τα Groups με τα Lessons που ανήκουν σε

αυτό. Η λειτουργία των παλιών θεμάτων και η λειτουργία του υπολογιστή μορίων. Ειδικότερα τώρα, η κατηγορία με τον αριθμό 1 χαρακτηρίζει τα Groups και τα Lessons που ανήκουν σε αυτό ως αντικείμενα που χρησιμοποιούνται μόνο για τη λειτουργία των παλιών θεμάτων. Η κατηγορία με τον αριθμό 2 τα χαρακτηρίζει ως αντικείμενα που χρησιμοποιούνται μόνο για τη λειτουργία του υπολογιστή μορίων. Έπειτα υπάρχουν οι κατηγορίες 0 και 1 οι οποίες είναι για Groups που δεν ανήκουν σε καμία από τις 2 παραπάνω λειτουργίες, ενώ η κατηγορία 3 σηματοδοτεί ότι τα Groups ανήκουν και στις δύο αυτές λειτουργίες.



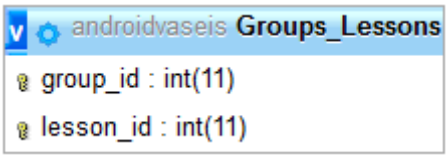
Σχήμα 6.2: Πίνακας Groups

Στη συνέχεια, υπάρχει ο πίνακας Lessons, οποίος περιγράφει τα διάφορα μαθήματα. Ο πίνακας συντελείται από τα πεδία lesson_id που είναι και το κύριο κλειδί του, από το full_name και το short_name τα οποία αφορούν την ονομασία του και από δύο ακόμα πεδία, αυτά του weight και του is_mandatory. Το πεδίο weight είναι τύπου double και περιέχει τον συντελεστή βαρύτητας του εκάστοτε μαθήματος, ενώ το πεδίο is_mandatory αντιπροσωπεύει αν το μάθημα είναι υποχρεωτικό ή όχι, καθώς όπως αναλύθηκε και σε προηγούμενο κεφάλαιο για την εισαγωγή σε συγκεκριμένα τριτοβάθμια τμήματα απαιτείται η εξέταση σε κάποια επιπρόσθετα μαθήματα εκτός από αυτά της ομάδας προσανατολισμού. Ο τύπος tinyint είναι αντίστοιχος και χρησιμοποιείται έναντι του boolean και μπορεί να πάρει τιμές 0 ή 1.



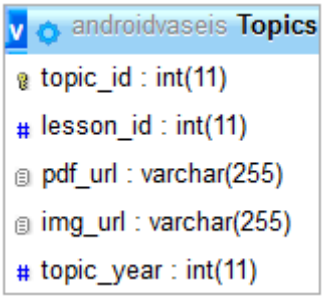
Σχήμα 6.3: Πίνακας Lessons

Παρόλο που στους πίνακες Lessons και Groups δεν υπάρχει κάποιο ξένο κλειδί να τους συνδέει οι δύο αυτοί πίνακες συνδέονται μεταξύ τους. Η σύνδεση αυτή πραγματοποιείται μέσω ενός τρίτου πίνακα και αυτό συμβαίνει διότι είναι σύνδεση «πολλά προς πολλά». Αυτό σημαίνει ότι ένα Lesson μπορεί να ανήκει σε πολλά Groups αλλά επίσης και ένα Group μπορεί να περιέχει πολλά Lessons. Ο τρίτος αυτό πίνακας που πραγματοποιεί την μεταξύ τους διασύνδεση είναι ο Groups_Lessons ο οποίος έχει μόνο δύο πεδία. Το πεδίο group_id το οποίο είναι ξένο κλειδί και αναφέρεται στο group_id του πίνακα Groups και το ξένο κλειδί lesson_id το οποίο, επίσης, ξένο κλειδί και αναφέρετε στο lesson_id του πίνακα Lessons.



Σχήμα 6.4: Πίνακας Groups_Lessons

Τέλος, υπάρχει ο πίνακας Topics ο οποίος αντιπροσωπεύει τα εξεταστέα θέματα από προηγούμενες χρονιές των πανελλήνιων εξετάσεων και αποτελείται από έξι πεδία. Το πεδίο topic_id που είναι το κύριο κλειδί του, το πεδίο lesson_id που είναι ξένο κλειδί, το πεδίο pdf_url που περιέχει ένα κείμενο με τον σύνδεσμο για το αρχείο των θεμάτων, το img_url το οποίο μπορεί να περιέχει κάποια εικόνα που σχετίζεται με το θέμα αυτό και τέλος το πεδίο topic_year το οποίο είναι τύπου int και αντιπροσωπεύει την χρονιά στην οποία ανήκει το συγκεκριμένο θέμα. Ο συγκεκριμένος πίνακας συσχετίζεται με τον πίνακα του μαθήματος με την σχέση ότι ένα θέμα ανήκει σε ένα συγκεκριμένο μάθημα. Η συσχέτιση αυτή πραγματοποιείται μέσω του ξένου κλειδιού lesson_id.



Σχήμα 6.5: Πίνακας Topics

6.2 API

Τα αρχεία του API μπορούν και έχουν διαχωριστεί σε τρεις κατηγορίες (φακέλους), το config , τα models και το api. Στα παραδείγματα κώδικα που θα ακολουθήσουν κάποιες τιμές λείπουν ή είναι αλλαγμένες διότι πρόκειται για ευαίσθητες πληροφορίες (πχ username, κωδικός της βάσης κ.α.).

Αρχικά, το config περιέχει ένα αρχείο AppDatabase.php στο οποίο πραγματοποιείται η σύνδεση με την βάση δεδομένων. Το αρχείο αυτό γίνεται include σε άλλα αρχεία. Για την σύνδεση με την βάση δεδομένων χρησιμοποιείται το PDO, κάτι το οποίο φαίνεται και στο Σχήμα 6.5 στις σειρές 15 με 22.

```php
<?php

class AppDataBase
{
    private $host = '';
    private $db_name = '';
    private $username = '';
    private $passwd = '';

    public function connect()
    {
        $this->conn = null;

        try {
            $this->conn = new PDO(
                dsn: 'mysql:host=' . $this->host . ';dbname=' . $this->db_name, $this->username, $this->passwd,
                array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"
                ));
            $this->conn->setAttribute( attribute: PDO::ATTR_ERRMODE,  value: PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            echo 'Connect Error: ' . $e->getMessage();
        }

        return $this->conn;
    }
}
```

Σχήμα 6.5:

Για την κατανόηση της λειτουργίας ενός ερωτήματος θα αναλυθεί το απλό παράδειγμα που επιστρέφει μία λίστα με όλα τα ExamTypes (πχ ΓΕΛ, ΕΠΑΛ, Εσπερινό ΓΕΛ κ.α.). Την σύνθεση του ερωτήματος απαρτίζουν δύο αρχεία ακόμα. Το αρχείο ExamType.php και το αρχείο examtypes.php. Το πρώτο από αυτά περιέχει την κλάση ExamType και υλοποιεί τον τύπο αντικειμένου που θα επιστρέψει το ερώτημα. Όπως φαίνεται και στο Σχήμα 6.6 το αντικείμενο ExamType έχει τρία πεδία (properties). Επίσης, στις σειρές 19 με 28 υλοποιείται η μέθοδος που διαβάζει από τη βάση τα Exam Types και τα επιστρέφει.

```php
<?php
class ExamType  {
    //DB STAFF
    private $conn;
    private $table = 'type_exams';

    //Properties
    public $lesson_id;
    public $full_name;
    public $short_name;

    //Constructor with DB
    public function __construct($db)
    {
        $this->conn = $db;
    }

    //Get
    public function read()  {
        $query = 'SELECT id, fname,snameFROM ' . $this->table;

        //Prepare statement
        $stmt = $this->conn->prepare($query);
        //Execute statement
        $stmt->execute();

        return $stmt;
    }
}
```

Σχήμα 6.6: ExamType.php

Από την άλλη, το αρχείο examtypes.php ουσιαστικά είναι το ίδιο το ερώτημα καθώς αυτό θα καλέσει ο πελάτης (client). Στο Σχήμα 6.7 αρχικά ορίζονται δύο απαραίτητοι headers για τον ορισμό κάποιον αναγκαίων πληροφοριών. Πιο συγκεκριμένα στη σειρά 4 ορίζεται ο τύπος των δεδομένων που επιστρέφεται ο οποίος είναι json και η κωδικοποίηση η οποία είναι η utf8mb4. Έπειτα γίνονται include τα δύο αρχεία που επεξηγήθηκαν παραπάνω και δημιουργείτε μία σύνδεση με τη βάση η οποία χρησιμοποιείται για να δημιουργηθεί ένα αντικείμενο τύπου ExamType. Κατόπιν, καλείτε η κλάση του αντικειμένου read η οποία θα τρέξει το SQL ερώτημα και θα επιστρέψει το αποτέλεσμα.

```php
<?php
//Headers
header( header: 'Access-Control-Allow-Origin: *');
header( header: 'Content-Type: application/json; charset=utf8mb4');

include_once '../config/AppDataBase.php';
include_once '../models/ExamType.php';

// Instantiate DB and connect
$database = new AppDatabase();
$db = $database->connect();

// Instantiate Exam Type
$examType = new ExamType($db);

// Get ExamTypes
$result = $examType->read();

$num = $result->rowCount();    //Get row count

```

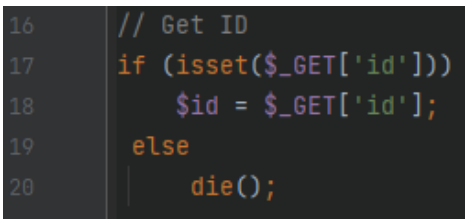
Σχήμα 6.7: examtypes.php – Μέρος ½

Στη συνέχεια, το Σχήμα 6.8 δείχνει τον κώδικα με τον οποίο το αποτέλεσμα που πάρθηκε από τη μέθοδο read του αντικείμενου ExamType θα μεταφραστεί σε json και θα επιστραφεί. Έτσι το ερώτημα θα έχει ολοκληρωθεί και θα έχει δοθεί η ανάλογη απάντηση στον πελάτη.

```php
//Check if there are lessons
if ($num > 0) {
    $exam_type_arr = array();
    $exam_type_arr['exam_types'] = array();

    while ($row = $result->fetch(PDO::FETCH_ASSOC)) {
        /**
         * @var string $exam_type_id
         * @var string $full_name
         * @var string $short_name
         */
        extract( &array: $row);

        $exam_type_item = array(
                'exam_type_id' => $exam_type_id,
                'full_name' => $full_name,
                'short_name' => $short_name,
            );

            //Push to "data"
            array_push( &array: $exam_type_arr['exam_types'], $exam_type_item);
    }

    echo json_encode($exam_type_arr);
} else {
    echo json_encode(array('error' => 'No Lessons found'));
}
```

Σχήμα 6.8: examtypes.php - Μέρος 2/2

Επιπρόσθετα, σε περίπτωση που υπάρχει η ανάγκη κάποιας παραμέτρου στο ερώτημα, όπως κάποιο id για την επιστροφή ενός συγκεκριμένου αποτελέσματος, το ερώτημα θα πραγματοποιηθεί με την χρήση παραμέτρου και τη χρήση prepared statements για την εκτέλεση του SQL κώδικα. Ο τρόπος ανάγνωσης μίας τέτοιας παραμέτρου (πχ ?id=1) σε ένα GET ερώτημα φαίνεται στο Σχήμα 6.9



Σχήμα 6.9: Παράμετρος σε ερώτημα GET

Κεφάλαιο 7ο Παρουσίαση εφαρμογής

Στο κεφάλαιο αυτό θα πραγματοποιηθεί η ενδελεχής παρουσίαση της εφαρμογής. Η εφαρμογή αποτελείται από τις βασικές λειτουργίες, της προβολής των τμημάτων και των βάσεών τους, των παλιών θεμάτων από τις πανελλήνιες εξετάσεις και του υπολογιστή μορίων για τον εύκολο και γρήγορο υπολογισμό τους. Για την απλούστευση της παρουσίασης, η εφαρμογή χωρίστηκε στις τρεις αυτές κατηγορίες και επιπρόσθετα αυτή των ρυθμίσεων που αποτελούν τους κύριους τομείς της εφαρμογής. Στην τελευταία κατηγορία περιλαμβάνονται οι ποικίλες ρυθμίσεις και προτιμήσεις που μπορεί να ορίσει ο χρήστης. Η εναλλαγή αυτών των κατηγοριών επιτυγχάνεται μέσω της μπάρας που υπάρχει στο κάτω μέρος της οθόνης.

7.1 Βάσεις

Η κατηγορία των βάσεων είναι η αρχική οθόνη της εφαρμογής. Σε αυτήν υπάρχει η μπάρα αναζήτησης και μία λίστα με τα τμήματα. Επίσης, στη κάτω δεξιά γωνία υπάρχει το κουμπί των φίλτρων. Όταν ο χρήστης δεν έχει αλλάξει κάποιο φίλτρο τότε εμφανίζονται όλα τα ενεργά τμήματα στη λίστα. Ο χρήστης μπορεί να δει άμεσα για κάθε τμήμα το όνομα του, το ίδρυμα και την πόλη στην οποία βρίσκεται το τμήμα αυτό. Επίσης, στο αριστερό μέρος βρίσκεται η εικόνα του ιδρύματος. Στην κορυφή της λίστας αυτής αναγράφεται ο αριθμός των τμημάτων βάση της τρέχουσας αναζήτησης και των επιλεγμένων φίλτρων. Με την αναζήτηση, η οποία χρησιμοποιεί τα τρία προαναφερόμενα χαρακτηριστικά, ο χρήστης μπορεί να ψάξει συγκεκριμένα τμήματα.

Σχήμα 7.1

Επιπρόσθετα, υπάρχει η δυνατότητα για τον ορισμό κάποιων φίλτρων. Η πρώτη επιλογή που δίνεται είναι η αλλαγή της ταξινόμησης. Έπειτα, υπάρχει το φίλτρο των μη ενεργών τμημάτων. Μέσω αυτού ο χρήστης έχει την δυνατότητα να προσθέσει στην λίστα τμήματα τα οποία δεν είναι πλέον ενεργά,

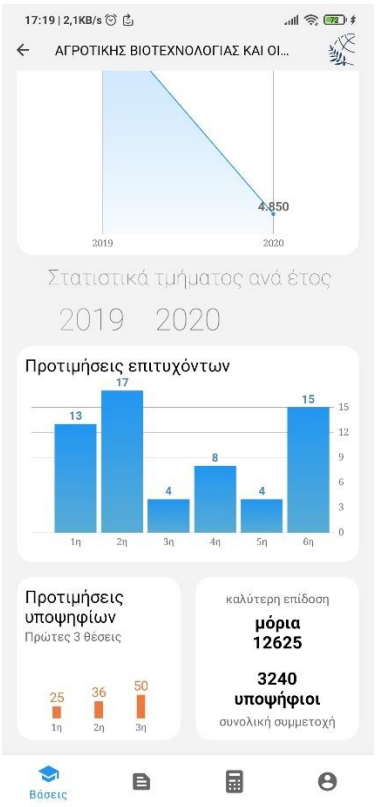
δηλαδή που η λειτουργία τους δεν υφίσταται πλέον. Τρίτο φίλτρο είναι ο περιορισμός της λίστα σε ένα ή περισσότερα πεδία. Το φίλτρο αυτό είναι από τα πιο σημαντικά καθώς οι υποψήφιοι θα έχουν την δυνατότητα να βλέπουν μόνο τα τμήματα στα οποία έχουν την δυνατότητα να περάσουν. Στη συνέχεια, βρίσκεται το φίλτρο των ιδρυμάτων μέσω του οποίο επιλέγονται τμήματα που ανήκουν σε αυτά. Στο κάτω μέρος της οθόνης υπάρχει το κουμπί του καθαρισμού το οποίο κάνει εκκαθάριση οποιουδήποτε φίλτρου έχει τοποθετηθεί ενώ το κουμπί εφαρμογή κάνει οριστική εφαρμογή.

Πατώντας ο χρήστης σε κάποιο συγκεκριμένο τμήμα ανοίγει η οθόνη που φαίνεται στο Σχήμα 7.2 και 7.3. Στην οθόνη αυτή φαίνονται οι βασικές πληροφορίες του επιλεγμένου τμήματος, όπως το όνομα του, το πλήρες όνομα του ιδρύματος που ανήκει καθώς και η συντομογραφία του, η πόλη στην οποία βρίσκεται και τα πεδία από τα οποία έχει πρόσβαση ο εξεταζόμενος. Έπειτα, υπάρχει ένα σχεδιάγραμμα με το ιστορικό των παλιών βάσεων. Σε περίπτωση που οι βάσεις αυτές είναι παραπάνω από πέντε ο χρήστης θα μπορεί να σύρει το διάγραμμα προς τα αριστερά για να δει και τα παλιότερα αποτελέσματα.

Σχήμα 7.2: Οθόνη τμήματος 1/2

Παρακάτω παρουσιάζονται αναλυτικά στατιστικά ανά έτος. Υπάρχει ένα σχεδιάγραμμα με μπάρες στο οποίο εμφανίζονται οι προτιμήσεις των επιτυχόντων από την πρώτη έως την έκτη θέση. Ένα αντίστοιχο σχεδιάγραμμα παρουσιάζει τις προτιμήσεις για τις πρώτες τρείς θέσεις όλων των υποψηφίων. Τα στατιστικά αυτά δίνονται με τα ακριβή άτομα ανά θέση. Τα δύο τελευταία στατιστικά που μπορεί να δει ο χρήστης σε αυτή την οθόνη είναι η συνολική συμμετοχή των υποψηφίων για το συγκεκριμένο τμήμα καθώς και τους συνολικούς επιτυχόντες σε αυτό. Στην κορυφή αυτών τον στατιστικών και κάτω από το σχεδιάγραμμα του ιστορικού βάσεων υπάρχει η λίστα με τις διαθέσιμες

χρόνιες στατιστικών. Αλλάζοντας την επιλογή ο χρήστης έχει τη δυνατότητα να προβάλει στατιστικά από παλιότερα έτη .



Σχήμα 7.3: Οθόνη τμήματος 2/2

7.2 Παλιά Θέματα

Στην κατηγορία των θεμάτων ο χρήστης έχει την δυνατότητα να προβάλει διάφορα θέματα από προηγούμενα έτη. Στην αρχή, υπάρχει διαχωρισμός των θεμάτων μέσω των τύπων εξετάσεων. Ως προεπιλεγμένος τύπος εξετάσεων έχει οριστεί το Γενικό Λύκειο, όμως αυτή η προεπιλογή μπορεί να αλλάξει αν ο χρήστης επιλέξει μέσω των ρυθμίσεων κάποιον διαφορετικό τύπο εξέτασης. Ύστερα τα θέματα προβάλλονται ανά μάθημα σε οριζόντιες λίστες, όπως φαίνεται και στο Σχήμα 7.4. Προς το παρόν η πληροφορία που προσφέρεται για κάθε θέμα είναι η χρονολογία του. Ο χρήστης επιλέγοντας κάποιο θέμα το προβάλει μέσω μίας τρίτης εφαρμογής του κινητού για προβολή PDF αρχείων. Σε περίπτωση που υπάρχουν περισσότερες από μία τέτοιες εφαρμογές, ο χρήστης επιλέγει αυτήν που επιθυμεί. Επίσης, του δίνεται η επιλογή να αποθηκεύσει την προτίμηση του έτσι ώστε κάθε φορά που ανοίγει κάποιο pdf αρχείο μέσα από την εφαρμογή να το ανοίγει με την καθορισμένη επιλογή. Σε περίπτωση που το κινητό δεν διαθέτει καμία εφαρμογή με τη δυνατότητα προβολής pdf αρχείου ο χρήστης θα μεταφερθεί στον προεπιλεγμένο browser από τον οποίο θα έχει την δυνατότητα να κατεβάσει το αρχείο.

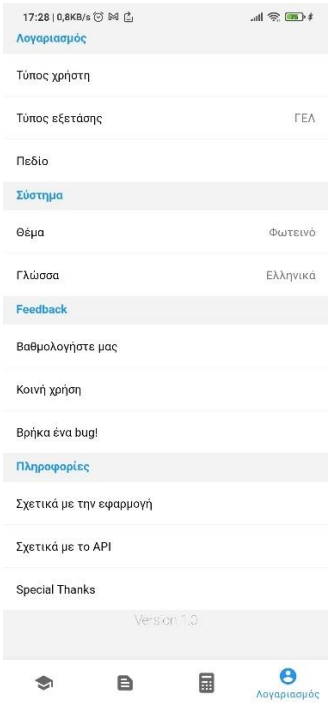
Σχήμα 7.4: Παλιά Θέματα

7.3 Αριθμομηχανή

Στην ενότητα της αριθμομηχανής ο χρήστης θα έχει την δυνατότητα για πιο εύκολο και γρήγορο υπολογισμό των μορίων βάση της βαθμολογίας του. Μπορεί να επιλέξει τον τύπο εξέτασης και το πεδίο που τον ενδιαφέρει και τα μαθήματα θα συμπληρωθούν αυτόματα από την εφαρμογή. Με κάθε αλλαγή που κάνει στις βαθμολογίες τα τελικά μόρια ενημερώνονται αυτόματα επιτρέποντας τον να δοκιμάζει διαφορετικές βαθμολογίες με εύχρηστο και ταχύτατο τρόπο.

7.4 Ρυθμίσεις

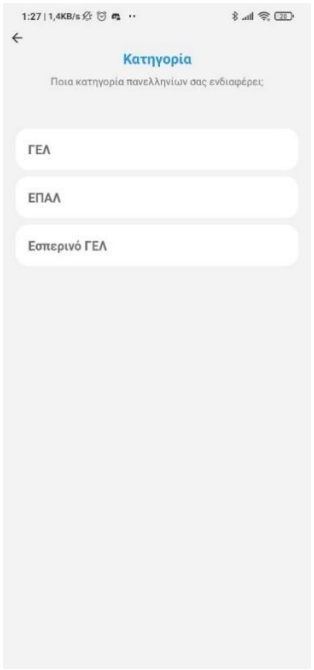
Στην ενότητα των ρυθμίσεων υπάρχουν διάφορες παραμετροποιήσεις που μπορεί να πραγματοποιήσει ο χρήστης, καθώς και πληροφορίες για την εφαρμογή. Η οθόνη αποτελείται από μία λίστα η οποία έχει τέσσερις υποενότητες.



Σχήμα 7.5: Ρυθμίσεις

7.4.1 Λογαριασμός

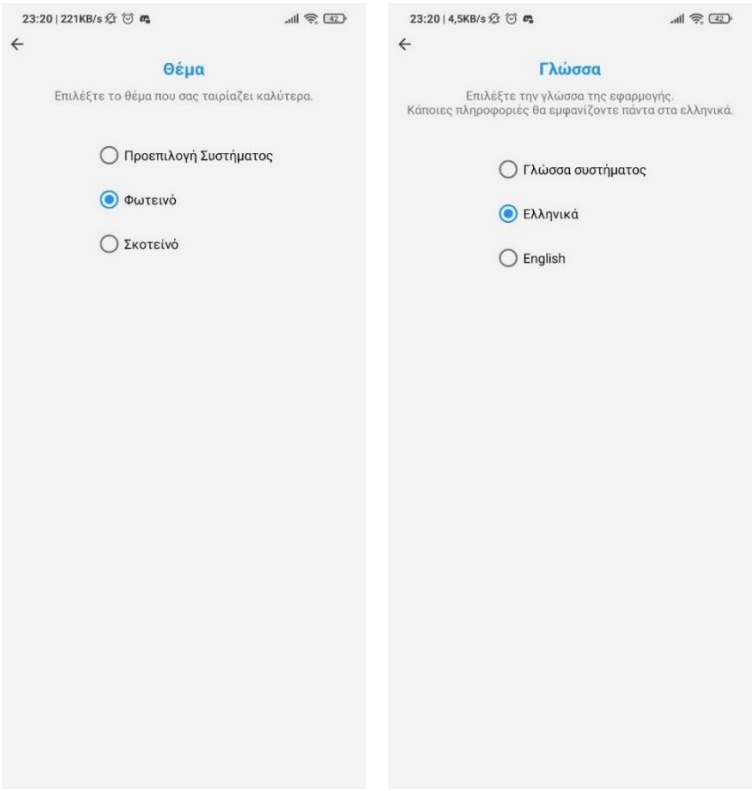
Αρχικά, υπάρχει η υποενότητα του Λογαριασμός. Σε αυτό το σημείο ο χρήστης μπορεί να επιλέξει και να αλλάξει τον τύπο εξέτασης και το πεδίο που τον ενδιαφέρει περισσότερο. Βάση αυτών των επιλογών η εφαρμογή θα κάνει συγκεκριμένες αλλαγές, παραδείγματος χάρη να εμφανίζεται πρώτος ο τύπος εξετάσεων και αριθμομηχανής μορίων που έχει επιλεχθεί, επιτυγχάνοντας την εξατομίκευση της εφαρμογής για τον χρήστη, κάνοντας την πιο προσιτή προς εκείνον.



Σχήμα 7.6: Επιλογή Τύπου εξέτασης

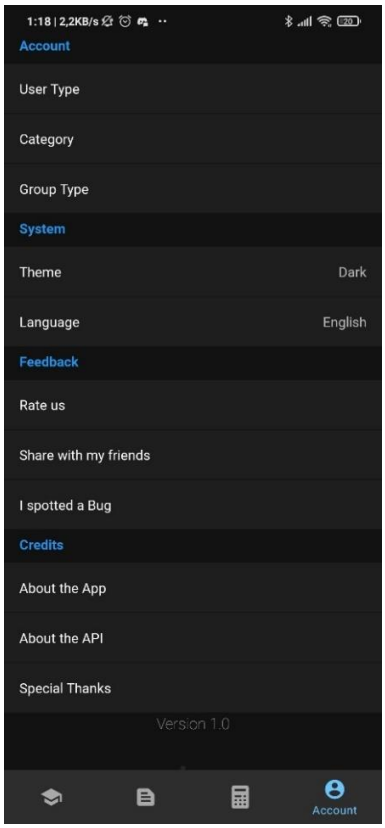
7.4.2 Σύστημα

Ακολουθεί η υποενότητα «Σύστημα». Αποτελείται από δύο ρυθμίσεις που μπορεί να πραγματοποιήσει ο χρήστης, αυτή του θέματος και της γλώσσας. Πρόκειται για δύο παρόμοιες οθόνες. Ο χρήστης έχει την επιλογή να επιλέξει την προεπιλογή συστήματος είτε και για την γλώσσα είτε για το θέμα. Και στις δύο περιπτώσεις η εφαρμογή θα συντονίζεται με την εκάστοτε επιλογή του συστήματος. Πέρα από, θα του παρέχετε η δυνατότητα να επιλέξει ανάμεσα σε σκοτεινό ή φωτεινό θέμα και ελληνική ή αγγλική λέξη. Για την αποθήκευση των ρυθμίσεων αρκεί να γυρίσει στην προηγούμενη οθόνη και η επιλογή του θα αποθηκευτεί.



Σχήμα 7.7: Ρυθμίσεις γλώσσας και θέματος

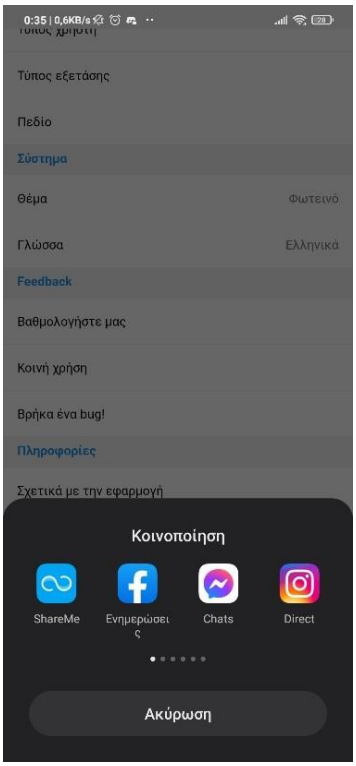
Στο Σχήμα 7.8 που ακολουθεί φαίνεται η οθόνη των ρυθμίσεων αφού ο χρήστης έχει επιλέξει για θέμα της εφαρμογής το σκοτεινό και για γλώσσα τα αγγλικά. Είναι σημαντικό να τονιστεί ότι κανένα από τα δύο API που χρησιμοποιούνται από την εφαρμογή δεν υποστηρίζει ως δεύτερη γλώσσα τα αγγλικά. Αυτό έχει ως αποτέλεσμα η εφαρμογή να είναι μερικώς μεταφρασμένη. Πιο συγκριμένα έχουν μεταφραστεί όλα τα κείμενα τα οποία υπάρχουν μέσα στην εφαρμογή, ενώ πληροφορίες που λαμβάνονται από τα API παραμένουν στα ελληνικά ανεξαρτήτου επιλογής.



Σχήμα 7.8: Επιλεγμένη γλώσσα ελληνικά, επιλεγμένο θέμα σκοτεινό

7.4.3 Feedback

Έπειτα, βρίσκεται η υποενότητα του «Feedback». Μέσω των επιλογών που παρέχονται στον χρήστη, του δίνονται οι δυνατότητες να βαθμολογήσουν την εφαρμογή στο Play Store, να κάνουν κοινή χρήση της εφαρμογής στέλνοντας ένα προκαθορισμένο κείμενο το οποίο περιέχει το URL της εφαρμογής στο Play Store και τέλος, να αποστείλουν κάποια αναφορά για κάποιο σφάλμα που έχουν εντοπίσει στην εφαρμογή. Με άλλα λόγια, μπορούν να εκφράσουν τα θετικά ή αρνητικά σχόλια τους είτε μέσω της κριτικής και των σχολίων του Play Store, είτε στέλνοντας απευθείας email εκφράζοντας τυχόν παράπονα ή σφάλματα που έχουν από την εφαρμογή.



Σχήμα 7.9: Κοινοποίηση εφαρμογής

7.4.4 Πληροφορίες

Τέλος, υπάρχει η υποενότητα «Πληροφορίες». Σε αυτήν ο χρήστης μπορεί να βρει πληροφορίες σχετικά με την εφαρμογή, με τις βιβλιοθήκες ελεύθερου λογισμικού που χρησιμοποιήθηκαν, την έκδοση της εφαρμογής, καθώς και μια σύντομη περιγραφή του API των βάσεων που χρησιμοποιήθηκε. Πρόκειται, δηλαδή, για μία εντελώς πληροφοριακή υποενότητα.

Κεφάλαιο 8ο Συμπεράσματα και βελτιστοποίηση

8.1 Συμπεράσματα

Το Υπουργείο Παιδείας και Θρησκευμάτων κάθε χρόνο δημοσιεύει τα δεδομένα των βάσεων

εισαγωγής και τα στατιστικά για την εισαγωγή στην τριτοβάθμια εκπαίδευση. Ο χώρος τα οποία αυτά δημοσιεύονται πρόκειται για τη σελίδα ανακοινώσεων του Υπουργείο, κάτι που καθιστά δύσκολη την αναζήτηση των αρχείων. Πάνω σε αυτά τα δεδομένα δημιουργήθηκε ένα RESTful API για την δομημένη και οργανωμένη προσφορά των δεδομένων αυτών.

Ο σκοπός αυτής της διπλωματικής εργασίας ήταν η δημιουργία μιας Android εφαρμογής η οποία θα προσφέρει διάφορα δεδομένα και πληροφορίες για όλους όσους τους ενδιαφέρει η διαδικασία της εισαγωγής σε κάποιο τριτοβάθμιο τμήμα μέσω των πανελλήνιων εξετάσεων. Πιο συγκεκριμένα, αξιοποιήθηκαν τα δεδομένα που προσφέρει το ήδη υλοποιημένο RESTful API για τις παλιές βάσεις και τα στατιστικά τους. Με την χρησιμοποίηση του API αυτούς, ο χρήστης μπορεί να αναζητήσει οποιαδήποτε τμήμα, ενεργό ή μη ενεργό. Μπορεί να δει τις βάσεις εισαγωγής παλιότερων ετών καθώς και διάφορα άλλα στατιστικά. Έπειτα στην ιδέα ενός πιο ολοκληρωμένου βοηθήματος για τους υποψηφίους των πανελληνίων δημιουργήθηκε ένα βοηθητικό API που προσφέρει τα παλιά θέματα των πανελληνίων καθώς και τις πληροφορίες για την υλοποίηση ενός έγκυρου υπολογιστή μορίων. Έτσι, ο χρήστης έχει την δυνατότητα μέσω της Android συσκευής του να αναζητήσει και να προβάλει παλιά θέματα εξετάσεων. Επίσης, του προσφέρεται η δυνατότητα μέσω του υπολογιστή μορίων να υπολογίζει με μεγαλύτερη ευκολία και ταχύτητα τα τελικά μόρια μέσω των βαθμολογιών που θα επιλέξει. Μέσω του βοηθητικού API που υλοποιήθηκε η εφαρμογή μπορεί να είναι συνέχεια ενημερωμένη με τις τελευταίες αλλαγές χωρίς την ανάγκη της ενημέρωσης της ίδιας της εφαρμογής από τον χρήστη. Παράλληλα, ο χρήστης μπορεί να εξατομικεύσει κατά κάποιον τρόπο της εφαρμογή σύμφωνα με τις δικιές του προτιμήσεις.

8.2 Μελλοντικές επεκτάσεις

Παρόλο που η εφαρμογή αποτελεί ήδη ένα ολοκληρωμένο βοήθημα για τους ενδιαφερόμενους των πανελληνίων εξετάσεων, θα μπορούσαν να πραγματοποιηθούν διάφορες βελτιστοποιήσεις και επεκτάσεις είτε τεχνικού θέματος είτε περιεχομένου.

Σχετικά με τις παλιές βάσεις εξετάσεων είναι εφικτό να προστεθούν επιπλέον φίλτρα για την καλύτερη αναζήτηση των χρηστών. Θα μπορούσαν να προσφέρονται φίλτρα σχετικά με τα μόρια του κάθε τμήματος. Ο χρήστης θα μπορούσε να δίνει ένα εύρος μορίων με το οποίο θα περιόριζε την αναζήτηση του μόνο σε τμήματα που η τελευταία τους βάση εισαγωγής είναι μέσα στο εύρος αυτό. Ένα ακόμα φίλτρο που θα μπορούσε να προστεθεί στην αναζήτηση είναι και η ταξινόμηση μέσω των μορίων. Φυσικά, πρέπει να τονιστεί πως για ορισμένες από της παραπάνω αλλαγές θα πρέπει να υπάρξουν και αντίστοιχες αναβαθμίσεις στο API των βάσεων.

Αντίθετα, με τις παραπάνω επεκτάσεις που χρειάζονται την αντίστοιχη αναβάθμιση του API των βάσεων κάποιες από τις επεκτάσεις που μπορούν να πραγματοποιηθούν χωρίς κάποια επιπλέον αναβάθμιση είναι αυτές του ιστορικού και των αγαπημένων τμημάτων. Στο πλαίσιο των αναβαθμίσεων αυτών ο χρήστης θα έχει την δυνατότητα να αποθηκεύσει τμήματα που τον ενδιαφέρουν και να τα προβάλει με μεγαλύτερη ευκολία καθώς και να βρίσκει εύκολα τμήματα που έχει αναζητήσει στο παρελθόν. Τελευταία, αλλά μπορεί και πιο σημαντική, επέκταση που μπορεί να υλοποιηθεί πάνω στα

τμήματα, είναι η σύγκριση τους, έτσι ώστε ο χρήστης να μπορεί να επιλέξει περισσότερα από ένα τμήματα και να τα συγκρίνει μεταξύ τους βλέποντας όλα τα χαρακτηριστικά τους.

Στην ενότητα οι βελτιστοποιήσεις που μπορούν να πραγματοποιηθούν είναι περισσότερο περιεχομένου παρά τεχνικές. Πιο συγκεκριμένα παράλληλα με τα παλιά θέματα εξετάσεων θα ήταν ιδιαιτέρως χρήσιμο να προσφέρονται και οι αντίστοιχες απαντήσεις. Επιπρόσθετα, τα θέματα της κάθε χρονιάς μπορούν να υπάρχουν και τα θέματα της Ομοσπονδία Εκπαιδευτικών Φροντιστών Ελλάδος (Ο.Ε.Φ.Ε.) μαζί με τις αντίστοιχες απαντήσεις.

Τέλος σε πιο γενικές γραμμές μπορεί να υλοποιηθεί μία λογική τοπικής αποθήκευσης για το σύνολο των πληροφοριών που προσφέρονται στην εφαρμογή. Παρότι, που στην ερώτηση του ερωτηματολογίου περί της διαθεσιμότητας στο διαδίκτυο οι περισσότεροι χρήστες απάντησαν πως δεν αντιμετωπίζουν τέτοιου είδους προβλήματα στην καθημερινότητα τους, η χρήση τοπικής αποθήκευσης θα είχε αρκετά πλεονεκτήματα. Θα περιοριζόταν σημαντικά η χρήση δεδομένων από το διαδίκτυο, κάτι το οποίο είναι πολύ σημαντικό για άτομα που χρησιμοποιούν συχνά τα δεδομένα του κινητού για πρόσβαση στο ίντερνετ. Επίσης, θα υπήρχε ελάφρυνση στα δύο API καθώς δεν θα πραγματοποιόντουσαν ερωτήματα σε αυτά με την ίδια συχνότητα.

Αυτές ήταν κάποιες από τις πιο σημαντικές και επεκτάσεις και αναβαθμίσεις που μπορούν να υλοποιηθούν μελλοντικά και να βελτιστοποιήσουν στο μέγιστο την εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] P. Deitel, H. Deitel and A. Deitel, Android: How to Program 2nd Edition. Pearson, 2014

[2] Google, “Android Studio documentation.” https://developer.android.com/studio. [Online; accessed 28. May. 2021].

[3] Google, “Android SDK Platform.” https://developer.android.com/studio/releases/platforms. [Online; accessed 28. May. 2021].

[4] Kotlinlang.org, “Kotlin documentation.” https://kotlinlang.org/docs/. [Online; accessed 30. May. 2021].

[5] gradle.org, “Gradle documentation.” https://docs.gradle.org/current/userguide/userguide.html. [Online; accessed 28. May. 2021].

[6] git-scm.com, “Git documentation.” https://git-scm.com/doc. [Online; accessed 28. May. 2021].

[7] github.com, “GitHub documentation.” https://docs.github.com/en. [Online; accessed 28. May. 2021].

[8] Google, “Guide to app architecture.” https://developer.android.com/jetpack/guide. [Online; accessed 5. June. 2021].

[9] Robert C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson, 2008.

[10] Google, “Saving UI States.” https://developer.android.com/topic/libraries/architecture/saving-states. [Online; accessed 5. June. 2021].

[11] Google, “Dependency Injection.” https://developer.android.com/training/dependency-injection. [Online; accessed 5. June. 2021].

[12] Google, “Dependency Injection with Hilt.” https://developer.android.com/training/dependency-injection/hilt-android. [Online; accessed 5. June. 2021].

[13] kotlinlang.org, “Kotlin coroutines on Android.” https://kotlinlang.org/docs/coroutines-overview.html. [Online; accessed 5. June. 2021].

[14] Google, “Kotlin coroutines on Android.” https://developer.android.com/kotlin/coroutines. [Online; accessed 5. June. 2021].

[15] Google, “Shrink, obfuscate, and optimize your app.” https://developer.android.com/studio/build/shrink-code. [Online; accessed 30. May. 2021].

[16] Google, “Firebase Google documentation” https://firebase.google.com/docs, 2021. [Online; accessed 5. June. 2021].

[17] Google, “Firebase Google: Crashlytics documentation” https://firebase.google.com/docs/crashlytic, 2021. [Online; accessed 15. June. 2021].

[18] Google, “Firebase Google: Analytics documentation” https://firebase.google.com/docs/analytics, 2021. [Online; accessed 15. June. 2021].

[19] Glide, “Glide documentation.” https://github.com/bumptech/glide. [Online; accessed 15. June. 2021].

[20] Google, “Google Material documentation.” https://material.io/design/introduction. [Online; accessed 15. June. 2021].

[21] Moshi, “Moshi documentation.” https://github.com/square/moshi. [Online; accessed 30. May. 2021].

[22] OkHttp, “OkHttp documentation.” https://square.github.io/okhttp/. [Online; accessed 2. June. 2021].

[23] Retrofit, “Retrofit documentation.” https://square.github.io/retrofit/. [Online; accessed 2. June. 2021].

[24] Glide, “Glide documentation.” https://github.com/bumptech/glide. [Online; accessed 2. June. 2021].

[25] MPAndroidChart, “MPAndroidChart documentation.” https://github.com/PhilJay/MPAndroidChart. [Online; accessed 2. June. 2021. 2021].

[26] MySQL, “MySQL 8.0 Reference Manual: What is MySQL?.” https://dev.mysql.com/doc/refman/8.0/en/whatismysql.html, 2021. [Online; accessed 2. June. 2021].

[27] mariadb.com, “MariaDB versus MySQL Compatibility.” https://mariadb.com/kb/en/mariadbvsmysqlcompatibility. Mar 2021. [Online; accessed 15. June.2021].
[28] json.org , “JSON.” https://www.json.org/jsonel. html. [Online; accessed 11. Mar. 2021].
[29] PHP Manual Contributors, “History of php.” https://www.php.net/manual/en/history.php.php, 2021. [Online; accessed 8. June. 2021].
[30] PDO, “PDO Data Objects” https://www.php.net/manual/en/book.pdo.php, 2021. [Online; accessed 8. June. 2021].
[31] L. Richardson and M. Amundsen, *RESTful Web APIs*, ch. 3. O’Reilly Media, Inc., 2013.
[32] PhpStorm, “PhpStorm documentation” https://www.jetbrains.com/phpstorm/documentation/, 2021. [Online; accessed 8. June. 2021].
[33] phpMyAdmin, “phpMyAdmin documentation” https://docs.phpmyadmin.net/en/latest/, 2021. [Online; accessed 10. June. 2021].
[34] R. Ramakrishnan, J. Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill, 2002
[35] Google, “Android Manifest.” https://developer.android.com/guide/topics/manifest/manifest-intro. [Online; accessed 10. June. 2021].
[36] Google, “Android Activities.” https://developer.android.com/guide/components/activities/intro-activities. [Online; accessed 10. June. 2021].
[37] Google, “Activity Lifecycle.” https://developer.android.com/guide/components/activities/activity-lifecycle. [Online; accessed 10. June. 2021].
[38] Google, “Android Fragment.” https://developer.android.com/guide/fragments. [Online; accessed 10. June. 2021].
[39] Google, “Fragment Lifecycle.” https://developer.android.com/guide/fragments/lifecycle. [Online; accessed 10. June. 2021].
[40] Google, “Navigation Component.” https://developer.android.com/guide/navigation. [Online; accessed 10. June. 2021].
[41] Google, “ViewBinding.” https://developer.android.com/topic/libraries/view-binding. [Online; accessed 10. June. 2021].
[42] material.io, “Bottom Navigation.” https://material.io/components/bottom-navigation/android. [Online; accessed 10. June. 2021].
[43] Google, “Android LiveData.” https://developer.android.com/topic/libraries/architecture/livedata. [Online; accessed 10. June. 2021].
[44] Google, “Android ViewModel.” https://developer.android.com/topic/libraries/architecture/viewmodel. [Online; accessed 10. June. 2021].
[45] Google, “Android RecyclerView.” https://developer.android.com/guide/topics/ui/layout/recyclerview. [Online; accessed 10. June. 2021].
[46] Google, “Android ListAdapter.” https://developer.android.com/reference/androidx/recyclerview/widget/ListAdapter. [Online; accessed 10. June. 2021].
[47] kotlinlang.org, “Lazy properties.” https://kotlinlang.org/docs/delegated-properties.html#lazy-properties. [Online; accessed 10. June. 2021]
[48] kotlinlang.org, “Lateinit.” https://kotlinlang.org/docs/properties.html#late-initialized-properties-and-variables. [Online; accessed 10. June. 2021].
[49] kotlinlang.org, “Extensions.” https://kotlinlang.org/docs/extensions.html. [Online; accessed 10. June. 2021]
[50] Google, “Shared Preferences.” https://developer.android.com/reference/android/content/SharedPreferences. [Online; accessed 10. June. 2021].

[51] Google, “Save key-value data.” https://developer.android.com/training/data-storage/shared-preferences. [Online; accessed 10. June. 2021].
[52] Google, “Motion Layout.” https://developer.android.com/training/constraint-layout/motionlayout. [Online; accessed 10. June. 2021].
[53] Google, “Saving State.” https://developer.android.com/guide/fragments/saving-state. [Online; accessed 10. June. 2021].
[54] Google, “String Resources.” https://developer.android.com/guide/topics/resources/string-resource. [Online; accessed 10. June. 2021].
[55] Google, “Intent.” https://developer.android.com/reference/android/content/Intent. [Online; accessed 10. June. 2021].
[56] Google, “Interacting with Other Apps.” https://developer.android.com/training/basics/intents. [Online; accessed 10. June. 2021].