ΚΑΤΕΥΘΥΝΣΗ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Πτυχιακή Εργασία

Ανάπτυξη εφαρμογής για κινητά Android

Του φοιτητή:

Κούτση Σίμων

(*ΑΜ: 201400104)*

Επιβλέπων Καθηγητής:

Διονύσης Ρεΐσης

Αθήνα 2023

Περιεχόμενα

Εισαγωγή

Τα τελευταία χρόνια, γίνεται εύκολα αντιληπτό, πως η ανθρωπότητα βιώνει μια ραγδαία ανάπτυξη στις επιστήμες και την τεχνολογία. Οι δυο αυτοί τομείς έρευνας και εξέλιξης λειτουργούν σαν συγκοινωνούντα δοχεία όπου η πρόοδος του ενός τομέα τροφοδοτεί την ανάπτυξη του άλλου. Η τεχνολογική αυτή επανάσταση έχει καταφέρει σε σύντομο χρονικό διάστημα να βελτιώσει αισθητά την ποιότητα ζωής των ανθρώπων και να προσφέρει νέες δυνατότητες στις σύγχρονες κοινωνίες. Αυτό που κάνει όμως αυτήν την τεχνολογική ανάπτυξη πιο ξεχωριστή είναι πως δεν απευθύνεται μόνο σε συγκεκριμένες κοινωνικές ομάδες, αλλά ένα μεγάλο μέρος της είναι εύκολα προσβάσιμο για τον μέσο άνθρωπο.

Το καλύτερο, ίσως, παράδειγμα που δείχνει πως η τεχνολογία έχει διεισδύσει στην καθημερινότητα των ανθρώπων είναι το διαδίκτυο σε συνδυασμό με τις ηλεκτρονικές συσκευές. Καθημερινά όλο ένα και περισσότερες συσκευές αποκτάνε τον τίτλο της «έξυπνης» συσκευής, καθώς έχουν τη δυνατότητα να επικοινωνούν άμεσα με το διαδίκτυο και να τρέχουν εφαρμογές κάτι που αυξάνει το εύρος των λειτουργιών τους αλλά προσφέρει και μεγαλύτερη αλληλεπίδραση μεταξύ συσκευής και ανθρώπου ή συσκευής με άλλη συσκευή.

Όταν αναφερόμαστε στον όρο «έξυπνη» συσκευή, η πρώτη συσκευή που έρχεται στο μυαλό του μέσου ανθρώπου είναι το κινητό τηλέφωνο και όχι άδικα. Τα κινητά τηλέφωνα σε πολύ σύντομο χρονικό διάστημα έχουν εξελιχθεί από απλά τηλέφωνα σε μικρούς ισχυρούς υπολογιστές με δυνατότητες να υλοποιήσουν πληθώρα λειτουργιών. Για τον σκοπό αυτό έχει αναπτυχθεί ένας μεγάλος αριθμός εφαρμογών για κινητά (και όχι μόνο), που έχουν ως στόχο την πραγματοποίηση κάποιου έργου με όσο το δυνατόν πιο άμεσο, απλό και κατανοητό τρόπο για τον χρήστη.

Με βάση και τα παραπάνω, μπορούμε εύκολα να καταλάβουμε, γιατί οι κλάδοι γύρω από την ανάπτυξη, τη συντήρηση, τη βελτίωση όλων των ειδών των εφαρμογών, γνωρίζουν τεράστια αύξηση στη ζήτηση της αγοράς εργασίας.

Οι εφαρμογές, πλέον στις μέρες μας, χαρακτηρίζονται από μεγάλη ποικιλομορφία καθώς μια εφαρμογή μπορεί να είναι από μια απλή αριθμομηχανή, μια εφαρμογή για τον καιρό, ένα μέσο κοινωνικής δικτύωσης, ένα ηλεκτρονικό κατάστημα αλλά και ένα σύνθετο παιχνίδι που συνδυάζει περίπλοκα χαρακτηριστικά, εικόνα και ήχο.

Στην παρούσα πτυχιακή εργασία έχουμε αναπτύξει μια εφαρμογή, για κινητά τηλέφωνα τα οποία διαθέτουν το λειτουργικό σύστημα Android.

Περίληψη

Το περιεχόμενο της συγκεκριμένης διπλωματικής εργασίας, όπως προδιαθέτει και ο τίτλος της, αφορά ως επί το πλείστων τη δημιουργία μιας εφαρμογής για κινητά που διαθέτουν το λειτουργικό σύστημα «Android». Η εφαρμογή που αναπτύχθηκε έχει κατά κύρια βάση χαρακτηριστικά που θυμίζουν μέσο κοινωνικής δικτύωσης, το οποίο όμως έχει ως βασικό στόχο την ανάδειξη και την προβολή καλλιτεχνών που δραστηριοποιούνται στον χώρο της δερματοστιξίας. Πέρα όμως από την περιγραφή της εφαρμογής και της λειτουργικότητας της, στην εργασία γίνεται περιεκτική αναφορά στο λειτουργικό σύστημα «Android» και στα πλεονεκτήματα που αυτό προσφέρει σε όποιον το επιλέγει. Ακόμα αναφέρονται οι βασικοί προγραμματιστικοί τρόποι με τους οποίους μπορεί να αναπτυχθεί μία εφαρμογή για κινητά «Android», με ιδιαίτερη αναφορά στο framework που χρησιμοποιήθηκε για την ανάπτυξη της συγκεκριμένης εφαρμογής, το «Flutter». Τέλος, ένα κομμάτι της εργασίας έχει αφιερωθεί στην αρχιτεκτονική του κώδικα. Σε αυτό το κομμάτι γίνεται μία περιγραφή της φιλοσοφίας της αρχιτεκτονικής που επιλέχθηκε, αναδεικνύοντας τα πλεονεκτήματα της, καθώς επίσης αναλύεται και ο τρόπος που η φιλοσοφία αυτή εφαρμόστηκε τελικά στον κώδικα της παρούσας εφαρμογής, εστιάζοντας σε μερικά πιο τεχνικά ζητήματα.

Συνοπτική περιγραφή της εφαρμογής

Η εφαρμογή η οποία αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας, ως ιδέα, είναι ένας συνδυασμός μέσου κοινωνικής δικτύωσης και μιας εφαρμογής παροχής υπηρεσιών.

Ο κεντρικός στόχος της εφαρμογής είναι να δημιουργηθεί ένα σημείο αναφοράς το οποίο θα συγκεντρώνει καταστήματα αλλά και ανεξάρτητους καλλιτέχνες που δραστηριοποιούνται γύρω από τη δερματοστιξία. Ουσιαστικά, δηλαδή, κάθε καλλιτέχνης θα μπορεί να δημιουργήσει ένα επαγγελματικό προφίλ μέσα από το οποίο θα μπορεί να διαφημίζει και να αναδεικνύει τη δουλειά του.

Ταυτόχρονα όμως στην εφαρμογή θα μπορούν να έχουν προφίλ και απλοί χρήστες οι οποίοι θα είναι και οι εν δυνάμει πελάτες για τα επαγγελματικά προφίλ. Οι απλοί χρήστες θα μπορούν εύκολα να αναζητήσουν καλλιτέχνες με βάση τις προτιμήσεις τους, να συγκρίνουν τη δουλειά τους, να διαβάσουν τις απόψεις άλλων χρηστών και τελικά να επικοινωνήσουν με όποιους επαγγελματίες επιθυμούν για περεταίρω λεπτομέρειες. Ακόμα όμως και αν δεν ενδιαφέρονται άμεσα για τις υπηρεσίες των επαγγελματικών προφίλ θα μπορούν να χρησιμοποιούν την εφαρμογή σαν ένα μέσο που μπορούν να περάσουν ευχάριστα την ώρα τους βλέποντας δημιουργικές δημοσιεύσεις.

Επιλογή Λειτουργικού Συστήματος

Ένα σημαντικό βήμα για την ανάπτυξη μιας εφαρμογής για κινητά είναι η επιλογή του λειτουργικού συστήματος πάνω στην οποία αυτή θα χτιστεί. Στην αγορά αυτή τη στιγμή υπάρχει πληθώρα λειτουργικών συστημάτων αν και στην πραγματικότητα δυο είναι αυτά που ξεχωρίζουν και απολαμβάνουν πρακτικά την καθολική προτίμηση των χρηστών. Φυσικά, τα λειτουργικά συστήματα στα οποία αναφερόμαστε είναι το Android που αναπτύσσεται κυρίως από την Google και το iOS που αναπτύσσεται από την Apple.

Για να γίνει αντιληπτό το μέγεθος της μερίδας που καταλαμβάνουν αυτά τα δυο συστήματα στην αγορά μπορούμε να δούμε τα στοιχεία στον πίνακα (1), όπου μας δείχνει το πως διαμορφώθηκε η προτίμηση λειτουργικού συστήματος το πρώτο τετράμηνο του 2023, σε παγκόσμιο επίπεδο.

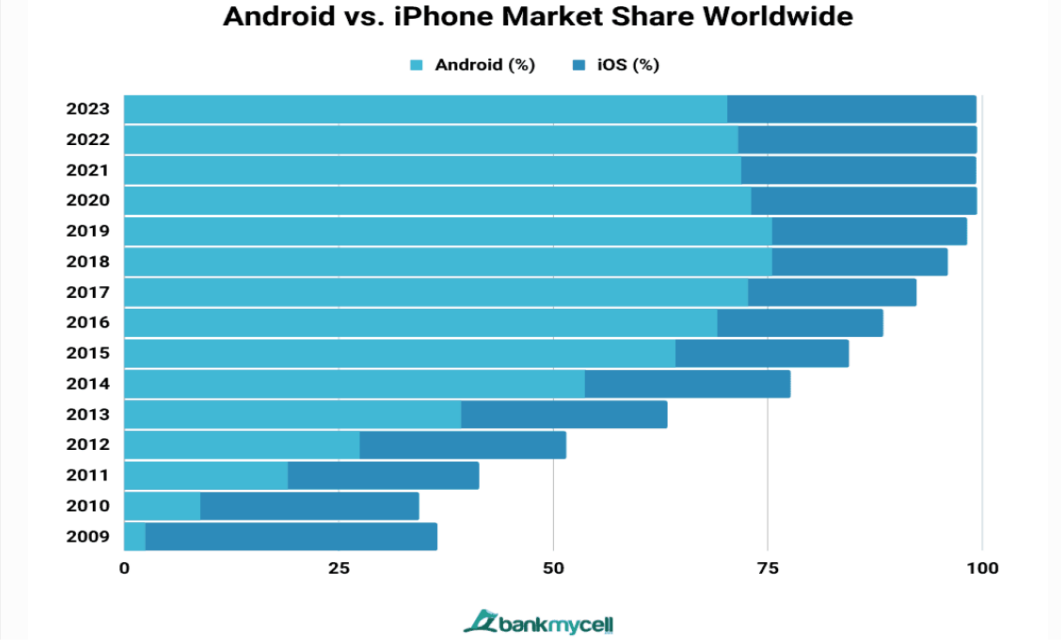
Operating System	Quarterly Market Share (%)
Android	70.89
iOS	28.36
Samsung	0.38
KaiOS	0.18
Windows	0.02
Other	0.16

Πίνακας 1: Ποσοστό προτίμησης λειτουργικών συστημάτων σε παγκόσμιο επίπεδο, με βάση στοιχεία για το πρώτο εξάμηνο του 2023.

(πηγή: https://www.bankmycell.com/blog/android-vs-apple-market-share/)

Ένα ακόμα συμπέρασμα που μπορούμε να εξάγουμε από τα στοιχεία του πίνακα (1) είναι πως αν πλέον απομονώσουμε το Android και το iOS, βλέπουμε πως υπάρχει μια ξεκάθαρη προτίμηση υπέρ του συστήματος Android, με επτά στους δέκα χρήστες παγκοσμίως να το προτιμούν.

Ακόμα ένα ενδιαφέρον διάγραμμα είναι αυτό στην εικόνα (1) καθώς μας δείχνει πως έχει εξελιχθεί η δυναμική των Android και iOS, παγκοσμίως, σε σχέση με τον ανταγωνισμό αλλά και μεταξύ τους, τα τελευταία χρόνια που εξελίχθηκαν οι έξυπνες συσκευές.



Εικόνα 1: Δυναμική εξέλιξη των Android και iOS στην προτίμηση των χρηστών.

(πηγή : https://www.bankmycell.com/blog/android-vs-apple-market-share/)

Λίγα λογία για το Android

Τα απλά στατιστικά στοιχεία που παραθέσαμε παραπάνω είναι ικανά να δώσουν μια σύντομη αλλά πειστική απάντηση στο ερώτημα «γιατί κάποιος που θέλει να αναπτύξει μια εφαρμογή να επιλέξει το Android;». Η απάντηση είναι πως το Android παραμένει σταθερά το νούμερο ένα λειτουργικό σύστημα που χρησιμοποιούν οι χρήστες, με αξιοσημείωτη διαφορά από το δεύτερο. Οπότε είναι φυσιολογικό κάθε εταιρία ή ιδιώτης που σκοπεύει να κυκλοφορήσει στην αγορά μια νέα εφαρμογή να έχει ως πρώτο μέλημα το προϊόν του να έχει τη δυνατότητα να απευθυνθεί και να μπορεί να είναι προσιτό από όσο το δυνατόν περισσότερους καταναλωτές.

Φυσικά ,ένα λειτουργικό σύστημα , για να γνωρίζει τέτοια σταθερή αύξηση στη δημοτικότητα του τα τελευταία χρόνια θα πρέπει να έχει και κάποιο σημαντικό πλεονέκτημα συγκριτικά με τον ανταγωνισμό. Στην περίπτωση του Android τα βασικά πλεονεκτήματα του πηγάζουν από το γεγονός ότι είναι ένα λειτουργικό σύστημα του οποίου το λογισμικό είναι ανοιχτού κώδικα.

Ένα πρότζεκτ ανοιχτού κώδικα δίνει τη δυνατότητα σε οποιονδήποτε, ελεύθερα, να το χρησιμοποιήσει, να το μελετήσει, να το αναπτύξει αλλά και να το διανείμει ως μέρος ενός καινούργιου πρότζεκτ. Έτσι και η βασική πλατφόρμα του Android, γνωστή ως Android Open Source Project (AOSP) είναι διαθέσιμη για όποιον θέλει να τη χρησιμοποιήσει, χωρίς να χρειάζεται να πληρώσει κάποιο αντίτιμο για άδεια χρήσης. Το γεγονός ότι στην πράξη η χρήση της AOSP είναι δωρεάν, οδήγησε πολλές εταιρείες να αναπτύξουν τα προϊόντα τους ως συσκευές Android. Μεταξύ αυτών των εταιριών περιλαμβάνονται και κολοσσοί όπως η Samsung, η Xiaomi, η LG, η Huawei και πολλές άλλες ακόμα που δραστηριοποιούνται σε όλων των ειδών τις ηλεκτρικές συσκευές.

Τα γεγονός ότι ένα πρότζεκτ ανοιχτού κώδικα μπορεί να χρησιμοποιηθεί και να διανεμηθεί δωρεάν δεν είναι το μοναδικό όφελος που απολαμβάνει όποιος το επιλέγει. Ένα πλεονέκτημα, ίσως ακόμα πιο σημαντικό είναι πως ένα τέτοιο πρότζεκτ δίνει πολλούς βαθμούς ελευθερίας σε όσους το χρησιμοποιούν. Από τη στιγμή που υπάρχει άμεση πρόσβαση στον πηγαίο κώδικα του λειτουργικού συστήματος, κάθε εταιρεία που αναπτύσσει ένα προϊόν μπορεί να εστιάσει την ανάπτυξη και τη βελτίωση του κώδικα με βάση τις δικές της ανάγκες και προτιμήσεις και με βάση τις δυνατότητες που προσφέρει το προϊόν της. Με απλά λόγια, μια εταιρεία που αναπτύσσει ένα κινητό Android έχει τη δυνατότητα να προσαρμόσει το λειτουργικό σύστημα όπως η ίδια το κρίνει, ώστε να έχει τον απόλυτο έλεγχο στο πως θα χρησιμοποιηθούν οι πόροι και η δυνατότητες που προσφέρονται από τη συσκευή της. Κάτι τέτοιο δεν θα ήταν δυνατό με τη χρήση ενός κλειστού κώδικα λειτουργικό σύστημα, όπως είναι το iOS, καθώς σε αυτή την περίπτωση ο κώδικας δεν είναι προσβάσιμος, με αποτέλεσμα ο τρόπος που επικοινωνεί το λειτουργικό σύστημα με τη συσκευή να είναι προκαθορισμένος χωρίς να υπάρχει η δυνατότητα για εξατομίκευση. Κάποιος που φτιάχνει μια συσκευή με το Android μπορεί να έχει ακόμα ελευθερία στο πως θα γίνεται η διαχείριση διαφόρων δεδομένων που θα συλλέγει η συσκευή

του, καθώς δεν υπάρχουν σκοτεινά σημεία στον κώδικα του, με αποτέλεσμα να αυξάνονται τα επίπεδα ασφάλειας.

Συνοψίζοντας τα παραπάνω, τελικά ένα λειτουργικό σύστημα κλειστού κώδικα δημιουργεί μια σχέση εξάρτησης μεταξύ της πλευράς που χρησιμοποιεί το σύστημα και της πλευράς που το παρέχει. Πάντα η μια πλευρά θα είναι δέσμια της άλλης σε ότι αφορά τη συντήρηση, τη βελτίωση, την επίλυση προβλημάτων του λειτουργικού συστήματος. Με το Android όπως περιγράψαμε, κάθε ένας που το επιλέγει έχει την ανεξαρτησία και την ελευθερία να αναπτύξει το πρότζεκτ με βάση το δικό του σχεδιάγραμμα και χωρίς την εξάρτηση από τρίτους παρόχους.

Τέλος αξίζει να σημειώσουμε πως ένα μέρος της ευελιξίας που προσφέρουν τα Android φτάνει και στον τελικό χρήστη. Ένας χρήστης κινητού Android έχει πολύ περισσότερες δυνατότητες να κάνει παρεμβάσεις στη συσκευή του, είτε αυτές έχουν αισθητικό χαρακτήρα είτε οργανωτικό κάτι που δεν έχουν οι χρήστες των iPhone. Φυσικά, η ελευθερία που έχει ο τελικός χρήστης δεν είναι καθολική καθώς αν και μια εταιρεία μπορεί να χρησιμοποίησε ως βάση το AOSP, κομμάτια του νέου κώδικα που ανέπτυξε μπορεί να μην είναι ανοικτού κώδικα με αποτέλεσμα ο χρήστης να μην έχει πρόσβαση σε αυτά. Ακόμα και έτσι όμως δεν παύουν, τα Android, να προσφέρουν πολύ μεγαλύτερη ευελιξία σε όσους τις επιλέγουν συγκριτικά με τα iOS.

Ανάπτυξη εφαρμογής για Android

Έχοντας σκεφτεί την ιδέα της εφαρμογής και το λειτουργικό σύστημα πάνω στο οποίο θα τη δημιουργήσουμε, μένει να αποφασίσουμε σε ποια γλώσσα προγραμματισμού θα την αναπτύξουμε. Εδώ μας προσφέρονται πρακτικά δυο τρόποι για να επιλέξουμε.

Ο ένας είναι να αναπτύξουμε την εφαρμογή στις γλώσσες που θεωρούνται βέλτιστες για το εκάστοτε λειτουργικό σύστημα. Στην περίπτωση μας, που έχουμε επιλέξει το Android, η Java υπήρξε η τυπική επιλογή για τους προγραμματιστές. Τα τελευταία χρόνια όμως την προτίμηση έχει κερδίσει η Kotlin καθώς είναι μια γλώσσα με πιο μοντέρνα χαρακτηριστικά που μπορούν να οδηγήσουν σε πιο καθαρό κώδικα αλλά και σε καλύτερες επιδόσεις. Καθώς όμως η Kotlin είναι συμβατή με την Java δεν είναι απίθανο να συναντήσει κάποιος σε εφαρμογές, τον συνδυασμό των δυο. Η ανάπτυξη προγραμμάτων με αυτή την επιλογή μπορούμε να πούμε σε ελεύθερη μετάφραση, από τον αγγλικό όρο native app development, πως είναι ο πιο καθαρός τρόπος να αναπτυχθεί μια εφαρμογή για Android. Επειδή η ανάπτυξη γίνεται αποκλειστικά για την πλατφόρμα του Android, η εφαρμογή μπορεί να έχει τις βέλτιστες επιδόσεις και να εκμεταλλευτεί πλήρως τα χαρακτηριστικά και τις δυνατότητες που προσφέρονται από το λειτουργικό σύστημα.

Η δεύτερη επιλογή που μπορούμε να επιλέξουμε είναι, σε αγγλική ορολογία, το cross-platform development. Ουσιαστικά δηλαδή να χρησιμοποιήσουμε ένα framework το οποίο μας επιτρέπει να αναπτύξουμε μια εφαρμογή δημιουργώντας μια μοναδική βάση κώδικα, σε μία μόνο γλώσσα

προγραμματισμού, που όμως να μπορεί να τρέξει και σε Android και σε iOS, με μικρές παραλλαγές στον κώδικα για κάθε περίπτωση. Καταλαβαίνουμε άμεσα πως για κάποιον που θέλει να κυκλοφορήσει μια εφαρμογή που θα είναι συμβατή και σε Android και σε iOS, περίπτωση που είναι και η πιο σύνηθες, η επιλογή του cross-platform development δίνει σημαντικά πλεονεκτήματα. Το πιο σημαντικό πλεονέκτημα είναι πως επειδή η εφαρμογή θα αναπτύσσεται σε μία γλώσσα και θα έχει μία βάση, χρειάζεται μία μόνο ομάδα προγραμματιστών που θα την αναπτύσσουν και θα τη συντηρούν. Σε αντίθετη περίπτωση, επειδή θα είχαμε native app development για κάθε λειτουργικό σύστημα, θα ήταν απαραίτητο περισσότερο και πιο εξιδεικευμένο προσωπικό να αναπτύσσει και να συντηρεί δυο διαφορετικές βάσης κώδικα, σε τουλάχιστον δυο διαφορετικές γλώσσες. Γίνεται, έτσι, εύκολα αντιληπτό πως η επιλογή του cross-platform development μπορεί να μειώσει σημαντικά το κόστος της ανάπτυξης και συντήρησης μιας εφαρμογής, ειδικά για μια νεοσύστατη επιχείρηση που μπορεί να έχει ένα περιορισμένο προϋπολογισμό.

Ένα τέτοιο framework που μας προσφέρει τις δυνατότητες της δεύτερης επιλογής είναι το Flutter.

Το Flutter

Το Flutter είναι ένα framework ανοικτού κώδικα που δημιουργήθηκε από την Google. Εκτός όμως από την Google, ως πρότζεκτ ανοικτού κώδικα, υποστηρίζεται από μια μεγάλη κοινότητα προγραμματιστών που συνεισφέρουν σημαντικά στην ανάπτυξη και τη βελτίωση του. Αυτή η δυναμική της κοινότητας και της πληθώρας πληροφοριών και βοήθειας που μπορεί να βρει ένας προγραμματιστής που επιλέγει το Flutter είναι ένα πρώτο σημαντικό πλεονέκτημα που αυτό προσφέρει. Ένα ακόμα αξιόλογο πλεονέκτημα που προκύπτει, είναι πως το Flutter μπορεί να το χρησιμοποιήσει ο καθένας δωρεάν.

Η γλώσσα προγραμματισμού που χρησιμοποιείται για την ανάπτυξη εφαρμογών στο Flutter είναι η Dart. Η Dart είναι μια γλώσσα αντικειμενοστραφή προγραμματισμού, ανοιχτού κώδικα, που επίσης αναπτύχθηκε από την Google και έχει πολύ καλή υποστήριξη από την κοινότητα.

Γενικά το Flutter είναι ένα framework που βελτιώνεται συνεχώς. Το αποτέλεσμα που μπορεί να πέτυχει κάποιος αναπτύσσοντας την εφαρμογή του στο Flutter είναι ανάλογου επιπέδου με αντίστοιχες εφαρμογές που αναπτύχθηκαν με native τρόπο ενώ τις περισσότερες φορές οι διαφορές σε επιδόσεις και ποιότητα είναι αναξιόλογες. Για αυτό άλλωστε όλο και περισσότερες εταιρείες πλέον το επιλέγουν για να αναπτύξουν τις εφαρμογές τους, συμπεριλαμβανομένου και μεγάλων κολοσσών.

Τα παραπάνω συνέβαλαν στο να επιλεγεί το Flutter για την ανάπτυξη και της εφαρμογής της παρούσας πτυχιακής εργασίας.

Παρουσίαση της εφαρμογής

Νωρίτερα περιγράψαμε συνοπτικά την ιδέα πάνω στην οποία βασίζεται η εφαρμογή που αναπτύχθηκε για τα πλαίσια της παρούσας πτυχιακής εργασίας. Όμως για να μπορέσει να αποτυπωθεί πλήρως η έμπνευση σε μια ολοκληρωμένη και απολύτως λειτουργική εφαρμογή θα χρειαζόντουσαν περισσότερες γνώσεις (όπως κώδικας σχετικά με την επικοινωνία της εφαρμογής με κάποια βάση δεδομένων) και χρόνος, σε επίπεδο που θα ξεφευγε από τα πλαίσια μιας ατομικής πτυχιακής εργασίας. Παρόλα αυτά η εφαρμογή που αναπτύχθηκε καταφέρνει επάξια να αναδείξει τη λειτουργικότητα και το ύφος που θα θέλαμε, καθώς υλοποιεί ένα σημαντικό κομμάτι από αυτό που θα ήταν η πλήρως λειτουργική έκδοσης της.

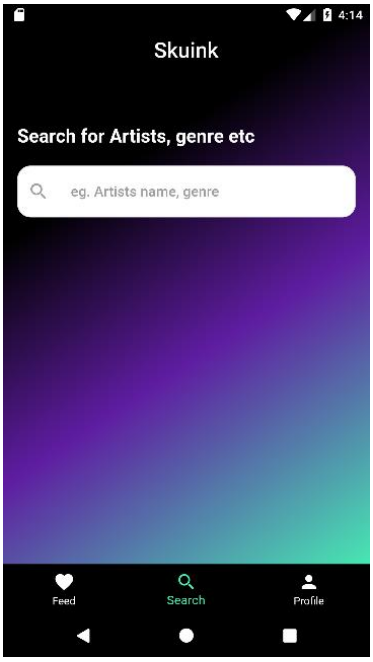
Το κομμάτι που υλοποιήθηκε είναι αυτό που θα συναντούσε ένας εγγεγραμμένος απλός χρήστης, που εισέρχεται στην εφαρμογή. Ουσιαστικά το κομμάτι της εγγραφής και της ταυτοποίησης των χρηστών, καθώς και της επεξεργασίας των επαγγελματικών προφίλ δεν το υλοποιήσαμε καθώς για να έχουν πραγματική ουσία θα προϋπέθεταν πραγματική επικοινωνία με κάποια βάση δεδομένων.

Αισθητικά και λειτουργικά, η εφαρμογή έχει στοιχεία από υπάρχουσες γνωστές εφαρμογές, κυρίως τύπου μέσων κοινωνικής δικτύωσης. Αυτό έχει ως στόχο η πλοήγηση εντός της εφαρμογής να είναι όσο το δυνατόν πιο απλή και να βγαίνει φυσικά στον χρήστη.

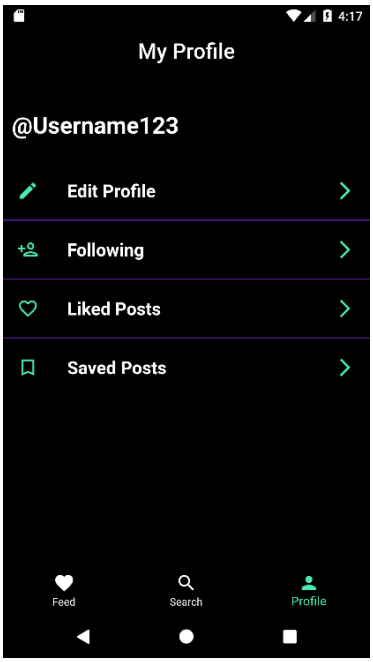
Ανοίγοντας, λοιπόν, την εφαρμογή η πρώτη σελίδα που θα συναντήσει ο χρήστης θα είναι μια σελίδα αναζήτησης. Στη σελίδα αυτή ο χρήστης θα μπορεί να αναζητήσει επαγγελματικά προφίλ με βάση το όνομα τους αλλά και με βάση το είδος του τατουάζ πάνω στο οποίο αυτά εξειδικεύονται. Επίσης στο κάτω μέρος της οθόνης του θα συναντήσει μια μπάρα (bottom tab bar) η οποία θα του επιτρέπει να περιηγηθεί μεταξύ των σελίδων «Feed», «Search», «Profile» πατώντας τα αντίστοιχα κουμπιά. (Εικόνα 2)

Επιλέγοντας την σελίδα «Feed» ο χρήστης θα μπορεί να δει μια ροή από δημοσιεύσεις οι οποίες έχουν ανεβεί στην εφαρμογή από τα επαγγελματικά προφίλ που ακολουθεί. Όπως είναι λογικό, αν ο χρήστης δεν ακολουθεί κανένα προφίλ, η σελίδα αυτή θα εμφανίζεται κενή.

Στην σελίδα «Profile» ο χρήστης θα έχει τη δυνατότητα να τροποποιήσει κάποια βασικά στοιχεία του προφίλ του αλλά και να βρει συγκεντρωτικά τις λίστες με τα προφίλ που ακολουθεί, τις δημοσιεύσεις που έχει πατήσει «μου αρέσει (Like)» και τις δημοσιεύσεις που έχει αποθηκεύσει. (Εικόνα 3)

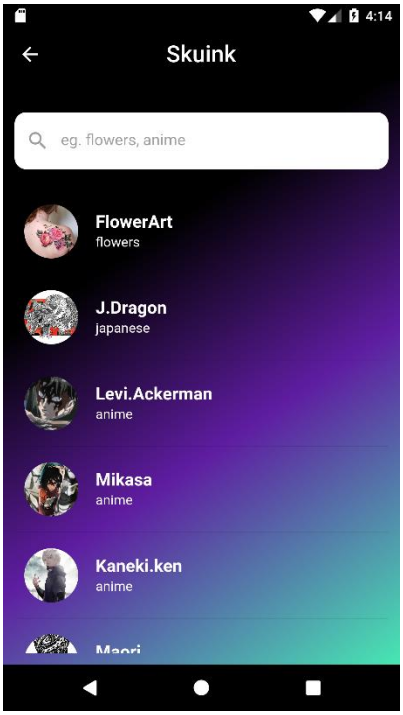


Εικόνα 2: Αρχική οθόνη/ οθόνη αναζήτησης. Κάτω βλέπουμε τα κουμπιά "Feed", "Search", "Profile".

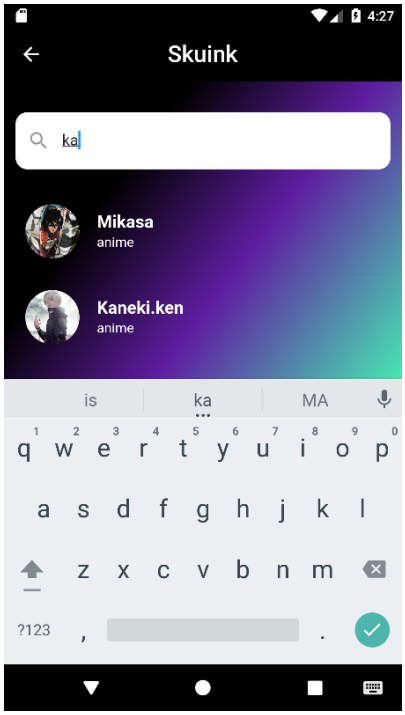


Εικόνα 3: Οθόνη "Profile". Το 'username123' είναι προεπιλογή και αλλάζει στο ''Edit Profile"

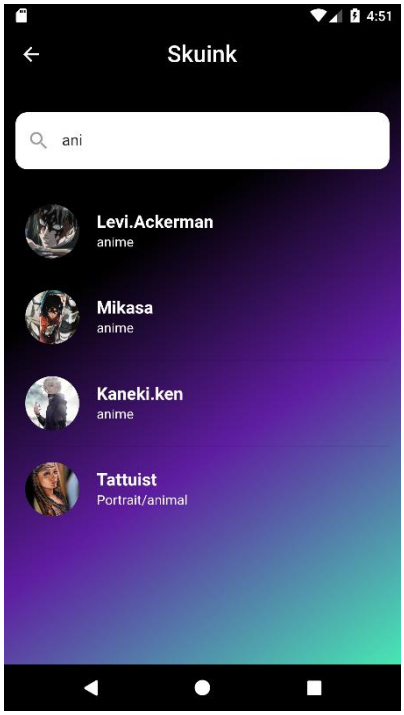
Αφού εξοικειωθήκαμε με τον βασικό κορμό της εφαρμογής μπορούμε να επιστρέψουμε στη σελίδα αναζήτησης, να ελέγξουμε τη λειτουργικότητα της. Πατώντας πάνω στο πλαίσιο της αναζήτησης, ο χρήστης θα ανακατευθυνθεί σε μια παρόμοια σελίδα αναζήτησης που θα έχει επιπρόσθετα κάποιες προτάσεις με προφίλ. Εδώ ο χρήστης μπορεί να επιλέξει οποιαδήποτε πρόταση της λίστας των προεπιλογών, πατώντας πάνω σε κάποιο προφίλ ή να προχωρήσει στη δικιά του αναζήτηση και να επιλέξει ένα από τα προφίλ που θα του εμφανιστούν. Η λίστα αναζήτησης μπορεί να «συρθεί» προς τα πάνω (είναι scrollable). (Εικόνες 4,5,6)



Εικόνα 4: Οθόνη αφού έχουμε πατήσει τη μπάρα αναζήτησης. Βλέπουμε τη λίστα με τις προτάσεις.



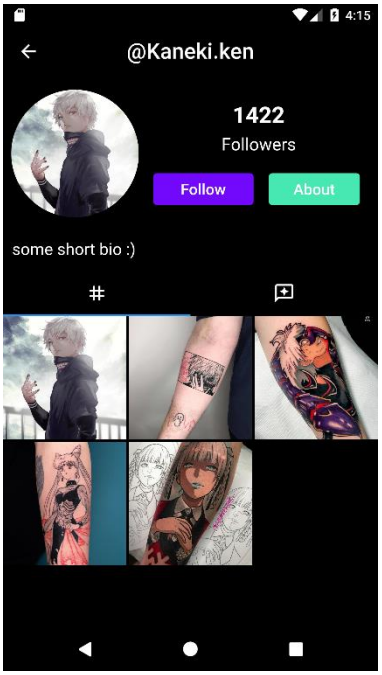
Εικόνα 5: Αναζήτηση με βάση το όνομα.



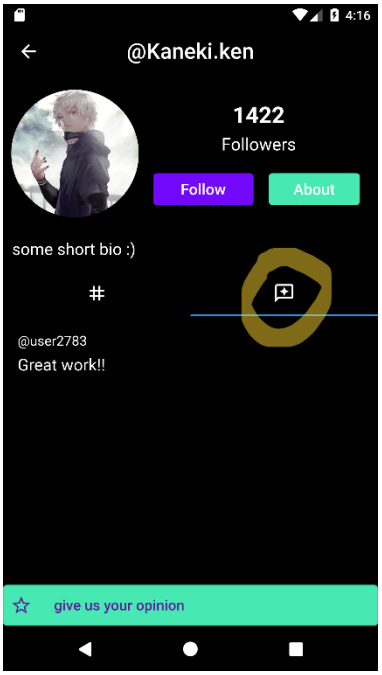
Εικόνα 6: Αναζήτηση με βάση το genre.

Αφού ο χρήστης πατήσει πάνω σε οποιαδήποτε επιλογή από τη λίστα αναζήτησης θα ανακατευθυνθεί στο αντίστοιχο προφίλ που επέλεξε. Εκεί θα έχει τη δυνατότητα να κάνει «Follow» το προφίλ το οποίο επέλεξε καθώς επίσης και να πατήσει το «About» κουμπί όπου θα περιέχονται κάποιες βασικές πληροφορίες σχετικά με τον επαγγελματικό χαρακτήρα του προφίλ (Εικόνα 7). Επίσης μπορεί να δει τις δημοσιεύσεις του προφίλ ή να πατήσει, στη μπάρα, το κουμπί που αντιστοιχεί στα «reviews» όπου θα μπορεί δει τις κριτικές που έχουν αφήσει άλλοι χρήστες για το κατάστημα (Εικόνα 8). Ακόμα, του δίνεται η δυνατότητα να αφήσει και αυτός την κριτική του, πατώντας στο πλαίσιο «give us your opinion» (Εικόνα 9).

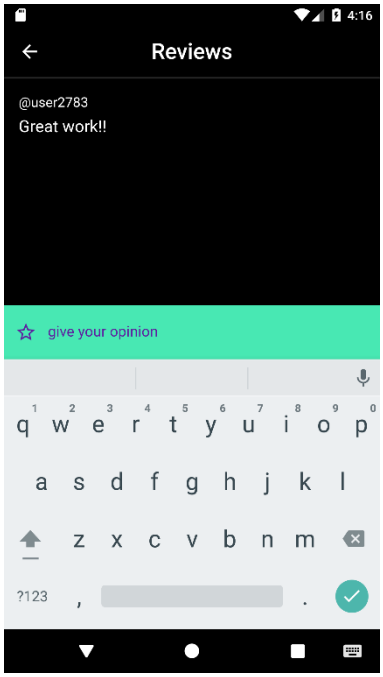
Σε κάθε προφίλ, όλη η σελίδα είναι «scrollable» (Εικόνα 10). Επίσης κάθε δημοσίευση μπορεί να επιλεχθεί πατώντας επάνω της με αποτέλεσμα η συγκεκριμένη δημοσίευση να γίνεται το επίκεντρο της οθόνης (Εικόνα 11) . Ακόμα δίνεται η δυνατότητα στον χρήστη να κάνει «μου αρέσει», να σχολιάσει αλλά και να αποθηκεύσει την αντίστοιχη δημοσίευση (Εικόνες 12, 13).



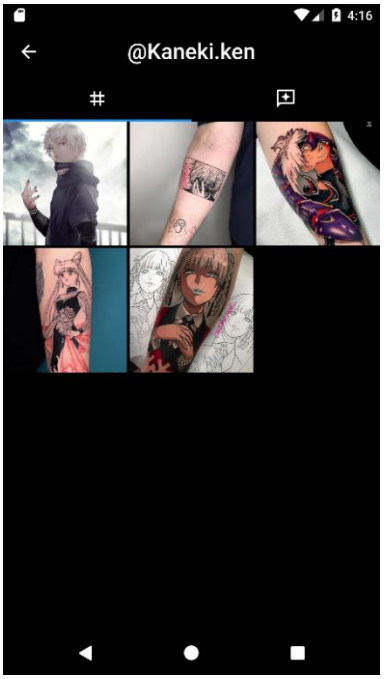
Εικόνα 7: Στην εικόνα επιλέξαμε για παράδειγμα το προφίλ "Kaneki.ken". Βλέπουμε τα κουμπιά "Follow" και "About".



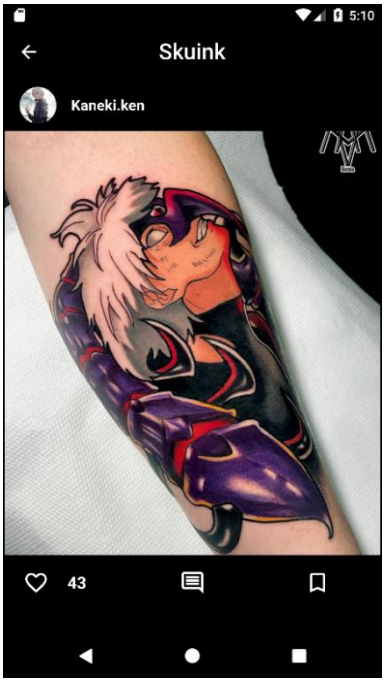
Εικόνα 8: Βλέπουμε το tab "Reviews". Για να αφήσει ο χρήστης την κριτική του πατάει στο "give us your opinion".



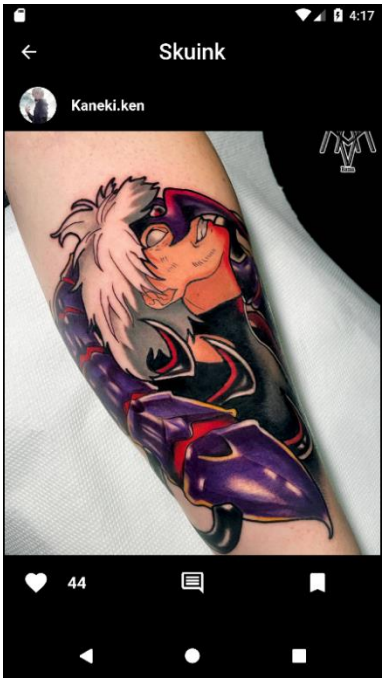
Εικόνα 9: Αφού έχει πατηθεί το "give us your opinion".



Εικόνα 10: Βλέπουμε ότι η σελίδα γίνεται scroll προς τα πάνω. Τα icons του tab bar δεν χάνονται αλλά παραμένουν στην κορυφή της σελίδας.

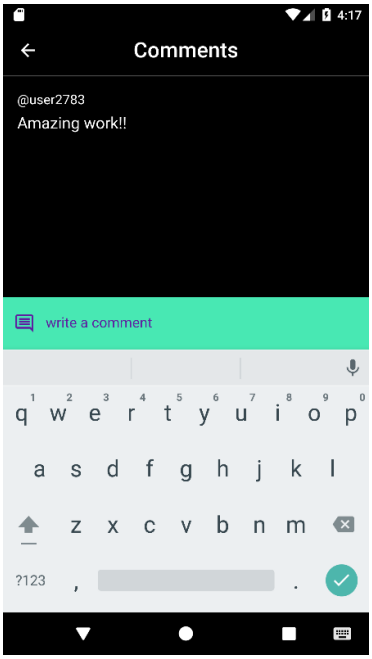


Εικόνα 11: Η οθόνη που θα έβλεπε ο χρήστης πατώντας σε μια δημοσίευση από το προφίλ "Kaneki.ken".

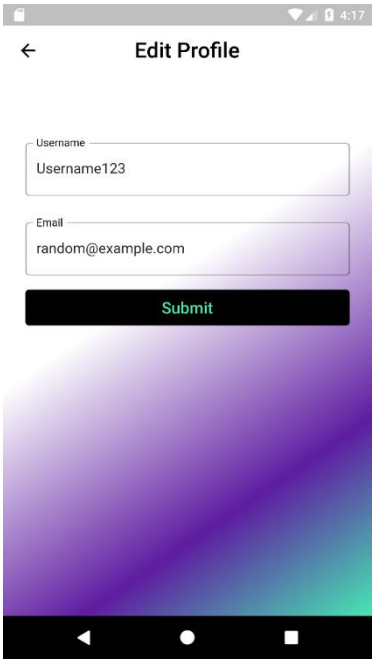


Εικόνα 12: Παρατηρούμε πως έχουν πατηθεί τα κουμπιά "Like" και "Save".

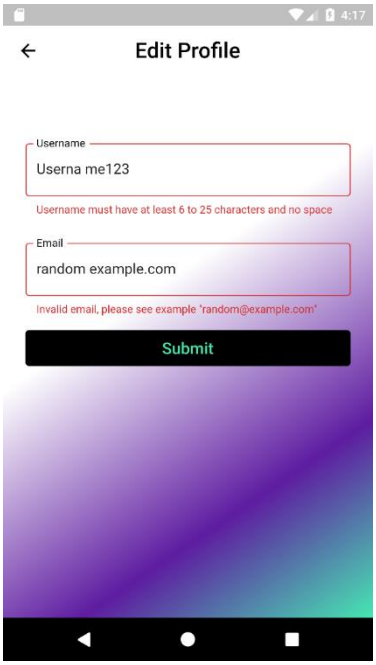
Μια ακόμα σελίδα που περιέχει λειτουργικότητα της εφαρμογής είναι η σελίδα «Edit Profile», εντός της σελίδας «Profile». Σε αυτή τη σελίδα ο χρήστης μπορεί να τροποποιήσει το όνομα χρήστη του (username) και τη διεύθυνση του ηλεκτρονικού του ταχυδρομείου (e-mail) που αρχικά θα είχε καταχωρήσει στην εφαρμογή κατά την εγγραφή του. Σε περίπτωση που τα νέα στοιχεία που καταχωρεί δεν καλύπτουν τις προϋποθέσεις που ζητούνται, τότε ο χρήστης θα λαμβάνει μήνυμα σφάλματος και τα στοιχεία δεν θα ανανεώνονται. (Εικόνες 14, 15)



Εικόνα 13: Οθόνη αφού πατήσουμε το κουμπί "comment".



Εικόνα 14: Οθόνη όταν επιλέξουμε "Profile" -> "Edit Profile".

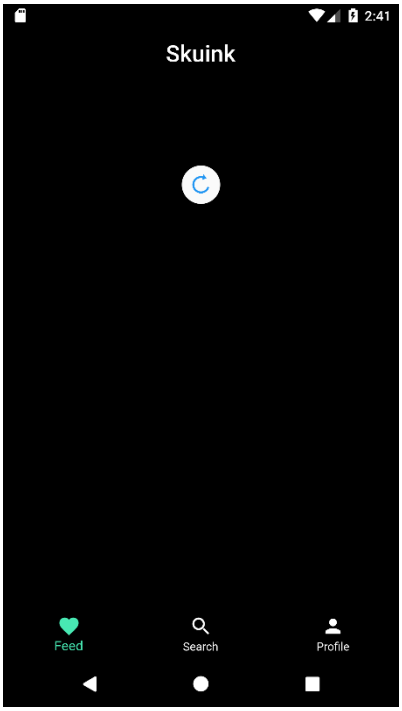


Εικόνα 15: Μήνυμα λάθους όταν δεν βάζουμε σωστά το username και το email.

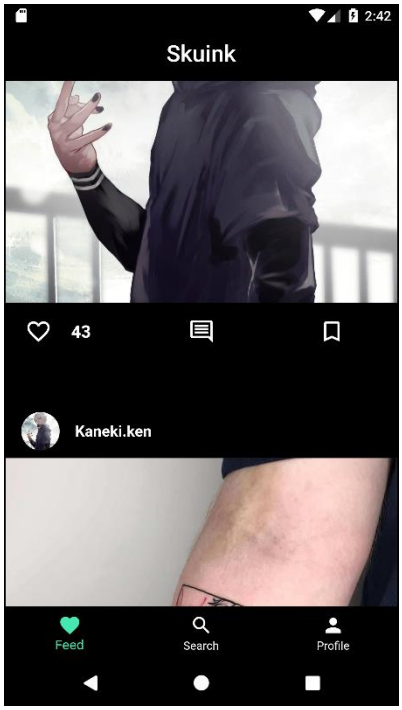
Ολοκληρώνοντας την παρουσίαση της εφαρμογής και των λειτουργιών της, μένει να δούμε τη σελίδα «Feed», όταν ο χρήστης τροποποιεί τη λίστα με τα προφίλ που ακολουθεί. Όπως αναφέραμε, αν ο χρήστης δεν ακολουθεί κανέναν λογαριασμό η σελίδα αυτή θα είναι κενή. Όταν θα επιλέξει να ακολουθήσει κάποιον λογαριασμό, επιστρέφοντας στη σελίδα «Feed», θα πρέπει να σύρει τη σελίδα προς τα κάτω έτσι ώστε η σελίδα να ανανεωθεί. Μόλις αυτή ανανεωθεί, θα δει στην οθόνη του, η οποία είναι «scrollable», τις δημοσιεύσεις από τον λογαριασμό που ακολούθησε. Γενικότερα κάθε φορά που ο χρήστης ακολουθεί ή σταματάει να ακολουθεί ένα προφίλ, επιστρέφοντας στη σελίδα «Feed», θα πρέπει να την ανανεώσει ώστε αυτή να ενημερώνεται και να δείχνει τα ανανεωμένα δεδομένα. (Εικόνες 16, 17)

Αξίζει να σημειώσουμε, πως ένα ακόμα στοιχείο της εφαρμογής, που είναι και αναμενόμενο για την ομαλή εμπειρία και λειτουργία της, είναι το γεγονός πως όλες οι λειτουργικότητες που περιέχονται εντός αυτής ενημερώνονται σωστά και συντονισμένα από οποιαδήποτε σελίδα και

αν ο χρήστης της πραγματοποιήσει. Δηλαδή, παραδείγματος χάρη, από οποιαδήποτε σελίδα της εφαρμογής και αν πατήσει κάποιος «μου αρέσει» σε μια δημοσίευση, η συγκεκριμένη δημοσίευση θα αναγράφει τον σωστό αριθμό μου αρέσει, σε οποιαδήποτε σελίδα και αν προβάλλεται, εντός της εφαρμογής.



Εικόνα 16: Όταν ο χρήστης ανανεώνει τη σελίδα "Feed" βλέπει στην οθόνη του ένα τυπικό animation ανανέωσης έως ότου αυτή ολοκληρωθεί.

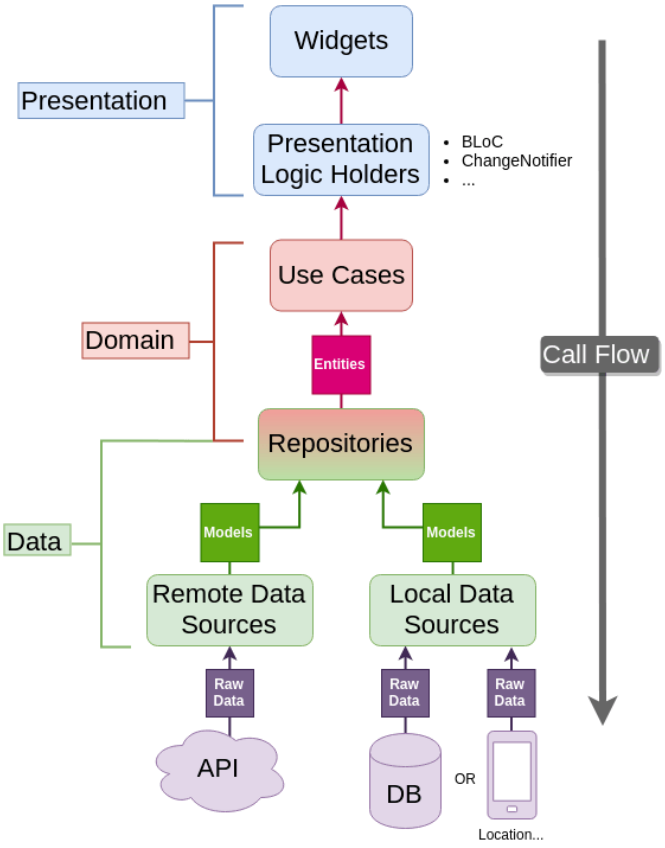


Εικόνα 17: Η σελίδα "Feed" αφού έχει ολοκληρωθεί η ανανέωση της σελίδας. Ο χρήστης μπορεί να σύρει προς τα κάτω την λίστα των δημοσιεύσεων.

Η αρχιτεκτονική που επιλέχθηκε για τον κώδικα

Ένα σημαντικό κομμάτι στην υλοποίηση ενός πρότζεκτ προγραμματιστικού χαρακτήρα είναι ο όσο το δυνατόν πιο καθαρός κώδικας. Με αυτόν τον όρο ουσιαστικά εννοούμε πως ο κώδικας ενός προγράμματος, εκτός από λειτουργικός, πρέπει να είναι δομημένος και οργανωμένος με τέτοιο τρόπο, που να τον καθιστά ευανάγνωστο, κατανοητό και να και όσο γίνεται πιο εύκολο στην τροποποίηση του. Για να επιτευχθεί αυτό, σιγουρά παίζει σημαντικό ρολό η ικανότητα και η εμπειρία ενός προγραμματιστή, υπάρχουν όμως και κάποιες συνηθισμένες πρακτικές δημιουργίας κώδικα που βοηθάνε σε μεγάλο βαθμό στην καθαρότητα του.

Στην παρούσα εφαρμογή ακολουθήθηκε η φιλοσοφία της καθαρής αρχιτεκτονικής, βασισμένη στον τρόπο που ορίστηκε από τον Robert C. Martin, στο blog του με όνομα «Uncle Bob». Η συγκεκριμένη αρχιτεκτονική ακολουθεί κάποιες καλές και διαδεδομένες πρακτικές για την υλοποίηση μιας εφαρμογής. Η βάση της λογικής της είναι ότι χωρίζει τον κώδικα σε τρία διαφορετικά επίπεδα, με το κάθε επίπεδο να είναι ανεξάρτητο από το άλλο (Εικόνα 18). Κάθε επίπεδο έχει διαφορετικό ρόλο, με το πρώτο επίπεδο να αφορά το κομμάτι του κώδικα που περιλαμβάνει την οπτική σχεδίαση της εφαρμογής, δηλαδή το κομμάτι που βλέπει και αλληλοεπιδρά ο χρήστης (presentation layer). Το κεντρικό επίπεδο περιέχει τις κύριες λογικές οντότητες της εφαρμογής (domain layer), ενώ το τελευταίο επίπεδο αποτελεί το κομμάτι της επικοινωνίας της εφαρμογής με τη βάση δεδομένων (data layer).



Εικόνα 18: Παρουσίαση της καθαρής αρχιτεκτονικής.

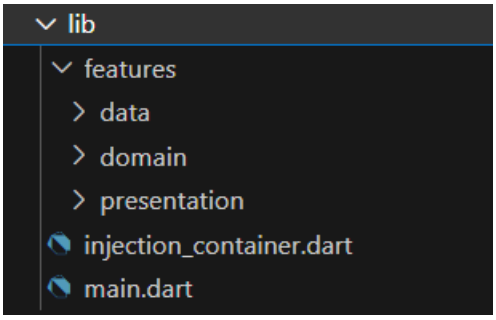
(πηγή: https://flutterawesome.com/a-sample-app-that-implement-uncle-bobs-clean-architecture-in-flutter/)

Η ανεξαρτησία μεταξύ των τριών επίπεδων αλλά και ο τρόπος που επικοινωνούν μεταξύ τους, βοηθάει πολύ στον έλεγχο (testing) του κώδικα σε κάθε βήμα και κατά επέκταση στην αποφυγή λαθών. Ακόμα όμως και αν προκύψει κάποιο λάθος, με αυτή τη μέθοδο είναι πιο εύκολο να απομονώσεις και να ελέγξεις τα επιμέρους κομμάτια ξεχωριστά.

Η ανεξαρτησία των επίπεδων σε αυτή τη μέθοδο προσφέρει ένα δυνατό ακόμα πλεονέκτημα. Επιτρέπει την πραγματοποίηση αλλαγών ή προσθηκών στων κώδικα χωρίς να επηρεάζονται τα άλλα κομμάτια του. Για παράδειγμα στην περίπτωση που θελήσει κάποια στιγμή κάποιος να αλλάξει τη βάση με την οποία ανταλλάζει δεδομένα η εφαρμογή του, αυτό που πρέπει να κάνει (με βάση και την εικόνα 18) είναι να αλλάξει τα μοντέλα που μεταφράζουν τα δεδομένα από τη γλώσσα της βάσης στη γλώσσα της εφαρμογής, χωρίς ο υπόλοιπος σκελετός της εφαρμογής να επηρεάζεται από αυτή την αλλαγή.

Απεικόνιση της αρχιτεκτονικής στην παρούσα εφαρμογή

Σε αυτό το σημείο, εφόσον πλέον έχει επιλεχθεί η αρχιτεκτονική που θα ακολουθηθεί, μπορεί να γίνει η παρουσίαση του τρόπου με τον οποίο υλοποιήθηκε η φιλοσοφία της, στη συγκεκριμένη εφαρμογή.

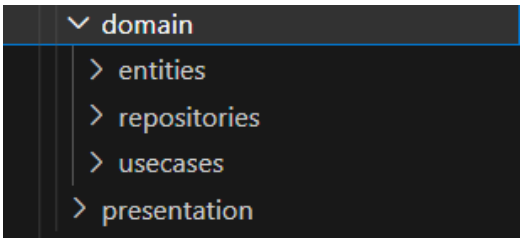


Εικόνα 19: Περιεχόμενο κυρίως φακέλου του κώδικα της εφαρμογής.

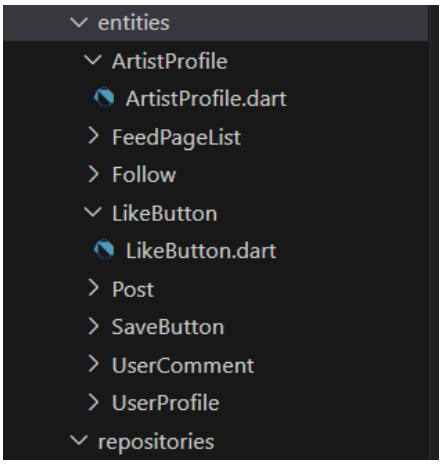
Ξεκινώντας, μέσα στον κύριο φάκελο που περιλαμβάνεται ο κώδικας της εφαρμογής, υπάρχει το αρχείο «main.dart» το οποίο είναι και το βασικό αρχείο από όπου ξεκινά να εκτελείται ο κώδικας, καθώς και το αρχείο «injection_container.dart» του οποίου ο ρόλος θα φανεί αργότερα. Επίσης υπάρχει και ένας ακόμα φάκελος με το όνομα «features», μέσα στον οποίο βρίσκονται τρεις νέοι φάκελοι με ονόματα «data», «domain» και «presentation», έτσι ώστε κάθε ένας από αυτούς να αντιπροσωπεύει το αντίστοιχο επίπεδο της αρχιτεκτονικής.

Domain layer

Όπως φάνηκε σύντομα προηγουμένως, το επίπεδο «domain» λειτουργεί σαν συνδετικός κρίκος μεταξύ των επιπέδων «data» και «presentation», κάνοντας το, ύπο μία έννοια τον πυρήνα της αρχιτεκτονικής. Ύπο αυτό το πρίσμα, αξίζει να συνεχίσει η παρουσίαση, πρώτα, με το περιεχόμενο αυτού του φακέλου. Κάτω, λοιπόν, από αυτόν τον φάκελο υπάρχουν τρεις νέοι φάκελοι με τα ονόματα «entities», «repositories» και «usecases». Ήδη, έως αυτό το σημείο, αρχίζει και γίνεται αντιληπτό πως ο τρόπος που έχουν οργανωθεί τα αρχεία, ακολουθούν την εικόνα (18).

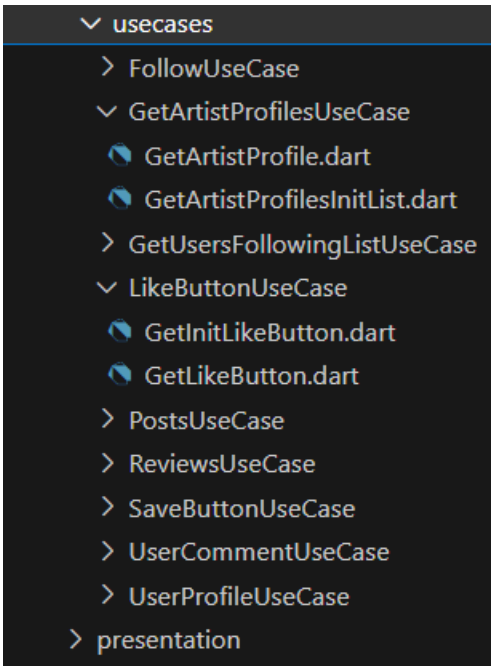


Εικόνα 20: Περιεχόμενο φακέλου «domain».



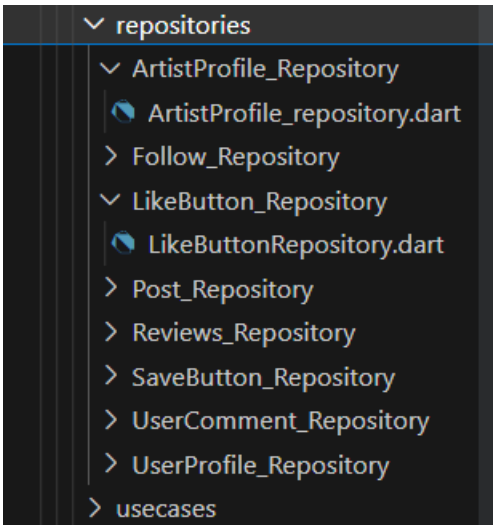
Εικόνα 21: Περιεχόμενο φακέλου «entities». Φαίνεται ενδεικτικά πως μέσα σε κάθε υποφάκελο υπάρχει το ομώνυμο αρχείο.

Στον φάκελο «entities» έχουν δημιουργηθεί κλάσεις οι οποίες αποτελούνται από δεδομένα τα οποία συγκροτούν λογικές οντότητες εντός της εφαρμογής. Τέτοιες οντότητες για παράδειγμα είναι ένα επαγγελματικό προφίλ στην εφαρμογή ή το κουμπί «like». Αυτές οι κλάσεις αποτελούνται ουσιαστικά μόνο από έναν constructor, χωρίς να περιλαμβάνουν κάποια λειτουργικότητα.



Εικόνα 22: Περιεχόμενο φακέλου «usecases». Το όνομα κάθε υποφακέλου υποδεικνύει το κομμάτι της εφαρμογής με το οποίο σχετίζονται. Ενδεικτικά φαίνονται οι κλάσεις εντός δύο υποφακέλων.

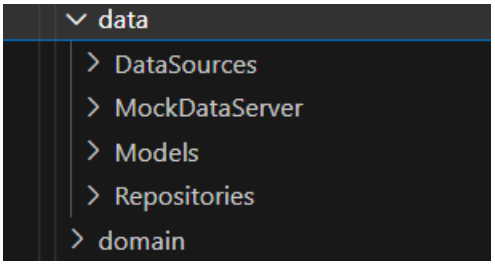
Στον φάκελο «usecases» βρίσκονται οι κλάσεις που στην ουσία κατέχουν τη μεθοδολογία που έχει επιλεχθεί σχετικά με τον τρόπο που θα καλούνται, θα αλλάζουν και θα αποθηκεύονται δεδομένα, ανάλογα με τις λειτουργίες που πραγματοποιούνται στην εφαρμογή. Κάθε κλάση σε αυτόν τον φάκελο σχετίζεται με μία μόνο διαδικασία της εφαρμογής που έχει να κάνει είτε με αλλαγές στα δεδομένα είτε με τον τρόπο που τα δεδομένα αυτά απεικονίζονται στην οθόνη. Για να μπορέσει να εξυπηρετήσει τον λόγο ύπαρξης της, κάθε κλάση καλεί την κατάλληλη συνάρτηση από την αντίστοιχη κλάση, στον φάκελο «repositories». Ο τρόπος που τα δεδομένα φτάνουν από το σχετικό repository στο σχετικό use case είναι μέσω του κατάλληλου αντικειμένου entity που έχει δημιουργηθεί.



Εικόνα 23: Περιεχόμενο φακέλου «repositories». Φαίνεται ενδεικτικά πως μέσα σε κάθε υποφάκελο υπάρχει το ομώνυμο αρχείο.

Στον φάκελο «repositories» έχουν δημιουργηθεί κλάσεις στις οποίες περιλαμβάνονται οι συναρτήσεις που θα εκτελούν τις επιθυμητές διαδικασίες σχετικά με την επεξεργασία και διαχείριση των δεδομένων της εφαρμογής. Όμως, όπως αναφέρθηκε προηγουμένως, πρέπει να υπάρχει ανεξαρτησία μεταξύ των επιπέδων «domain» και «data», κάτι που δεν φαίνεται ξεκάθαρα πως θα επιτευχθεί από την εικόνα (18). Έτσι λοιπόν, για να επιτευχθεί αυτός ο σκοπός, στα repositories του επιπέδου «domain» έχουν γίνει μόνο οι ορισμοί των διαφόρων συναρτήσεων, δηλαδή έχουν δημιουργηθεί, όπως ονομάζονται στα Αγγλικά, «abstract» κλάσεις. Η υλοποίηση αυτών των κλάσεων πραγματοποιείται στο επίπεδο «data». Με αυτόν τον τρόπο επιτυγχάνεται η ανεξαρτησία μεταξύ των δύο επιπέδων.

Data layer



Εικόνα 24: Περιεχόμενο φακέλου «data».

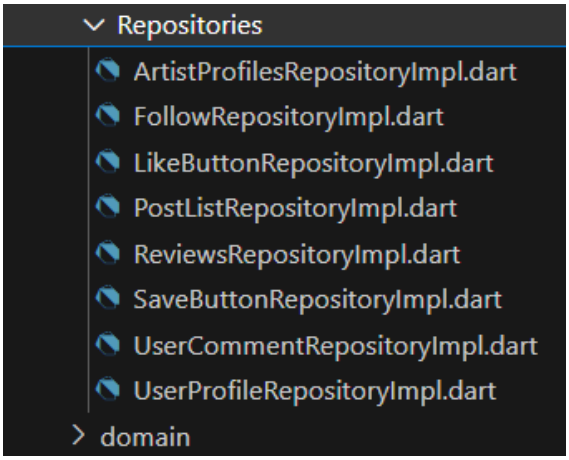
Στο επίπεδο «data» υλοποιείται το πρακτικό κομμάτι που σχετίζεται με την επικοινωνία της εφαρμογής με τις βάσεις δεδομένων. Κάτω από αυτόν τον φάκελο βρίσκονται οι φάκελοι, «DataSources», «MockDataServer», «Models» και «Repositories».

Στον φάκελο «Models», όταν η εφαρμογή θα έχει επικοινωνία με πραγματικές βάσεις δεδομένων, θα υπάρχουν οι κλάσεις εκείνες που θα είναι υπεύθυνες για τη μετατροπή των δεδομένων από τη γλώσσα της βάσης στη γλώσσα της εφαρμογής και αντίστροφα. Σε αυτό το στάδιο δεν έχει υλοποιηθεί ακόμα αυτό το κομμάτι με αποτέλεσμα ο φάκελος αυτός να είναι κενός.

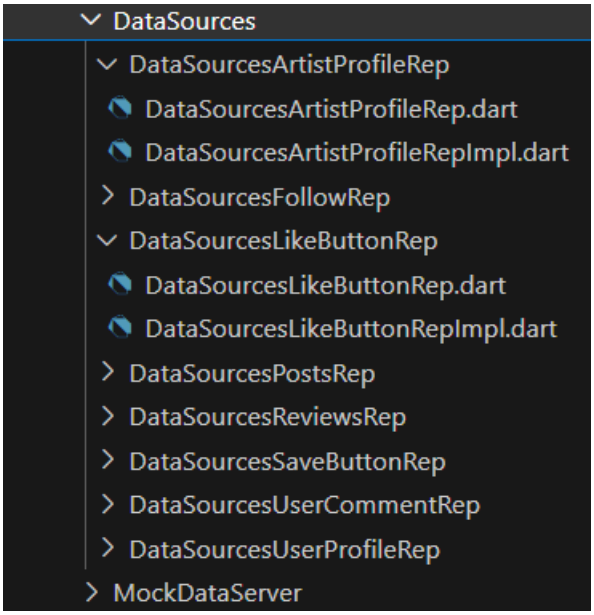
Ο φάκελος «MockDataServer» επί της ουσίας δεν είναι μέρος της αρχιτεκτονικής. Δημιουργήθηκε απλά για να μιμείται, με απλό τρόπο, τη λειτουργία κάποιας βάσης δεδομένων, έτσι ώστε να μπορούν να φανούν στην πράξη οι λειτουργίες της εφαρμογής.

Στον φάκελο «Repositories» γίνονται, όπως ονομάζεται στα Αγγλικά, override οι συναρτήσεις που ορίστηκαν στον ομώνυμο φάκελο στο «domain layer». Τα δεδομένα που συλλέγονται στα repositories, σε μορφή entities, διοχετεύονται από τον φάκελο «DataSources». Οι κλάσεις στα «DataSources» λειτουργούν ουσιαστικά σαν το εργαστήριο που συλλέγουν τα δεδομένα που

μεταφέρονται από και προς τις βάσεις δεδομένων και ακολουθώντας όλα τα απαραίτητα λογικά βήματα, συγκρίνοντας και μεταβάλλοντας δεδομένα, συγκροτούν τελικά τα entities που απαιτούνται από τα repositories.

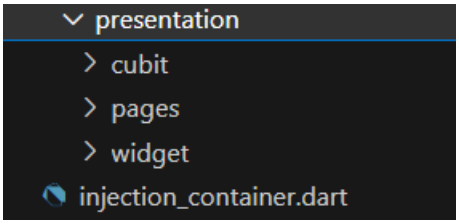


Εικόνα 25: Περιεχόμενο φακέλου «Repositories»



Εικόνα 26: Περιεχόμενο φακέλου «DataSources». Φαίνεται ενδεικτικά ο τρόπος που έχουν οργανωθεί οι κλάσεις. Σε κάθε υποφάκελο υπάρχουν δύο κλάσεις, μία abstract και μία που γίνεται η υλοποίηση (implementation).

Presentation layer

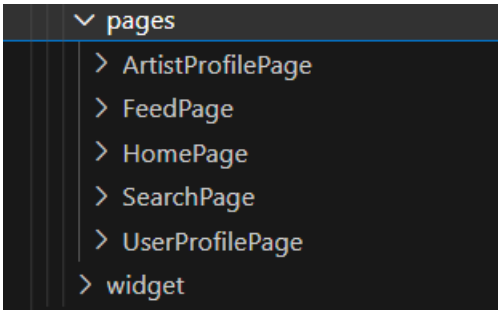


Εικόνα 27: Περιεχόμενο φακέλου «presentation»

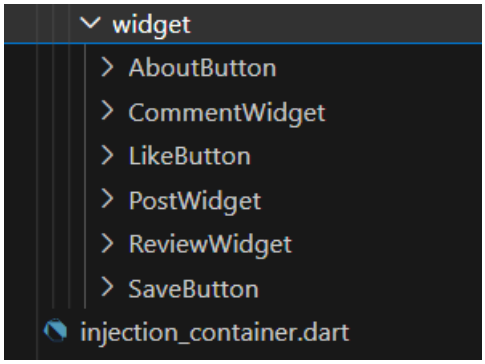
Το επίπεδο «presentation» αποτελεί το κομμάτι της εφαρμογής που έχει να κάνει με το οπτικό αποτέλεσμα που θα φτάσει στον χρήστη. Με άλλα λόγια, αφορά το οπτικό περιεχόμενο της εφαρμογής, ως προς την αισθητική της αλλά και ως προς τον τρόπο που θα γίνεται η αλληλεπίδραση του χρήστη με αυτήν. Έτσι, σε αυτόν τον φάκελο θα βρούμε τρεις νέους φακέλους με τα ονόματα «cubit», «pages» και «widget».

Ο φάκελος «pages» περιλαμβάνει τις κλάσεις που είναι υπεύθυνες για τον τρόπο που θα παρουσιάζονται όλα τα στοιχεία εικόνας σε κάθε σελίδα της εφαρμογής. Κάθε σελίδα της εφαρμογής είναι αποτέλεσμα μιας ξεχωριστής κλάσης. Ο φάκελος «widget» περιέχει κλάσεις που δημιουργούν στοιχεία εικόνας. Τέτοια στοιχεία εικόνας είναι για παράδειγμα το κουμπί «Like» ή μία δημοσίευση στην εφαρμογή. Τα widget ως οπτικά στοιχεία μπορεί να είναι μέρος ενός μεγαλύτερου widget. Ο συνδυασμός διαφόρων widget είναι ένα μέρος, από τη σύνθεση

συνολικά μιας σελίδας της εφαρμογής. Αξίζει να σημειωθεί πως οι κλάσεις αυτές που είναι υπεύθυνες για τη δημιουργία σελίδων και widget, δεν κατέχουν καμία λογική σχετικά με το κομμάτι του τρόπου διαχείρισης και επεξεργασίας των δεδομένων, παρά μόνο είναι υπεύθυνα για την οπτική απεικόνιση των δεδομένων που λαμβάνουν.



Εικόνα 28: Περιεχόμενο φακέλου «pages».



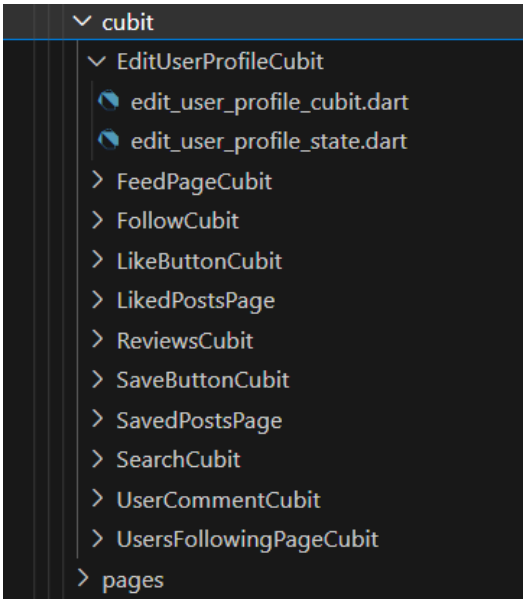
Εικόνα 29: Περιεχόμενο φακέλου «widget».

Η βιβλιοθήκη «bloc»

Στις περισσότερες σελίδες της εφαρμογής υπάρχουν λειτουργικότητες που οδηγούν τη σελίδα ή κάποιο widget της σελίδας στο να μεταβληθεί. Αυτές οι σελίδες και τα widget που επιδέχονται αλλαγές, έχουν έναν συγκεκριμένο αριθμό διαφορετικών καταστάσεων που μπορούν να λάβουν. Οι αλλαγές πραγματοποιούνται ανάλογα με τη δραστηριότητα που λαμβάνει χώρα εντός της εφαρμογής. Ένα απλό παράδειγμα είναι οι δύο απλές καταστάσεις που έχει το κουμπί «Like», οι οποίες είναι μία για όταν το κουμπί δεν έχει πατηθεί, δηλαδή δεν έχει γίνει ‘like’ και μία όταν αυτό έχει πατηθεί. Στην πράξη βέβαια και το κουμπί ‘like’ μπορεί να έχει και περισσότερες καταστάσεις, όπως μία κατάσταση για την περίπτωση αναμονής ή μία για την περίπτωση σφάλματος. Όσο όμως, μία εφαρμογή γίνεται πιο μεγάλη και πιο σύνθετη, με περισσότερες σελίδες και widgets, τόσο πιο δύσκολη αρχίζει και γίνεται η διαχείριση όλων των διαφορετικών καταστάσεων στα διάφορα σημεία της εφαρμογής και κατά επέκταση ο συγχρονισμός και η ομαλή λειτουργία της. Σε αυτό το σημείο έρχεται η βιβλιοθήκη «flutter_bloc» η οποία φτιάχτηκε ακριβώς με σκοπό να δώσει έναν πρακτικό τρόπο διαχείρισης καταστάσεων των στοιχείων μιας εφαρμογής. Μια επέκταση αυτής της βιβλιοθήκης είναι το «cubit» που χρησιμοποιήθηκε στην παρούσα εφαρμογή.

Το cubit δεν είναι τίποτα άλλο από μια πιο ελαφριά εκδοχή του bloc, για αυτό και επιλέχθηκε καθώς χρειάζεται συγγραφή λιγότερου κώδικα δίνοντας όμως τα επιθυμητά αποτελέσματα. Η φιλοσοφία του τρόπου λειτουργίας του είναι κάθε φόρα που προκύπτει μία ενέργεια στην εφαρμογή που θέλει να επιφέρει αλλαγή στην κατάσταση κάποιου στοιχείου της, η κατάλληλη συνάρτηση να καλείται μέσα στο cubit. Το cubit στέλνει ένα αίτημα στη βάση δεδομένων και περιμένει από αυτήν μία απάντηση (είτε το επιθυμητό αποτέλεσμα είτε κάποιο σφάλμα).

Ανάλογα με την απάντηση που λαμβάνει, το cubit ορίζει τη νέα κατάσταση και ανάλογα προβάλλεται η επιθυμητή οθόνη στον χρήστη.



Εικόνα 30: Περιεχόμενο φακέλου «cubit». Φαίνονται ενδεικτικά οι κλάσεις που περιλαμβάνει κάθε υποφάκελος.

Στην αρχιτεκτονική που ακολουθήθηκε στην παρούσα εφαρμογή, το cubit δεν επικοινωνεί κατευθείαν με τις όποιες βάσεις δεδομένων. Αντίθετα ενεργοποιεί το κατάλληλο use case για κάθε περίπτωση. Για να υπάρχει οργάνωση και να γίνεται καλύτερη διαχείριση καταστάσεων, φτιάχτηκε ένα διαφορετικό cubit για κάθε σελίδα ή widget που εναλλάσσει καταστάσεις. Κάθε cubit έχει δύο κλάσεις, μία που περιλαμβάνει όλες τις δυνατές καταστάσεις και μία που είναι υπεύθυνη να πυροδοτεί το κατάλληλο use case για κάθε κατάσταση.

Dependency Injection και βιβλιοθήκη «get_it»

Σε αυτό το σημείο έχει πρακτικά ολοκληρωθεί η παρουσίαση του τρόπου που εφαρμόστηκε η αρχιτεκτονική στην παρούσα εφαρμογή. Αξίζει όμως να γίνει ένα σχόλιο για τον σκοπό που εξυπηρετεί το αρχείο «injection_container.dart», που συναντήθηκε στην αρχή αυτού του κεφαλαίου. Ουσιαστικά, στην εφαρμογή, πολλές κλάσεις χρησιμοποιούν μεθόδους άλλων κλάσεων, με αποτέλεσμα να υπάρχει εξάρτηση μεταξύ κλάσεων. Αυτό το πρόβλημα λύνεται υλοποιώντας μία τεχνική που στον προγραμματισμό, στα Αγγλικά, λέγεται «Dependency Injection». Στην ουσία δηλαδή, η τεχνική που θα αναπτυχθεί θα πρέπει να επιτρέπει στις διάφορες κλάσεις να καλούν μεθόδους από άλλες κλάσεις, διατηρώντας όμως την ανεξαρτησία τους. Αν και υπάρχουν διάφοροι τρόποι να πετύχει κάποιος αυτήν την ανεξαρτησία, ο τρόπος που επιλέχθηκε σε αυτήν την εφαρμογή, που είναι και ίσως ο πιο διαδεδομένος τρόπος στην κοινότητα του flutter, ήταν να χρησιμοποιηθεί η βιβλιοθήκη «get_it». Με τη βοήθεια αυτής της βιβλιοθήκης, αρκεί σε ένα αρχείο να καταχωρήσουμε (να τις κάνουμε register) όλες τις κλάσεις που έχουν εξάρτηση από άλλες κλάσεις, χρησιμοποιώντας τις κατάλληλες μεθόδους της get_it. Ακριβώς αυτή η διαδικασία έχει πραγματοποιηθεί στο αρχείο «injection_container.dart».

Γιατί επέλεξα αυτή την εφαρμογή

Έχοντας παρουσιάσει την εφαρμογή και αφού μιλήσαμε για διάφορα, σχετικά ζητήματα, επέλεξα να κλείσω την παρούσα εργασία αναφέροντας λίγα λόγια σχετικά με την επιλογή της εφαρμογής, ως προς το περιεχόμενο.

Αρχικά ένας βασικός άξονας για την επιλογή περιεχομένου εφαρμογής ήταν πως ήθελα αυτή, να έχει στοιχεία από κάποια κατηγορία εφαρμογών που συναντά το ενδιαφέρον στον πραγματικό κόσμο. Έτσι κατέληξα πως μια εφαρμογή με στοιχεία από μέσα κοινωνικής δικτύωσης θα ήταν μια ωραία επιλογή, καθώς παρατηρούμε πως τα μέσα αυτά, διαχρονικά, όχι μόνο δεν έχασαν καθόλου την απήχηση τους στον πληθυσμό αλλά αντίθετα το ενδιαφέρον αυτό είναι ολοένα και αυξανόμενο. Το αποτέλεσμα είναι από την μια τα υπάρχοντα μέσα κοινωνικής δικτύωσης να αναπτύσσονται συνεχώς, ενώ από την άλλη, κατά καιρούς, να δημιουργούνται καινούργια. Με βάση τα παραπάνω, η ιδέα του να ασχοληθώ με ένα μικρό κομμάτι από ένα τέτοιο πρότζεκτ μου φάνηκε αρκετά ενδιαφέρουσα, παραγωγική και χρήσιμη για την εμπειρία που θα μου προσέφερε.

Ο άλλος άξονας στον οποίο στήριξα την επιλογή αυτής της εφαρμογής ήταν πως ήθελα να βασίζεται σε μια ιδέα, που με βάση την κρίση μου, θα είχε κάποιες βάσιμες ελπίδες να σταθεί στην πραγματική αγορά, αν υλοποιούνταν ως μέρος ενός ολοκληρωμένου πρότζεκτ, από κάθε άποψη. Τα τελευταία χρόνια, συναντάμε το φαινόμενο όπου αναπτύσσονται εφαρμογές με σκοπό, ύπο μια έννοια, να “στεγάσουν” έναν ολόκληρο κλάδο. Εφαρμογές που κυριαρχεί αυτή η σκέψη είναι, για παράδειγμα, οι εφαρμογές διανομής φαγητού, εφαρμογές για τη βραχυχρόνια μίσθωση χώρων και ακίνητων, εφαρμογές για υπηρεσίες ταξί, εφαρμογές για εύρεση γιατρών και πολλές παρόμοιες ακόμα. Πάνω σε αυτή τη λογική, σκέφτηκα πως ένας κλάδος που παρουσιάζει διαχρονικά μια σταθερή και κατά περιόδους αυξανομένη δυναμική και συνεχώς εξελίσσεται είναι η δερματοστιξία. Εφόσον είδα ότι ο κλάδος αυτός δεν έχει κάποιο βασικό κοινό σημείο αναφοράς, κατέληξα τελικά στην εφαρμογή που σας παρουσίασα.

Διαδικτυακές πήγες

1. https://source.android.com/
2. https://flutter.dev/
3. https://opensource.guide/el/starting-a-project/
4. https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html