ΑΝΑΠΤΥΞΗ ΚΑΙ ΣΧΕΔΙΑΣΗ ΚΙΝΗΤΩΝ ΕΦΑΡΜΟΓΩΝ

REPORT

Υπεύθυνος καθηγητής: κ. Μάριος Φραγκούλης

Ιανουάριος 2025

Γουργιώτη Ζωή 8210028

Κουνάρα Φωτεινή-Ανδριανή 8210066

Νεραντζάκης Ιάσονας 8210103

Ρούση Κατερίνα 8210130

Εφαρμογή:

Η εφαρμογή μας αποτελεί ένα σύστημα κράτησης εστιατορίων. Για μια συγκεκριμένη αλυσίδα καταστημάτων, Juicy, ο χρήστης μπορεί να κάνει κράτηση σε οποιοδήποτε κατάστημα επιθυμεί. Ο χρήστης έχει την επιλογή να δημιουργήσει ή να συνδεθεί στο λογαριασμό του. Με τη δυνατότητα αυτή μπορεί στην οθόνη Profile να δει το ιστορικό των κρατήσεων του με πληροφορίες όπως το κατάστημα κράτησης, διεύθυνση, άτομα κράτησης.

Με την είσοδο του στην εφαρμογή ο χρήστης εισέρχεται στην αρχική σελίδα, Homepage. Στο πάνω μέρος της σελίδας βλέπει το όνομα χρήστη του. Στη συγκεκριμένη σελίδα ο χρήστης επιλέγει την ημέρα που επιθυμεί να κάνει κράτηση και τον αριθμό των ατόμων. Με βάση τις επιλογές αυτές εμφανίζονται τα διαθέσιμα καταστήματα. Επίσης στο πάνω μέρος της οθόνης ο χρήστης μπορεί να δει διάφορες προσφορές που ισχύουν στα καταστήματα. Ο χρήστης πατώντας πάνω στα offers ανακατευθύνεται στην επόμενη σελίδα, αυτή με τις πληροφορίες των καταστημάτων. Ωστόσο αν παραδείγματος χάρη η προσφορά για ένα κατάστημα ισχύει για τη μέρα Δευτέρα αλλά το κατάστημα δεν είναι διαθέσιμο τη συγκεκριμένη μέρα, ο χρήστης με ένα pop up ενημερώνεται για αυτό. Συνεπώς ο χρήστης ανακατευθύνεται στην επόμενη σελίδα είτε πατώντας στα offers είτε το κουμπί Book που υπάρχει σε κάθε διαθέσιμο κατάστημα.

Η σελίδα που ακολουθεί περιέχει τις πληροφορίες καταστήματος. Ο χρήστης μπορεί να δει τη διεύθυνση του καταστήματος, τις ώρες λειτουργίας, φωτογραφία του καταστήματος και τις διαθέσιμες ώρες που μπορεί να πραγματοποιήσει την κράτηση. Επίσης εμφανίζονται και αστεράκια τα οποία υποδεικνύουν τη βαθμολογία του καταστήματος η οποία προκύπτει από τις αξιολογήσεις. Από το σημείο αυτό ο χρήστης μπορεί να επισκεφτεί τη σελίδα των αξιολογήσεων. Στο κάτω μέρος της σελίδας (με τις πληροφορίες καταστήματος) εμφανίζεται η διεύθυνση του καταστήματος πάνω στο χάρτη, πληροφορία που αντλείται από το API του google maps. Ο χρήστης επιλέγοντας την ώρα που επιθυμεί να κάνει την κράτηση και πατώντας το κουμπί Proceed with Reservation ανακατευθύνεται στην επόμενη σελίδα. Στην επόμενη σελίδα ο χρήστης επιβεβαιώνει την κράτηση. Βλέπει συμπληρωμένα το τηλέφωνο του και το username του και επικυρώνει την κράτηση. Επιβεβαιώνοντας την, η κράτηση πλέον έχει πραγματοποιηθεί και μπορεί να προβληθεί από το προφίλ του. Από το προφιλ του μπορεί επίσης να ανακατευθυνθεί στη σελίδα των αξιολογήσεων στην οποία μπορεί να γράψει κριτική ή να δει ήδη υπάρχουσες κριτικές.

Ανάλυση User Interface και User Experience στην εφαρμογή Juicy:

Η εφαρμογή Juicy αναπτύχθηκε με στόχο να παρέχει στους χρήστες μια φιλική, αισθητικά ευχάριστη και λειτουργική εμπειρία για την κράτηση τραπεζιών στα εστιατόρια μας. Στην παρούσα ενότητα αναλύονται οι βασικές σχεδιαστικές επιλογές και η υλοποίηση των κύριων χαρακτηριστικών της εφαρμογής.

User Interface

Το UI της εφαρμογής σχεδιάστηκε με έμφαση στη χρηστικότητα και την απλότητα, επιλέγοντας μοντέρνα στοιχεία γραφικών:

- Χρωματική παλέτα: Επιλέχθηκαν φωτεινές αποχρώσεις, όπως πορτοκαλί (#FFA 726) για κουμπιά δράσης και γκρι για τα βοηθητικά κείμενα, εξισορροπώντας την αντίθεση και την αισθητική αρμονία.
- Καθαρή τυπογραφία, με ισορροπία ανάμεσα σε έντονους (bold) τίτλους και περιγραφικό κείμενο για καλή ανάγνωση, όπως οι τίτλοι στα widgets FoodCard και RestaurantCard που τονίζουν σημαντικές πληροφορίες, ενώ το περιγραφικό κείμενο έχει μικρότερο βάρος, βελτιώνοντας την αναγνωσιμότητα.
- Προσαρμοστικότητα και προσβασιμότητα: Χρησιμοποιούνται LazyColumn και LazyRow που προσφέρουν ομαλή κύλιση και εξασφαλίζουν την λειτουργική εμφάνιση των στοιχείων της κάθε λίστας στις οθόνες, ανεξαρτήτως μεγέθους.
- Χρήση Column, Row, και Box για δημιουργία δυναμικών layouts που προσαρμόζονται στις ανάγκες της κάθε οθόνης.
- Buttons προσφέρουν εύκολη επιλογή ώρας, πλοήγηση προς την κράτηση ή επιβεβαίωση κράτησης.
- Οπτικά Εφέ όπως γεμάτα ή άδεια αστεράκια για αξιολόγηση ή εμφάνιση του rate.
- Error dialog το οποίο είναι ειδικά σχεδιασμένο να διατηρεί τη θετική ατμόσφαιρα της εφαρμογής. Συγκεκριμένα σε περιπτώσεις όπου τα καταστήματα μας είναι πλήρη, εμφανίζεται ένα φιλικό μήνυμα: "Wow, this deal’s on fire! Check another day— we’d love to Juicy serve you ;)", που ενισχύει τον θετικό χαρακτήρα της εφαρμογής.

Αξίζει να σημειωθεί πως κατά τον σχεδιασμό και την ανάπτυξη της εφαρμογής αποφασίστηκε να μην χρησιμοποιηθούν βίντεο καθώς η εφαρμογή επικεντρώνεται κυρίως στην κράτηση τραπεζιών, την παρουσίαση του εστιατορίου με την προβολή βασικών πληροφοριών όπως ώρες λειτουργίας, τοποθεσία και διαθέσιμες υπηρεσίες και την πραγματοποίηση της κράτησης. Οι εικόνες παρέχουν επαρκή οπτική αναπαράσταση για να καλύψουν τις ανάγκες του χρήστη, χωρίς να προσθέτουν περιττά στοιχεία που δεν εξυπηρετούν τον βασικό σκοπό της εφαρμογής. Οι εικόνες αποθηκεύονται τοπικά στο res/drawable.

User Experience

Το UX επικεντρώνεται στη δημιουργία μιας ευχάριστης ροής και απλότητας στη διαχείριση κρατήσεων. Κύρια χαρακτηριστικά του UX περιλαμβάνουν:

- Εξατομίκευση: Κατά την είσοδο στην αρχική σελίδα, εμφανίζεται το μήνυμα: "Welcome back, [όνομα χρήστη]! Επίσης, εξατομίκευση παρατηρείται και πριν την πραγματοποίηση της κράτησης που προσυμπληρώνονται τα στοιχεία του εκάστοτε χρήστη.
- Αλληλεπιδραστικότητα: Η χρήση στοιχείων όπως το pop-up μήνυμα επιτυχίας κατά την πραγματοποίηση της κράτησης, καθώς και η χρήση φίλτρων ημέρας και αριθμού ατόμων στην κεντρική μας οθόνη, που μέσω του DaySelector και του PersonsSelector, επιτρέπει στον χρήστη να πλοηγηθεί άμεσα και να προσαρμόσει άμεσα τις παραμέτρους αναζήτησης.
- Ενημέρωση σε πραγματικό χρόνο: Με βάση τις επιλογές των φίλτρων ημέρας και αριθμού ατόμων και χρησιμοποιώντας την προσέγγιση LaunchedEffect και τα coroutines, τα δεδομένα μας ενημερώνονται δυναμικά.

Composable Design Patterns

Η εφαρμογή βασίζεται στη χρήση του Jetpack Compose, που παρέχει ευέλικτα και εύχρηστα εργαλεία για τον ορισμό των UI στοιχείων:

- Το Material3 διασφαλίζει μοντέρνο και φιλικό σχεδιασμό.
- Τα Cards περιλαμβάνουν πληροφορίες καταστημάτων και προσαρμοσμένα στοιχεία.
- Όλες οι σελίδες υλοποιήθηκαν με επαναχρησιμοποιούμενα components που συνδυάζουν αισθητική και λειτουργικότητα.
- Χρήση ViewModels για να διατηρεί τοπικά δεδομένα και να χειρίζεται συλλογές δεδομένων όπως Stores, Reviews,Bookings Users και Slots.
- Χρήση collectAsState για real-time προβολή δεδομένων. Επιπλέον με το remember και το collectAsState, επιτυγχάνεται δυναμική διαχείριση δεδομένων, μειώνοντας τις περιττές κλήσεις στο ViewModel.Το ViewModel χρησιμοποιεί το Flow API για την παρακολούθηση της βάσης δεδομένων σε πραγματικό χρόνο. Κάθε αλλαγή στις κριτικές ενημερώνει άμεσα τη λίστα μέσω του collectAsState στους composables. Ταυτόχρονα, η χρήση του viewModelScope.launch, επιτρέπει τον χειρισμό πόρων χωρίς να δεσμεύεται το UI thread.

Coroutines

Τα Coroutines στην εφαρμογή μας αξιοποιούνται σε διαφορετικά σημεία, εξασφαλίζοντας ότι η εμπειρία χρήστη παραμένει ομαλή και το UI δεν μπλοκάρει κατά την εκτέλεση χρονοβόρων λειτουργιών. Κάποιες από τις βασικές χρήσεις περιλαμβάνουν:

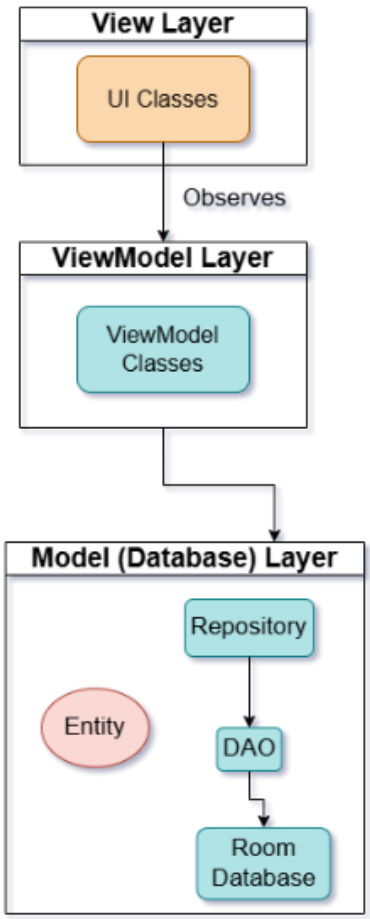
- Δυναμική ενημέρωση: Ο συνδυασμός rememberCoroutineScope και LaunchedEffect εξασφαλίζει τη λήψη δεδομένων χωρίς να επηρεάζεται η απόδοση. Για παράδειγμα η εμφάνιση των εστιατορίων στην κεντρική με βάση τη διαθεσιμότητα γίνεται ασύγχρονα, αποφεύγοντας καθυστερήσεις και μπλοκαρίσματα.

Αρχιτεκτονική:

Το μοντέλο αρχιτεκτονικής στο οποίο βασιστήκαμε ώστε να στηθεί το Juicy App είναι το MVVM μοντέλο (Model View View Model). Σε αυτό, μπορούν να παρατηρηθούν 3 διακριτά Layers αρχιτεκτονικής. Το πρώτο είναι το view layer, το οποίο περιέχει όλες τις κλάσεις και τα αρχεία του user interface και είναι αυτό με το οποίο αλληλεπιδρά ο χρήστης.

Στη συνέχεια, ακολουθεί το View Model Layer, το οποίο λειτουργεί ως ενδιάμεσος διαχειριστής της κατάστασης μεταξύ του UI και των δεδομένων που υπάρχουν στην τοπική βάση. Στις ViewModel κλάσεις (συνήθως υπάρχει μία για κάθε λειτουργική κλάση δεδομένων που έχει οριστεί), γίνεται δυναμική διατήρηση των προσωρινών δεδομένων του UI και πραγματοποιείται συνεχής παρακολούθηση για αλλαγές και ενημερώσεις καταστάσεων στο session.

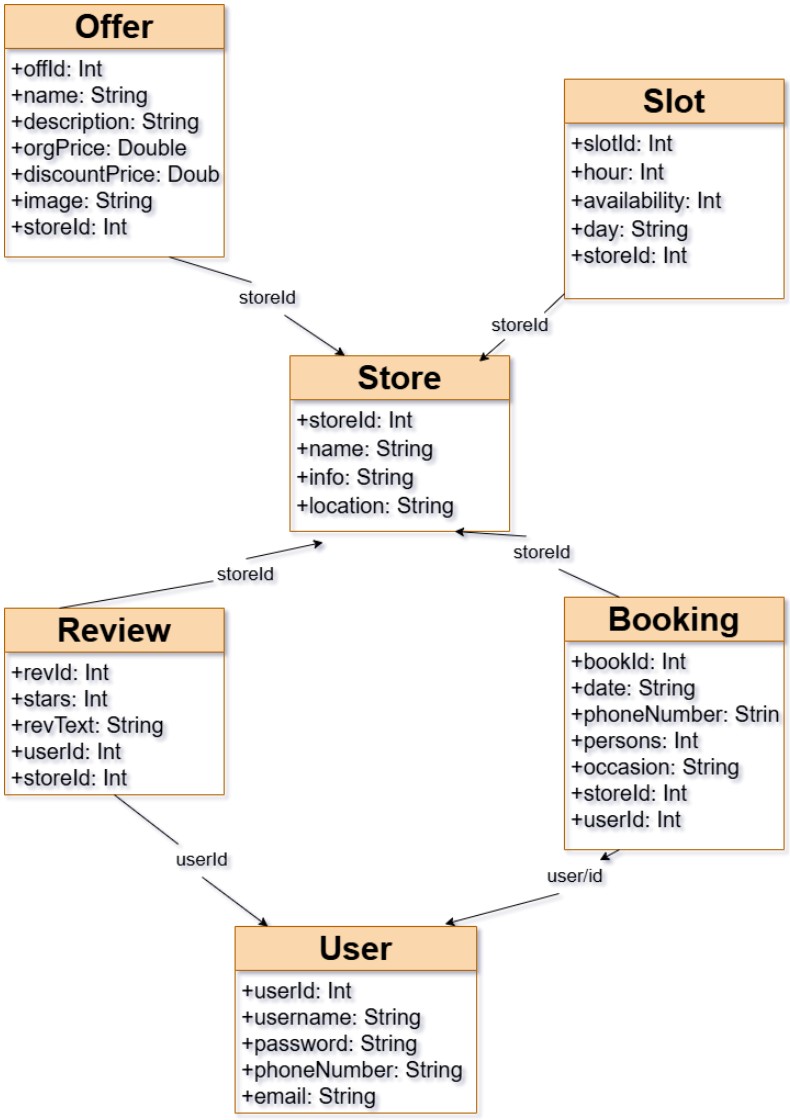
Τέλος, στο χαμηλότερο επίπεδο βρίσκεται το Model Layer, το οποίο στη συγκεκριμένη εφαρμογή υλοποιείται ως Database Layer. Εκεί, βρίσκεται το repository ώστε να παρέχεται μία αφαίρεση στη βάση, η οποία έχει υλοποιηθεί με χρήση της βιβλιοθήκης Room Database, η οποία στην ουσία παρέχει μια εύκολα διαχειρίσημη και αφαιρετική υλοποίηση της SQLite.



Αυτή λειτουργεί ως εξής: Κάθε πίνακας υλοποιείται ως μία οντότητα (Entity), στην οποία δηλώνονται τα πεδία και τα κλειδιά τα οποία διαθέτει για σύνδεση με άλλους πίνακες. Για πρόσβαση στα δεδομένα, έχουν δημιουργηθεί κλάσεις DAO (Data Access Object) τύπου διεπαφής (Interface), στις οποίες έχουν υλοποιηθεί όλα τα SQL Queries που χρειάζονται στη λειτουργικότητα της εφαρμογής. Αυτά τα queries συνδέονται με συναρτήσεις Kotlin, οι οποίες είναι υλοποιημένες στα αντίστοιχα ViewModels.

Παραπάνω φαίνεται ένα υψηλού επιπέδου διάγραμμα της βασικής αρχιτεκτονικής του app που περιγράφηκε.

Όσον αφορά στην αρχιτεκτονική και τον τρόπο δόμησης της Room βάσης δεδομένων μας, παρακάτω φαίνεται το schema της με όλες τις οντότητες που ορίστηκαν και τα πεδία που αυτές περιέχουν.



Κεντρική οντότητα αποτελεί το κατάστημα (Store). Με αυτό συνδέεται ο πίνακας Slot ο οποίος υλοποιεί και παρέχει την διαθεσιμότητα για κάθε κατάστημα σε επίπεδο εβδομάδας, σε μορφή time slots. Πιο συγκεκριμένα, η διάρκεια των κρατήσεων έχει αποφασιστεί στις 2 ώρες, και για κάθε κατάστημα αποθηκεύεται η ώρα έναρξης κάθε slot (για παράδειγμα αν ένα κατάστημα λειτουργεί 14.00 έως 24.00 τα slots θα δηλωθούν ως 2, 4, 6, 8, 10).

Το πόσες φορές θα μπορεί να γίνει reserve ένα slot σε ένα συγκεκριμένο κατάστημα εξαρτάται από το πλήθος των ατόμων που απαρτίζουν την κάθε κράτηση καθώς και από το πλήθος των τραπεζιών που το κατάστημα διαθέτει. Αξίζει να αναφερθεί πως, από τη στιγμή που η εφαρμογή είναι πλήρως λειτουργική μόνο από το μέρος των πελατών και όχι των καταστημάτων, μεταβλητές όπως η χωρητικότητα του κάθε

καταστήματος και τα μεγέθη των τραπεζιών που διαθέτει (δηλαδή το πόσα άτομα μπορεί να χωρέσει το καθένα) έχουν θεωρηθεί δεδομένες και έχουν δηλωθεί καρφωτά.

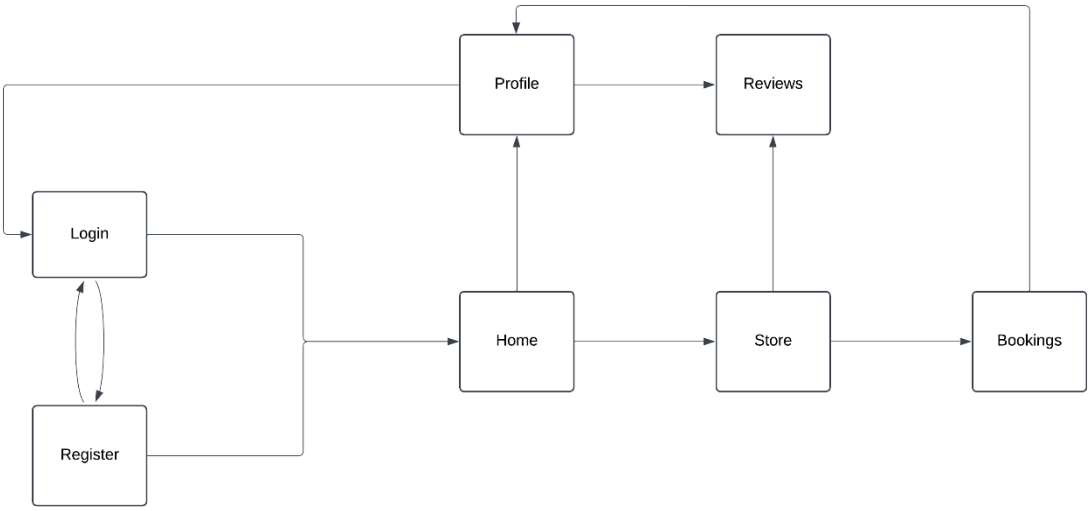
Πλοήγηση:

Η πλοήγηση εντός της εφαρμογής έχει υλοποιηθεί με την χρήση Jetpack Navigation. Το Jetpack Navigation διαχειρίζεται την πλοήγηση εντός της εφαρμογής, επιτρέποντας τη μετάβαση μεταξύ διαφορετικών οθονών (composable functions) με τρόπο καθαρό και κεντρικοποιημένο.

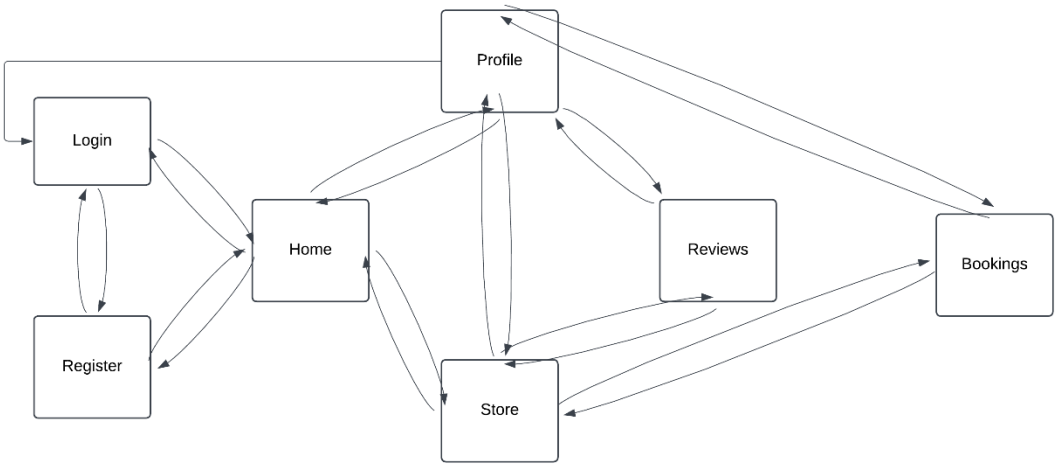
Η πλοήγηση από άποψη διεπαφής είναι εφικτή είτε μέσω της μπάρας πλοήγησης που υπάρχει στο τέλος της κάθε σελίδας (για πλοήγηση σε Home η Profile), είτε με κουμπία είτε με clickable icons. Επίσης, υπάρχει και δυνατότητα πλοήγησης προς τα πίσω σε κάθε σελίδα, είτε μέσω της μπάρας πλόηγησης είτε με clickable icon (βελάκι). Φυσικά, ο χρήστης μπορεί να πάει στην προηγούμενη σελίδα και με την χρήση του συστήματος πλοήγησης της συσκευής του.

Δημιουργείται ένας κεντρικός κόμβος (NavHost) που ορίζει τα διαθέσιμα routes της εφαρμογής και τις αντίστοιχες composable οθόνες. Ο NavController ελέγχει την πλοήγηση και την κατάσταση της στοίβας των οθονών. Μέσω navArgument, μεταφέρονται δυναμικά οι παραμέτροι μεταξύ των οθονών, επιτρέποντας την ευέλικτη προσαρμογή της πλοήγησης. Για να γίνει εφικτή η μεταφορά των παραμέτρων και για να οριστούν τα routes, χρησιμοποιείται και η κλάση Screen. Πιο συγκεκριμένα, μέσω της κλάσης Screen, κάθε οθόνη αντιπροσωπεύεται από ένα μοναδικό data object με προκαθορισμένο route, ενώ η μέθοδος withArgs() επιτρέπει τη δημιουργία δυναμικών πλοηγήσεων με παραμέτρους. Αυτή η προσέγγιση απλοποιεί την πλοήγηση, καθιστά τον κώδικα πιο ευανάγνωστο και διασφαλίζει τη συνοχή μεταξύ των οθονών της εφαρμογής. Σε κάθε σελίδα (δηλαδή και σε κάθε αρχείο του project), υπάρχει ένα composable function το οποίο αναλαμβάνει τον ορισμό της πλοήγησης για την σελίδα εκείνη.

Παρακάτω, φαίνεται η φυσιολογική ροή πλοήγησης των σελίδων, δηλαδή όλα τα σενάρια πλοήγησης, χωρίς να περιλαμβάνουμε την χρήση μπάρας πλοήγησης (στο κάτω μέρος της κάθε σελίδας) και χωρίς να περιλαμβάνουμε την χρήση κουμπιών για πλοήγηση προς τα πίσω.



Εάν συμπεριλάβουμε όλες τις δυνατές περιπτώσεις πλοήγησης, τότε προκύπτει το παρακάτω διάγραμμα .



Προκλήσεις:

Kotlin

Κανένα από τα μέλη της ομάδας δεν γνώριζε Kotlin ούτε είχε ασχοληθεί ποτέ με την ανάπτυξη native mobile app. Επομένως, στη διαδικασία της ανάπτυξης αυτής της εφαρμογής συναντήσαμε ορισμένες προκλήσεις, άλλες μεγαλύτερες και άλλες μικρότερες, τις οποίες φέραμε εις πέρας με τον τρόπο και τα εργαλεία που διαθέταμε.

Βάση

Αρχικά, ένα από τα μεγαλύτερα ζητήματα που μας απασχόλησαν ήταν η υλοποίηση της Room βάσης δεδομένων, και πιο συγκεκριμένα ο τρόπος διατήρησης των δεδομένων που προϋπήρχαν και που εισάγονταν μετά το κλείσιμο της εφαρμογής. Η

αρχική προσέγγιση ήταν μία συνάρτηση που κάθε φορά που η εφαρμογή θα έτρεχε θα έσβηνε τα υπάρχοντα ελλιπή δεδομένα και θα έκανε εκ νέου Insert τις προκαθορισμένες τιμές. Ωστόσο, επειδή αυτή η προσέγγιση ήταν μη πρακτική, και από τη στιγμή που δεν διαθέταμε τους πόρους και το χρόνο να κάνουμε τη βάση online, αποφασίστηκε η εισαγωγή δεδομένων στο αρχείο AppDatabase με SQL Queries με την εντολή onCreate, δηλαδή κάθε φορά που η εφαρμογή θα εγκαθίσταται στο smartphone ή στο vm στο οποίο τρέχει.

Ένα άλλο ζήτημα που μας απασχόλησε ήταν ο τρόπος με τον οποίο οι πίνακες θα συνδέονται μεταξύ τους (πως δηλαδή θα δομηθεί το σχήμα της βάσης) και συγκεκριμένα στην περίπτωση της διαχείρισης της διαθεσιμότητας των καταστημάτων. Δοκιμάστηκαν πολλοί διαφορετικοί τρόποι που αξιοποιούσαν είτε προγραμματιστικές τεχνικές σε Kotlin είτε διαχείριση σχέσεων πινάκων με κλειδιά σε SQL, ώσπου να καταλήξουμε στην απλούστερη και πιο αποδοτική, κατά την άποψή μας, τεχνική που περιγράφηκε παραπάνω και αξιοποιεί τον πίνακα Slots.

Τέλος, όσον αφορά στην υλοποίηση της βάσης δεδομένων, το τελευταίο πρόβλημα που αντιμετωπίσαμε ήταν η έλλειψη ξεκάθαρου σχεδιασμού του αναμενόμενου σχήματος και των σχέσεων μεταξύ των πινάκων. Καθώς το project προχωρούσε στα στάδια υλοποίησής του, εμφανίστηκαν ανάγκες για αλλαγές στον τρόπο δόμησης της βάσης, οι οποίες απαιτούσαν να γίνει migration από μία version στην επόμενη, διαδικασία η οποία μας δυσκόλεψε μέχρι να καταλάβουμε πως ακριβώς υλοποιείται. Αυτές οι αλλαγές θα μπορούσαν να είχαν μειωθεί εάν είχαμε εξαρχής ξεκάθαρο το σχήμα και τον τρόπο χρήσης της βάσης, ωστόσο πιθανότατα να μην είχαν αποφευχθεί λόγω της μη προϋπάρχουσας εξοικείωσής μας με τη γλώσσα και το περιβάλλον Android.

Κεντρική Οθόνη

Κατά την ανάπτυξη της κεντρικής οθόνης, παρουσιάστηκαν ορισμένες προκλήσεις που αφορούσαν τη διαχείριση δεδομένων, τη συσχέτιση πινάκων και την εξασφάλιση μιας βέλτιστης εμπειρίας χρήστη. Ένα από τα βασικά ζητήματα ήταν η ανάγκη φιλτραρίσματος των διαθέσιμων εστιατορίων (stores) με βάση τη διαθεσιμότητα των slots. Αυτό απαίτησε τη συσχέτιση δεδομένων από δύο διαφορετικές πηγές: τους πίνακες “stores” και “slots”. Για κάθε κατάστημα, έπρεπε να ληφθούν τα slots και να ελεγχθεί εάν υπήρχε διαθέσιμο slot για την επιλεγμένη ημέρα και τον αριθμό ατόμων που όρισε ο χρήστης.

Παράλληλα, οι προσφορές (“offers”) έπρεπε να οδηγούν στο σωστό κατάστημα, το οποίο μάλιστα έπρεπε να είναι διαθέσιμο για την επιλεγμένη ημέρα και αριθμό ατόμων. Αυτό παρουσίαζε δυσκολίες, καθώς έπρεπε να διασφαλιστεί ότι κάθε προσφορά συσχετίζεται σωστά με το αντίστοιχο κατάστημα και ότι το συγκεκριμένο κατάστημα περιλαμβάνεται στη φιλτραρισμένη λίστα διαθέσιμων καταστημάτων. Σε περιπτώσεις όπου το κατάστημα δεν ήταν διαθέσιμο, ήταν απαραίτητο να εμφανιστούν κατάλληλα μηνύματα σφάλματος, ώστε ο χρήστης να έχει μια φιλική και κατανοητή εμπειρία.

Η δυναμική φόρτωση εικόνων για κάθε κατάστημα και προσφορά αποτέλεσε επίσης σημαντική πρόκληση. Ήταν αναγκαίο να υπάρχει σωστή αντιστοίχιση εικόνων με βάση τα δεδομένα, όπως το storeId και το offer.name, ενώ έπρεπε να δοθεί έμφαση και στον τρόπο εμφάνισης των δεδομένων και στη προσαρμογή τους, ειδικά στις λίστες όπως το LazyRow και το LazyColumn.

Ένα ακόμη ζήτημα ήταν η πολυπλοκότητα στη δυναμική ενημέρωση των δεδομένων. Οι αλλαγές φίλτρων, όπως η ημέρα ή ο αριθμός ατόμων, επηρέαζαν τη λίστα των καταστημάτων και απαιτούσαν συνεχή ανανέωση. Η λογική φιλτραρίσματος έπρεπε να παρακολουθεί τις αλλαγές στις καταστάσεις (states) και να ενημερώνει τις λίστες σε πραγματικό χρόνο. Αυτό αύξανε την πολυπλοκότητα του κώδικα και απαιτούσε προσεκτική διαχείριση, ώστε να αποφευχθούν σφάλματα.

Συνολικά, η διαδικασία ανάπτυξης της κεντρικής οθόνης ήταν απαιτητική, καθώς έπρεπε να συνδυαστούν η διαχείριση δεδομένων, η συσχέτιση πινάκων και η εξασφάλιση μιας βέλτιστης εμπειρίας χρήστη, χωρίς να θυσιαστεί η απόδοση της εφαρμογής.

Google Maps

Σημαντική ήταν και η πρόκληση στο Google Maps API καθώς η διεύθυνση των καταστημάτων μας αποθηκεύεται ολογράφως για παράδειγμα Λεωφόρος Αλεξάνδρας 119. Το API που χρησιμοποιήσαμε αντί για ολόκληρη τη διεύθυνση δέχεται σαν παραμέτρους γεωγραφικές συντεταγμένες (longitude,latitude). Το πρόβλημα αυτό αντιμετωπίστηκε με τη μετατροπή της διεύθυνσης από ολογράφως σε συντεταγμένες με τη χρήση του Geocoder(μέρος του android.location package).

Οδηγίες εγκατάστασης: Για το τρέξιμο της εφαρμογής μας απαιτείται η λήψη των αρχείων από το αποθετήριο katerinarousi/Spook_Android, ή το clone του ίδιου του αποθετηρίου μέσω git με την χρήση της εντολής:

git clone https://github.com/katerinarousi/Spook_Android.git.

Μετά τη λήψη το μόνο που χρειάζεται για το τρέξιμο της εφαρμογής μας είναι το άνοιγμα των αρχείο στο περιβάλλον του Android Studio και η διαδικασία build.