ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σύστημα Κρατήσεων για Καταστήματα Εστίασης»

Των φοιτητών

Τσιφλίδη Γεώργιου
Αρ. Μητρώου: 164812

Τράκα Αλέξανδρου
Αρ. Μητρώου: 164800

Επιβλέπουσα

Ελβίρα- Μαρία Αρβανίτου

Ημερομηνία 10/9/2024

Τίτλος Δ.Ε. Σύστημα Κρατήσεων για Καταστήματα Εστίασης
Κωδικός Δ.Ε. 23165
Ονοματεπώνυμο φοιτητών Τσιφλίδης Γεώργιος – Τράκας Αλέξανδρος
Ονοματεπώνυμο εισηγητή Ελβίρα – Μαρία Αρβανίτου
Ημερομηνία ανάληψης Δ.Ε. 23-03-2023
Ημερομηνία περάτωσης Δ.Ε. 10-09-2024

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.

«Στον αγαπημένο συνάδελφο και φίλο Αλέξανδρο για το διάστημα που τον ανέχτηκα και με ανέχτηκε.
Στις οικογένειές μας που κάνουν τόσα χρόνια υπομονή! »

Πρόλογος

Η επιλογή του θέματος "Σύστημα Κρατήσεων για Καταστήματα Εστίασης" προέκυψε από την επιθυμία μας να συνδυάσουμε τις γνώσεις μας στην τεχνολογία και την πληροφορική με την ανάγκη της αγοράς για αποτελεσματικότερες λύσεις στη διαχείριση κρατήσεων. Στην εποχή της ψηφιακής μεταμόρφωσης, οι επιχειρήσεις εστίασης αντιμετωπίζουν την πρόκληση της οργάνωσης και της βελτιστοποίησης των υπηρεσιών τους. Ένα σύστημα κρατήσεων μπορεί να προσφέρει σημαντικά πλεονεκτήματα, όπως καλύτερη διαχείριση των τραπεζιών, αύξηση της ικανοποίησης των πελατών και μείωση των χαμένων κρατήσεων.

Η πτυχιακή αυτή εργασία, μας προσέφερε τη δυνατότητα να εμβαθύνουμε σε θέματα όπως ο σχεδιασμός φιλικών προς τον χρήστη εφαρμογών, η διαχείριση δεδομένων σε πραγματικό χρόνο και οι τεχνικές βελτιστοποίησης. Μέσω της ανάπτυξης του συστήματος αυτού, καταφέραμε να εφαρμόσουμε τις θεωρητικές γνώσεις μας στην πράξη, ενισχύοντας τις δεξιότητές μας στον προγραμματισμό και τη διαχείριση έργων. Επιπλέον, η εργασία αυτή μας βοήθησε να αντιληφθούμε καλύτερα τις ανάγκες και τις προκλήσεις που αντιμετωπίζουν οι επαγγελματίες του χώρου της εστίασης, προσφέροντάς μας μια πιο ολοκληρωμένη εικόνα της βιομηχανίας.

Περίληψη

Στην παρούσα πτυχιακή εργασία παρουσιάζεται η υλοποίηση μιας cross-platform εφαρμογής σε android και web αντίστοιχα, με θέμα ένα ολοκληρωμένο σύστημα κρατήσεων για καταστήματα εστίασης. Είναι ένα εγχείρημα το οποίο θα μπορούσε να ενσωματωθεί στο λειτουργικό σύστημα του καταστήματος, διευκολύνοντας τη διαδικασία κρατήσεων τόσο για τους πελάτες όσο και για τα καταστήματα εστίασης.

Αρχικό βήμα ήταν η κατανόηση του προβλήματος από όλες τις εμπλεκόμενες πλευρές, δηλαδή τους πελάτες και τους διαχειριστές του συστήματος, όπως προέκυψε από την ανάλυση των χαρακτηριστικών χρηστών. Αναλύθηκε ο ανταγωνισμός και παρόμοιες εφαρμογές, μέσω των οποίων καθορίστηκαν οι απαιτήσεις που το σύστημα έπρεπε να καλύψει.

Αφού ολοκληρώθηκε η φάση του σχεδιασμού, ξεκίνησε η υλοποίηση της εφαρμογής με τις κατάλληλες τεχνολογίες για την επίτευξη των στόχων. Η αρχιτεκτονική του συστήματος επεξηγείται λεπτομερώς. Συγκεκριμένα, χρησιμοποιήθηκε το Flutter για το frontend και το Firebase για το backend. Στην πορεία, γίνεται και μια περιήγηση στην εφαρμογή μέσω παρουσίασης των επιμέρους σελίδων της.

Τέλος, η αξιολόγηση πραγματοποιήθηκε σε δύο στάδια. Πρώτα έγινε σύγκριση των λειτουργικών απαιτήσεων που αναφέρθηκαν προηγουμένως, για να διαπιστωθεί ο βαθμός κάλυψής τους κατά τη διαδικασία της υλοποίησης. Στο δεύτερο στάδιο, πραγματοποιήθηκε αξιολόγηση με τους χρήστες που συμμετείχαν στον σχεδιασμό. Αυτή η διαδικασία έδειξε ότι οι στόχοι επιτεύχθηκαν, αποδεικνύοντας πως το σύστημα μπορεί να λειτουργήσει αποτελεσματικά σε πραγματικές συνθήκες.

«Reservation System for Restaurants»

«Georgios Tsiflidis & Alexandros Trakas»

Abstract

This thesis presents the implementation of a cross-platform in android and correspondingly web, with an integrated reservation system for restaurants. It is a venture that could be integrated into the store's operating system, it facilitates the reservation process for both customers and restaurants.

The initial step was the understanding of the problem by all parties involved, i.e. customers and system administrators, as derived from the analysis of user characteristics. Analyzed the competition and similar applications, through which the system had to cover.

After the design phase was completed, its implementation began with the appropriate technologies to select the targets. The system architecture is explained in detail. Specifically, Flutter was used for the frontend and Firebase for the backend. Along the way, a description of the application is made through a presentation of its individual pages.

Finally, the evaluation was carried out in two stages. First, a comparison was made of the previously mentioned functional requirements, in order to determine the extent to which they were covered during the implementation process. In the second stage, it was evaluated with the users involved in the design. This process showed that the objectives were achieved, proving that the system can work effectively in real-world conditions.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον φίλο μου Αλέξανδρο για την πολύτιμη συνεργασία και υποστήριξή του καθ' όλη τη διάρκεια της υλοποίησης αυτής της εργασίας. Επίσης, ευχαριστώ θερμά τις οικογένειές μας για την αδιάκοπη υποστήριξη και κατανόηση που μας παρείχαν.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Π.Ε.	Πτυχιακή Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
ERD	Entity Relationship Diagram
Web	Website
iOS	iPhone Operating System
A.P.I.	Application Programming Interface

Κεφάλαιο 1ο: Εισαγωγή

Η ανάπτυξη εφαρμογών που μπορούν να λειτουργούν απρόσκοπτα τόσο σε πλατφόρμες Ιστού όσο και σε κινητές συσκευές έχει καταστεί κρίσιμη στην εποχή της ψηφιακής καινοτομίας. Η υλοποίηση μιας cross-platform εφαρμογής επιτρέπει την εξοικονόμηση πόρων και χρόνου, καθώς ο κώδικας μπορεί να γραφτεί μία φορά και να εκτελεστεί σε πολλαπλές πλατφόρμες, συμπεριλαμβανομένων των iOS, Android, και web browsers. Αυτή η προσέγγιση όχι μόνο διασφαλίζει μια ενιαία και συνεπή εμπειρία χρήστη, αλλά και διευκολύνει τη συντήρηση και τις αναβαθμίσεις της εφαρμογής. Στην εποχή της ψηφιακής επανάστασης και της γρήγορης ψηφιοποίησης της καθημερινότητάς μας, ο τρόπος που κάνουμε κρατήσεις σε εστιατόρια εξελίσσεται συνεχώς. Η εφαρμογή της τεχνολογίας σε αυτόν τον τομέα έχει δημιουργήσει νέες δυνατότητες για τους καταναλωτές και τους επαγγελματίες της εστίασης. Το RESERVE-EAT, το όνομα της εφαρμογής μας, αποτελεί ένα παράδειγμα καινοτόμου συστήματος κρατήσεων που ανταποκρίνεται στις σύγχρονες ανάγκες και προσδοκίες των χρηστών του. Η συγκεκριμένη εφαρμογή, που αφορά το σύστημα κρατήσεων για καταστήματα εστίασης, ονομάζεται RESERVE-EAT και στοχεύει να συνδυάσει την ευελιξία και τη λειτουργικότητα, προσφέροντας στους χρήστες της μια πλούσια και διαισθητική εμπειρία ανεξαρτήτως της συσκευής που χρησιμοποιούν (android/web). Με τη χρήση της σύγχρονης τεχνολογίας Flutter, η εφαρμογή εκμεταλλεύεται τα πλεονεκτήματα του native development, εξασφαλίζοντας παράλληλα τη διαλειτουργικότητα και την απόδοση. Το σύστημα κρατήσεων RESERVE-EAT όχι μόνο αποδεικνύει τις δυνατότητες της τεχνολογίας cross-platform, αλλά και επιδεικνύει τη σημασία της καινοτομίας και της προσαρμοστικότητας στον σύγχρονο κόσμο των εφαρμογών, συμβάλλοντας στην αναβάθμιση των υπηρεσιών των καταστημάτων εστίασης καθώς και τη διευκόλυνση των πελατών τους.

1.1 Σκοπός της Εργασίας - Συστήματα Κρατήσεων για Καταστήματα Εστίασης

Η επιλογή ενός καταστήματος εστίασης για δείπνο ή για μια ξεχωριστή εμπειρία διαδραματίζει σημαντικό ρόλο στην αίσθηση ικανοποίησης του πελάτη. Ωστόσο, η διαδικασία κράτησης τραπεζιού μπορεί να είναι μια πρόκληση τόσο για το κατάστημα όσο και για τον πελάτη. Το κύριο αντικείμενο της εργασίας είναι να εξετάσει τη σημασία των συστημάτων κρατήσεων και να αναλύσει πώς εφαρμόζονται στον τομέα της εστίασης, εστιάζοντας στην εφαρμογή RESERVE-EAT ως παράδειγμα καινοτόμου λύσης.

Μέσω της ανάλυσης των δυνατοτήτων και των προκλήσεων των συστημάτων κρατήσεων, η εργασία θα διερευνήσει πώς αυτά τα συστήματα μπορούν να βελτιώσουν την εμπειρία των πελατών, να βοηθήσουν στην αποτελεσματική διαχείριση του χώρου και να ενισχύσουν την οργάνωση των επιχειρήσεων εστίασης. Το RESERVE-EAT αναδεικνύεται ως ένα εργαλείο που συμβάλλει στην αποτελεσματική και αποδοτική διαχείριση των κρατήσεων, προσφέροντας ταυτόχρονα εξατομικευμένες επιλογές για τους χρήστες του.

1.2 Τρόποι κράτησης ενός τραπεζιού

Τα τελευταία χρόνια, είναι δεδομένο ότι οι πελάτες επιθυμείτε να βρείτε μια εύχρηστη εφαρμογή για κράτηση τραπεζιών ή οποιεσδήποτε άλλες υπηρεσίες για να αποφύγετε το φυσικό περπάτημα, το ξενοδοχείο ή την επικοινωνία μέσω κλήσης ή κράτηση μέσω μιας γνωριμίας[1]. Ως εκ τούτου, έχει ως στόχο την ανάπτυξη μιας εφαρμογής για κράτηση τραπεζιού και online κράτηση μενού. Η ανάπτυξη της τεχνολογίας και η καθημερινή χρήση των κινητών και υπολογιστών και ασύρματη τεχνολογία, έχει μεγάλο αντίκτυπο στις ζωές μας στην παρούσα κατάσταση σε σύγκριση με τα προηγούμενα έτη. Στις μέρες μας οι άνθρωποι αναζητούν μια εφαρμογή που να ικανοποιεί τις ανάγκες τους με ακόμη μεγαλύτερη ακρίβεια πάνω στις κρατήσεις τραπεζιών σε μία έξοδό τους. Ακόμα και το οι βιομηχανίες εστιατορίων αναζητούν οποιαδήποτε εφαρμογή για κινητά και υπολογιστές που ενισχύει την ευκολία και ευχρηστία των ίδιων και των πελατών τους ταυτόχρονα, καθώς και αυξάνει το κέρδος που με τη σειρά του δεν είναι μόνο πλεονέκτημα για το εκάστοτε εστιατόριο, αλλά και για να επιλέξουν οι πελάτες το εστιατόριο της επιθυμητής επιλογής και τοποθεσίας, καθώς και το να βρουν διαθέσιμο τραπέζι στο μαγαζί της επιθυμίας τους. Με μία τέτοια εφαρμογή θα υπάρχει εξοικονόμηση χρόνου, μετρητών και χαρτιών, μετατρέποντας το εγχειρίδιο σύστημα κρατήσεων σε αυτοματοποιημένο σύστημα.

1.3 Ώθηση υλοποίησης της πτυχιακής

Όπως βλέπουμε και παραπάνω είναι σημαντικό να αναγνωριστεί ο ρόλος και η σημασία εφαρμογών κρατήσεων τραπεζιών σε καταστήματα της εστίασης. Αυτό το έργο στοχεύει στην απλοποίηση της διαδικασίας κράτησης, καθιστώντας την πιο προσιτή και μειώνοντας την ανάγκη για άμεση αλληλεπίδραση με το προσωπικό του εστιατορίου.

Η δημιουργία της εφαρμογής RESERVE-EAT όχι μόνο ανταποκρίνεται σε μια ανάγκη της αγοράς, αλλά ευθυγραμμίζεται και με τις προσωπικές μας και επαγγελματικές μας φιλοδοξίες. Ως προγραμματιστές λογισμικού, αυτό το έργο προσφέρει την ευκαιρία να βελτιώσουμε τις δεξιότητές μας και να συμβάλουμε σε μια καινοτόμο λύση που μπορεί να βελτιώσει ίσως και ολόκληρη την αγορά. Η επιτυχής ανάπτυξη και κυκλοφορία αυτής της εφαρμογής θα ήταν ένα σημαντικό επίτευγμα, δυνητικά καθιερώνοντας μια ισχυρή παρουσία στην αγορά με συνεχείς ενημερώσεις και βελτιώσεις.

Εν κατακλείδι, η εφαρμογή RESERVE-EAT αποτελεί παράδειγμα της συγχώνευσης προσωπικών κινήτρων και επαγγελματικής εξειδίκευσης για τη δημιουργία μιας λύσης που ανταποκρίνεται σε μια πραγματική ανάγκη. Με τον εξορθολογισμό της διαδικασίας κράτησης και τη βελτίωση της συνολικής εμπειρίας, αυτή η εφαρμογή έχει τη δυνατότητα να έχει ουσιαστικό αντίκτυπο στον κλάδο της εστίασης. Η επιτυχημένη εφαρμογή και προώθηση της εφαρμογής αυτής,θα καταδείξει την πρακτική εφαρμογή των δεξιοτήτων μας και θα συμβάλει στην ανάπτυξή μας ως προγραμματιστές λογισμικού.

1.4 Δομή πτυχιακής εργασίας

Στην παρούσα ενότητα, γίνεται μία σύντομη ανασκόπηση της παρούσας πτυχιακής εργασίας.

Στο πρώτο κεφάλαιο, γίνεται μία εισαγωγή́ στο θέμα που πραγματεύεται η παρούσα πτυχιακή εργασία, και παρουσιάζεται ο σκοπός για τον οποίο αναπτύσσεται. Ακόμα, παρουσιάζεται και η ώθηση για την σχεδίαση και την υλοποίηση του συστήματος, στο οποίο αναφέρεται.

Στο κεφάλαιο 2 παρουσιάζεται η ανάλυση σε εφαρμογές ίδιου είδους, και μέσω της ανάλυσης του ανταγωνισμού προκύπτουν οι στόχοι και οι απαιτήσεις της εφαρμογής που υλοποιείται. Ακόμα γίνεται

λόγος σχετικά με την ευχρηστία της εφαρμογής και οι απαιτήσεις που προκύπτουν για την ανάπτυξη αυτής της εφαρμογής.

Το κεφάλαιο 3 παρουσιάζονται τα εργαλεία σχεδίασης που χρησιμοποιήθηκαν και η διαδικασία με την οποία έγιναν κατανοητοί οι στόχοι του κάθε χρήστη και σχεδιάστηκε το σύστημα. Πιο συγκεκριμένα, γίνεται εκτενής αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν από το σύνολο της εφαρμογής, καθώς και σε εργαλεία που βοήθησαν κατά την ανάπτυξη της.

Στο κεφάλαιο 4 αναλύεται η ανάπτυξη της εφαρμογής σε συνολικό επίπεδο, με παραδείγματα κώδικα για την καλύτερη κατανόηση. Αποτελεί τεχνική αναφορά του έργου που υλοποιήθηκε, από το backend κομμάτι και τη βάση δεδομένων, μέχρι το front-end,δηλαδή την υλοποίηση της εφαρμογής. Στο τέλος του παρατίθενται όλες οι οθόνες της εφαρμογής σε μορφή screenshot-οθονών και γίνεται ανάλυση του τελικού προϊόντος και η συνολική χρήσης της εφαρμογής με αναφορές σε κομμάτια του πηγαίου κώδικα.

Τέλος, το κεφάλαιο 5 αναφέρεται σε μελλοντικές υλοποιήσεις και βελτιώσεις της εφαρμογής, έτσι ώστε να καταστεί ένα ολοκληρωμένο σύστημα κρατήσεων τραπεζιών στην αγορά.

Κεφάλαιο 2ο: Σχετικά με την εφαρμογή

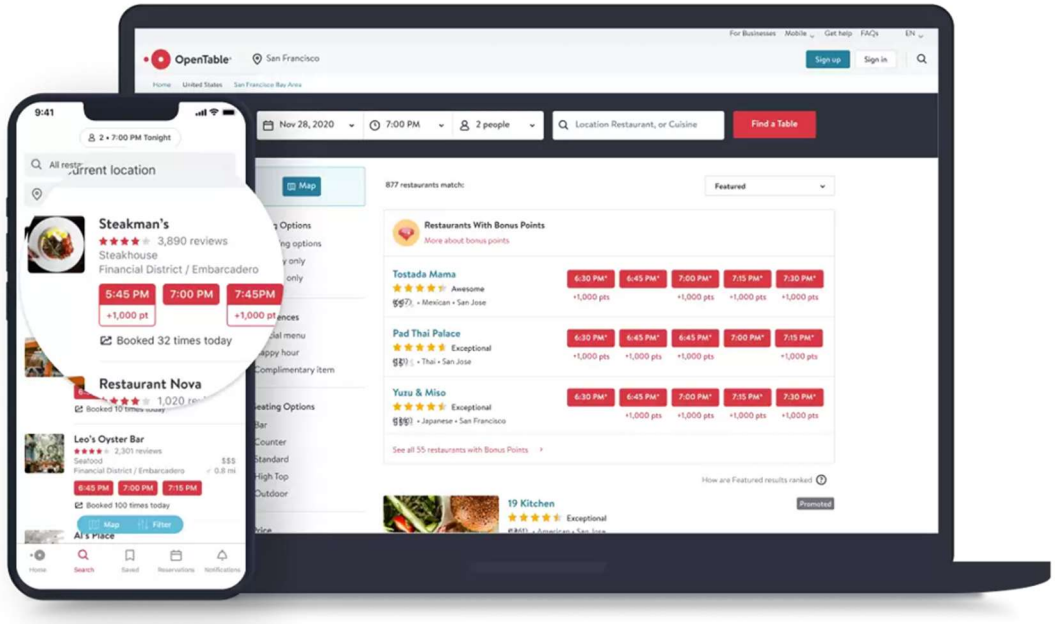
2.1 Ανάλυση ανταγωνισμού / Παρόμοιες εφαρμογές

Για να αναπτυχθεί ένα σύστημα κρατήσεων ενός εστιατορίου για καταστήματα εστίασης, είναι σημαντικό να αναλυθούν οι υπάρχουσες εφαρμογές και οι ανταγωνιστικές λύσεις που είναι διαθέσιμες στην αγορά. Αυτή η ανάλυση βοηθά στον εντοπισμό των δυνατοτήτων και των αδυναμιών των ανταγωνιστών, καθώς και στην αναγνώριση των βέλτιστων πρακτικών που μπορούν να ενσωματωθούν στη νέα εφαρμογή.

2.1.1 OpenTable

Το OpenTable είναι μια από τις κορυφαίες πλατφόρμες κρατήσεων εστιατορίων στον κόσμο, παρέχοντας μια ολοκληρωμένη λύση για τη διαχείριση κρατήσεων τόσο για τους πελάτες όσο και για τα εστιατόρια. Ιδρύθηκε το 1998 και από τότε έχει επεκταθεί σημαντικά, εξυπηρετώντας εκατομμύρια χρήστες και χιλιάδες εστιατόρια παγκοσμίως. Ξεκίνησε με στόχο να διευκολύνει τις κρατήσεις εστιατορίων μέσω διαδικτύου. Αρχικά προσέφερε τη δυνατότητα στους πελάτες να κάνουν κρατήσεις μέσω της ιστοσελίδας της, αλλά γρήγορα επεκτάθηκε σε εφαρμογές για κινητά και tablet. Σήμερα, το OpenTable αποτελεί μέρος της Booking Holdings, μια από τις μεγαλύτερες εταιρείες ταξιδιωτικών υπηρεσιών στον κόσμο. Αυτό μας δείχνει τη μεγάλη επιτυχία και απήχηση της εφαρμογής καθ' όλη τη διάρκεια της πορείας της μέχρι σήμερα. Οι χρήστες αναζητούν εστιατόρια και κρατήσεις με βάση παραμέτρους όπως ημερομηνίες, ώρες, κουζίνα και εύρος τιμών. Οι χρήστες που έχουν καταχωρήσει τη διεύθυνση email τους στο σύστημα θα λάβουν στη συνέχεια ένα email επιβεβαίωσης.[12] Τα εστιατόρια που χρησιμοποιούν, έχουν πρόσβαση σε μια σειρά εργαλείων για τη διαχείριση των κρατήσεων. Αυτά περιλαμβάνουν τη δυνατότητα να βλέπουν και να διαχειρίζονται τις κρατήσεις σε πραγματικό χρόνο, να διατηρούν αρχεία πελατών και να στέλνουν αυτόματες υπενθυμίσεις και επιβεβαιώσεις κρατήσεων. Οι χρήστες ακόμα μπορούν να γράψουν και να διαβάσουν κριτικές για τα εστιατόρια. Αυτές οι κριτικές βοηθούν άλλους χρήστες να επιλέξουν το κατάλληλο εστιατόριο και παρέχουν ανατροφοδότηση στους ιδιοκτήτες των εστιατορίων. Επιπρόσθετα, χρησιμοποιεί αλγορίθμους για να προτείνει εστιατόρια στους χρήστες με βάση τις προτιμήσεις και το ιστορικό κρατήσεών τους. Αυτό βοηθά τους χρήστες να ανακαλύψουν νέα μέρη που μπορεί να τους ενδιαφέρουν. Είναι ενδιαφέρον να σημειωθεί ότι έχει συνεργαστεί με άλλες πλατφόρμες και υπηρεσίες, όπως το Google Maps και το Yelp, επιτρέποντας στους χρήστες να κάνουν κρατήσεις απευθείας από αυτές τις εφαρμογές. Επιπλέον, ενσωματώνεται με διάφορα συστήματα διαχείρισης εστιατορίων, καθιστώντας τη διαχείριση των κρατήσεων ακόμα πιο ομαλή. Εν ολίγοις είναι σχεδιασμένο για να είναι εύκολο στη χρήση τόσο από τους πελάτες όσο και από τα εστιατόρια. Η διεπαφή είναι φιλική και οι διαδικασίες κρατήσεων και διαχείρισης είναι απλές και γρήγορες. Με εκατομμύρια χρήστες παγκοσμίως, η εφαρμογή αυτή προσφέρει στα εστιατόρια πρόσβαση σε ένα τεράστιο κοινό, αυξάνοντας την προβολή και τις πιθανότητες κρατήσεων. Τα εστιατόρια μπορούν να εκμεταλλευτούν τα αναλυτικά εργαλεία του περιβάλλοντος της εφαρμογής, για να κατανοήσουν τις τάσεις των κρατήσεων, τις προτιμήσεις των πελατών και να βελτιστοποιήσουν τη λειτουργία τους. Οι πελάτες λαμβάνουν προσαρμοσμένες προτάσεις και υπενθυμίσεις, καθιστώντας την εμπειρία χρήσης πιο προσωπική και ευχάριστη. Παρά τα πλεονεκτήματα, το OpenTable μπορεί να είναι δαπανηρό για τα εστιατόρια, ειδικά για τα μικρότερα. Τα κόστη περιλαμβάνουν τέλη εγγραφής και προμήθειες για κάθε κράτηση. Τα εστιατόρια που βασίζονται αποκλειστικά στην εφαρμογή για τις κρατήσεις τους μπορεί να βρεθούν σε δύσκολη θέση αν υπάρξει κάποιο πρόβλημα με την πλατφόρμα ή αν οι χρήστες στραφούν σε άλλες λύσεις. Αν και το

OpenTable ενσωματώνεται με μεγάλα συστήματα διαχείρισης, μπορεί να υπάρχουν περιορισμοί στην ενσωμάτωση με μικρότερα ή τοπικά συστήματα διαχείρισης εστιατορίων. Το OpenTable είναι μια ισχυρή και ευρέως αναγνωρισμένη πλατφόρμα κρατήσεων που προσφέρει πολλές δυνατότητες τόσο στους πελάτες όσο και στα εστιατόρια. Ενώ παρέχει εξαιρετικά εργαλεία διαχείρισης και ευκολίες για τους χρήστες, το κόστος και η εξάρτηση από την πλατφόρμα μπορεί να αποτελέσουν προκλήσεις για ορισμένα εστιατόρια. Παρ' όλα αυτά, η δυνατότητα να προσεγγίσει ένα ευρύ κοινό και να βελτιώσει την εμπειρία κρατήσεων καθιστά το OpenTable μια πολύτιμη λύση για πολλές επιχειρήσεις εστίασης.

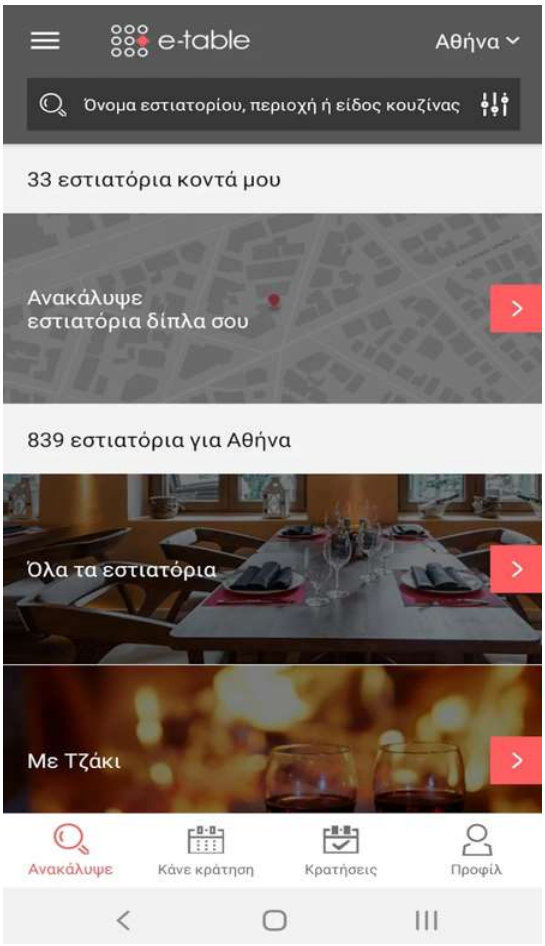


Εικόνα 2.1.1 OpenTable mobile/web αναζήτηση-αρχική

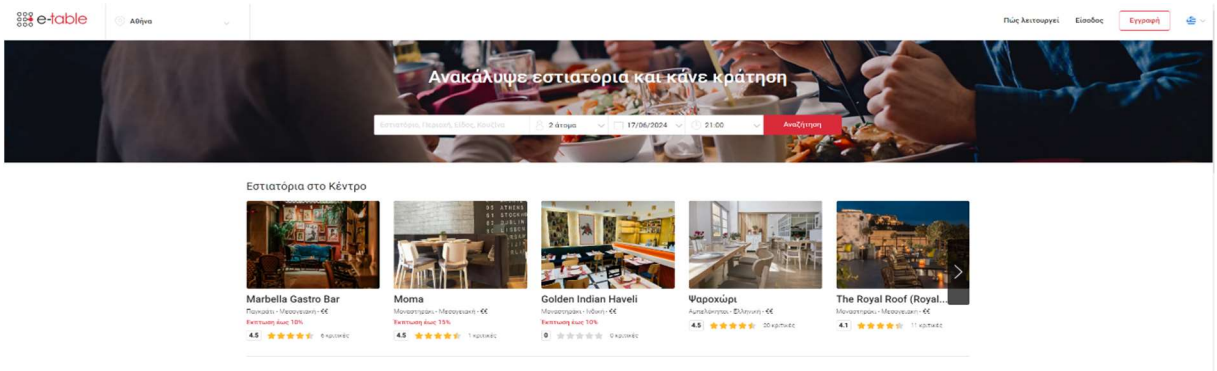
2.1.2 E-table

Το E-table είναι μια διαδικτυακή πλατφόρμα κρατήσεων που εξυπηρετεί κατά κύριο λόγο την Ελλάδα και την Κύπρο. Διευκολύνει απρόσκοπτες κρατήσεις σε περισσότερα από 1.500 εστιατόρια, παρέχοντας στους χρήστες έναν βολικό τρόπο να ανακαλύψουν, να κάνουν κράτηση και να απολαύσουν γευστικές εμπειρίες με αποκλειστικά προνόμια. Οι χρήστες μπορούν να κάνουν κράτηση τραπεζιών αμέσως χωρίς να χρειάζονται τηλεφωνικές κλήσεις ή επιβεβαιώσεις μέσω email. Αυτό εξασφαλίζει μια διαδικασία κράτησης χωρίς προβλήματα. Η πλατφόρμα δεν απαιτεί στοιχεία πιστωτικής κάρτας για κρατήσεις, καθιστώντας την προσβάσιμη και φιλική προς τον χρήστη. Οι χρήστες μπορούν να έχουν πρόσβαση σε μενού, φωτογραφίες και λεπτομερείς περιγραφές εστιατορίων για να κάνουν ενημερωμένες επιλογές. Η πλατφόρμα διαθέτει κριτικές από πραγματικούς πελάτες, οι οποίες βοηθούν τους χρήστες να αποφασίσουν πού θα δειπνήσουν με βάση τα αυθεντικά σχόλια. Επιπρόσθετα, προσφέρει εκπτώσεις έως και 30% στα συμμετέχοντα εστιατόρια, που εφαρμόζονται απευθείας στον τελικό λογαριασμό χωρίς την ανάγκη κουπονιών. Οι χρήστες κερδίζουν πόντους για κάθε κράτηση και κριτική, οι οποίοι μπορούν να εξαργυρωθούν για πρόσθετα προνόμια, προσθέτοντας ένα στοιχείο παιχνιδιού στην γευστική εμπειρία. Η εφαρμογή παρέχει πολλά φίλτρα αναζήτησης και κριτήρια, επιτρέποντας στους χρήστες να βρίσκουν γρήγορα εστιατόρια που ανταποκρίνονται στις προτιμήσεις τους όσον αφορά την κουζίνα, την τοποθεσία και το στυλ. Με παρουσία σε μεγάλες ελληνικές πόλεις και διάφορες περιοχές, το e-table προσφέρει μια ποικιλία επιλογών για φαγητό, από χαλαρά εστιατόρια μέχρι εκλεκτά εστιατόρια. Διαθέσιμη τόσο για Android και για iOS όσο και για χρήστες του web, η εφαρμογή για

κινητά επιτρέπει στους χρήστες να κάνουν κρατήσεις εν κινήσει, βελτιώνοντας την ευκολία και την ευελιξία ή ακόμα και από το σπίτι ή το γραφείο τους από έναν υπολογιστή. Έχει φτάσει να ανταγωνίζεται πολλές άλλες διαδικτυακές πλατφόρμες κρατήσεων όπως το OpenTable που αναφέρθηκε παραπάνω. Ωστόσο, διακρίνεται με συγκεκριμένα χαρακτηριστικά προσαρμοσμένα στην ελληνική αγορά, όπως τοπικά φίλτρα αναζήτησης, τοπικές συνεργασίες εστιατορίων και αποκλειστικές εκπτώσεις για χρήστες εντός Ελλάδας και Κύπρου. Για τους ιδιοκτήτες εστιατορίων, παρέχει εργαλεία για την αποτελεσματική διαχείριση των κρατήσεων, τη μείωση των μη εμφανίσεων και την προσέλκυση νέων πελατών μέσω των προωθητικών δυνατοτήτων της πλατφόρμας. Προσφέρει επίσης πληροφορίες και αναλυτικά στοιχεία για να βοηθήσει τα εστιατόρια να κατανοήσουν καλύτερα τη βάση των πελατών τους και να βελτιστοποιήσουν ανάλογα τις δραστηριότητές τους. Η εμπειρία χρήστη έχει σχεδιαστεί για να είναι απλή και ικανοποιητική. Από την ανακάλυψη νέων σημείων για φαγητό μέχρι την απόκτηση πόντων επιβράβευσης και τη λήψη άμεσων επιβεβαιώσεων κράτησης, στοχεύει να κάνει το φαγητό σε μια ευχάριστη και απρόσκοπτη εμπειρία. Ενώ το e-table προσφέρει μια βολική πλατφόρμα για κρατήσεις εστιατορίων, έχει αρκετά μειονεκτήματα. Η περιορισμένη γεωγραφική του κάλυψη περιορίζει τη χρηστικότητα εκτός Ελλάδας και Κύπρου και η εξάρτηση από την πρόσβαση στο Διαδίκτυο μπορεί να είναι προβληματική σε περιοχές με κακή συνδεσιμότητα. Η διεπαφή χρήστη μπορεί να δημιουργήσει προκλήσεις πλοήγησης, ιδιαίτερα για άτομα με λιγότερο γνώσεις τεχνολογίας, και οι καταχωρίσεις εστιατορίων της πλατφόρμας ενδέχεται να μην περιλαμβάνουν όλες τις επιλογές για φαγητό, ειδικά νεότερες ή μικρότερες εγκαταστάσεις. Οι χρόνοι απόκρισης υποστήριξης πελατών μπορεί να είναι αργοί, επηρεάζοντας αρνητικά την εμπειρία του χρήστη. Οι ερωτήσεις σχετικά με την αυθεντικότητα των κριτικών εξακολουθούν να υφίστανται και τεχνικές δυσλειτουργίες, όπως σφάλματα εφαρμογής ή σφάλματα κράτησης, μπορεί να απογοητεύσουν τους χρήστες. Επιπλέον, η έλλειψη ολοκληρωμένων λύσεων πληρωμής και η πιθανή σύγχυση σχετικά με την εφαρμογή εκπτώσεων μειώνουν περαιτέρω την ευκολία. Παρά τους περιορισμούς του, το e-table παραμένει σημαντικός παίκτης στον κλάδο των online κρατήσεων εστιατορίων, ιδιαίτερα στην Ελλάδα και την Κύπρο. Η ευκολία χρήσης της πλατφόρμας, οι εκτενείς καταχωρίσεις εστιατορίων και η δυνατότητα να προσφέρει κρατήσεις σε πραγματικό χρόνο την έχουν κάνει δημοφιλή επιλογή μεταξύ των γευμάτων.



Εικόνα 2.1.2 E-table αναζήτηση-αρχική οθόνη mobile.



Εικόνα 2.1.3 E-table αναζήτηση-αρχική οθόνη web.

Σύστημα	Ιστορικό πελάτη	Προβολή κριτικών	Επιλογή τραπεζιού	Αναμονή	Εξατομίκευση για εστιατόριο
OpenTable	✓	✓	×	×	×
E-table	✓	✓	×	×	×

Εικόνα 2.1.4 Σύγκριση παρόμοιων συστημάτων.

2.2.3 Συμπεράσματα από μελέτη παρόμοιων συστημάτων

Τα παραπάνω συστήματα χρησιμοποιούνται σαν διαμεσολαβητές ανάμεσα στον πελάτη και το εστιατόριο. Ο πελάτης πραγματοποιεί την κράτηση μέσω ανεξάρτητου συστήματος, το οποίο κατόπιν ενημερώνει το εστιατόριο για την κράτηση. Ενδεχόμενα προβλήματα που ίσως προκύψουν στη διαδικασία κράτησης απαιτούν την επέμβαση του τμήματος εξυπηρέτησης πελατών του κάθε συστήματος, με αποτέλεσμα την καθυστέρηση της εξόρμησης σε ένα εστιατόριο. Η παρούσα εφαρμογή επιτρέπει την άμεση κράτηση στο σύστημα του εστιατορίου, με τον διαχειριστή και τον πελάτη να παρακολουθούν σε πραγματικό χρόνο την νέα κράτηση. Η λειτουργία περιλαμβάνει επιλογή συγκεκριμένου τραπεζιού, προσφέροντας λειτουργικότητα που πολλά συστήματα φαίνεται να μην υποστηρίζουν. Επιπλέον, ο διαχειριστής, διαθέτοντας ένα ενσωματωμένο σύστημα στην εφαρμογή του, δεν απαιτείται να πληρώσει συνδρομή σε συνεργαζόμενη υπηρεσία, καθιστώντας τη διαχείριση των κρατήσεων αυτόνομη για το εστιατόριο.

2.2 Απαιτήσεις του συστήματος

Αυτή η εργασία στοχεύει να ικανοποιήσει τα κριτήρια μιας αποτελεσματικής εφαρμογής κρατήσεων για καταστήματα εστίασης. Μια τέτοια εφαρμογή πρέπει να διευκολύνει τόσο τους πελάτες όσο και τους διαχειριστές των εστιατορίων, προσφέροντας μια ολοκληρωμένη και εύχρηστη εμπειρία.

Οι κρατήσεις σε εστιατόρια αποτελούν σημαντικό στοιχείο της λειτουργίας τους, επηρεάζοντας την εμπειρία των πελατών και την αποδοτικότητα της διαχείρισης του χώρου. Σε δημοφιλή εστιατόρια, σε μεγάλες πόλεις και κατά τις ώρες αιχμής, όπως τα Σαββατοκύριακα και οι αργίες, η ζήτηση για κρατήσεις συχνά υπερβαίνει τη διαθεσιμότητα των τραπεζιών. Αυτό μπορεί να προκαλέσει δυσκολίες στη διαχείριση των κρατήσεων και απογοήτευση στους πελάτες, καθώς είναι πιθανό να μην βρουν κάτι της αρεσκείας τους. Ένα δημοφιλές εστιατόριο στο κέντρο της πόλης μπορεί να είναι πλήρες για κρατήσεις για το δείπνο του Σαββάτου από νωρίς μέσα στην εβδομάδα, αφήνοντας πολλούς πελάτες χωρίς τραπέζι.

Οι πελάτες μπορεί να αλλάξουν ή να ακυρώσουν τις κρατήσεις τους για διάφορους λόγους. Η έγκαιρη ενημέρωση και διαχείριση αυτών των αλλαγών είναι απαραίτητη για να μπορούν τα εστιατόρια να προσαρμόζονται και να βελτιστοποιούν τη διαθεσιμότητά τους. Ένας πελάτης ακυρώνει την κράτησή του την τελευταία στιγμή λόγω απρόβλεπτων περιστατικών, και το εστιατόριο χρειάζεται να βρει νέο πελάτη για το κενό τραπέζι.

Οι πελάτες συχνά έχουν ειδικές απαιτήσεις, όπως συγκεκριμένα τραπέζια, γευστικές προτιμήσεις ή εορταστικές εκδηλώσεις. Η ικανοποίηση αυτών των αιτημάτων είναι σημαντική για την εμπειρία τους και απαιτεί προσεκτική διαχείριση από το προσωπικό του εστιατορίου. Ένα ζευγάρι ζητάει να καθίσει σε ένα τραπέζι με θέα και να τους προσφερθεί ένα ειδικό γεύμα για την επέτειό τους.

Ένα σύστημα κρατήσεων επιτρέπει στους πελάτες να βλέπουν τη διαθεσιμότητα σε πραγματικό χρόνο και να κάνουν τις κρατήσεις τους online χωρίς να χρειάζεται να επικοινωνήσουν τηλεφωνικά με το εστιατόριο. Αυτό μειώνει τον φόρτο εργασίας για το προσωπικό και βελτιώνει την εμπειρία του πελάτη.

Ακόμα, μπορούν να κλείνουν τραπέζι μέσω της ιστοσελίδας ή της εφαρμογής οποιαδήποτε στιγμή της ημέρας.

Προδιαγραφές Απαιτήσεων για Σύστημα Κρατήσεων Καταστημάτων Εστίασης

1. Δημιουργία Λογαριασμού Χρήστη: Οι χρήστες μπορούν να δημιουργήσουν έναν λογαριασμό εισάγοντας τα προσωπικά τους στοιχεία, όπως όνομα, διεύθυνση email και αριθμό τηλεφώνου. Το σύστημα πρέπει να επιβεβαιώνει την εγκυρότητα της διεύθυνσης email μέσω αποστολής ενός συνδέσμου επιβεβαίωσης. Ο λογαριασμός επιτρέπει στους χρήστες να διαχειρίζονται τις κρατήσεις τους, να λαμβάνουν ειδοποιήσεις και να αποθηκεύουν τα αγαπημένα τους καταστήματα.

2. Σύνδεση Χρήστη: Οι χρήστες θα μπορούν να συνδέονται στον λογαριασμό τους χρησιμοποιώντας όνομα χρήστη/διεύθυνση email και κωδικό πρόσβασης. Το σύστημα πρέπει να υποστηρίζει ανάκτηση κωδικού πρόσβασης μέσω email σε περίπτωση που ο χρήστης τον ξεχάσει.

3. Αναζήτηση Καταστήματος: Οι χρήστες θα μπορούν να αναζητούν καταστήματα εστίασης χρησιμοποιώντας διάφορα κριτήρια όπως τοποθεσία (με χρήση διεύθυνσης), είδος κουζίνας, και διαθέσιμες ημερομηνίες και ώρες. Το σύστημα θα εμφανίζει τα αποτελέσματα της αναζήτησης σε μια λίστα με λεπτομέρειες για κάθε κατάστημα, όπως διεύθυνση, φωτογραφίες, αξιολογήσεις και διαθέσιμες ώρες κρατήσεων.

4. Κράτηση Τραπεζιού: Οι χρήστες θα μπορούν να κάνουν κράτηση τραπεζιού σε ένα συγκεκριμένο κατάστημα επιλέγοντας την επιθυμητή ημερομηνία, ώρα και αριθμό ατόμων. Το σύστημα θα πρέπει επίσης να επιτρέπει την επιλογή ειδικών προτιμήσεων ή αιτημάτων, όπως επιθυμητή θέση (π.χ. δίπλα στο παράθυρο) στα σχόλια κρατήσεων.

5. Ακύρωση ή Τροποποίηση Κράτησης: Οι χρήστες θα μπορούν να ακυρώσουν μια υπάρχουσα κράτηση μέσω του λογαριασμού τους. Το σύστημα θα ενημερώνει το κατάστημα για την ακύρωση ή την τροποποίηση και θα στέλνει επιβεβαίωση της αλλαγής στον χρήστη. Η διαδικασία ακύρωσης θα πρέπει να είναι εύκολη και γρήγορη.

6. Προβολή Ιστορικού Κρατήσεων: Οι χρήστες θα μπορούν να δουν το ιστορικό των κρατήσεων τους μέσω του λογαριασμού τους. Το σύστημα θα αποθηκεύει πληροφορίες για όλες τις προηγούμενες και μελλοντικές κρατήσεις, επιτρέποντας στους χρήστες να επανεξετάζουν τις εμπειρίες τους και να κάνουν επαναληπτικές κρατήσεις με ευκολία.

7. Υποστήριξη Πελατών: Οι χρήστες θα μπορούν να επικοινωνήσουν με την υποστήριξη πελατών για βοήθεια ή για να επιλύσουν προβλήματα σχετικά με τις κρατήσεις τους. Το σύστημα θα πρέπει να παρέχει διάφορους τρόπους επικοινωνίας, όπως email και τηλεφωνική υποστήριξη.

8.Περιήγηση στις Εικόνες του Εστιατορίου: Οι χρήστες θα έχουν τη δυνατότητα να περιηγηθούν στις φωτογραφίες του εστιατορίου μέσα από τη σελίδα του καταστήματος. Το σύστημα θα πρέπει να υποστηρίζει την προβολή πολλαπλών εικόνων, δίνοντας στους χρήστες μια οπτική αναπαράσταση του χώρου, του φαγητού και της ατμόσφαιρας του καταστήματος.

9. Προβολή Τοποθεσίας Εστιατορίου σε Χάρτη: Το σύστημα θα ενσωματώνει υπηρεσίες χαρτών (όπως Google Maps) για να επιτρέπει στους χρήστες να δουν την ακριβή τοποθεσία του εστιατορίου.

Οι χρήστες θα μπορούν να λάβουν οδηγίες για να φτάσουν στο κατάστημα ή να εξετάσουν τα κοντινά αξιοθέατα και δημόσια μέσα μεταφοράς.

10. Αποθήκευση και Αφαίρεση Καταστημάτων στα Αγαπημένα: Οι χρήστες θα μπορούν να αποθηκεύσουν εστιατόρια στα αγαπημένα τους, ώστε να έχουν εύκολη πρόσβαση σε αυτά για μελλοντικές κρατήσεις. Επιπλέον, το σύστημα θα παρέχει τη δυνατότητα αφαίρεσης ενός καταστήματος από τα αγαπημένα με ένα απλό κλικ, προσφέροντας ευελιξία στη διαχείριση των προτιμήσεων του χρήστη.

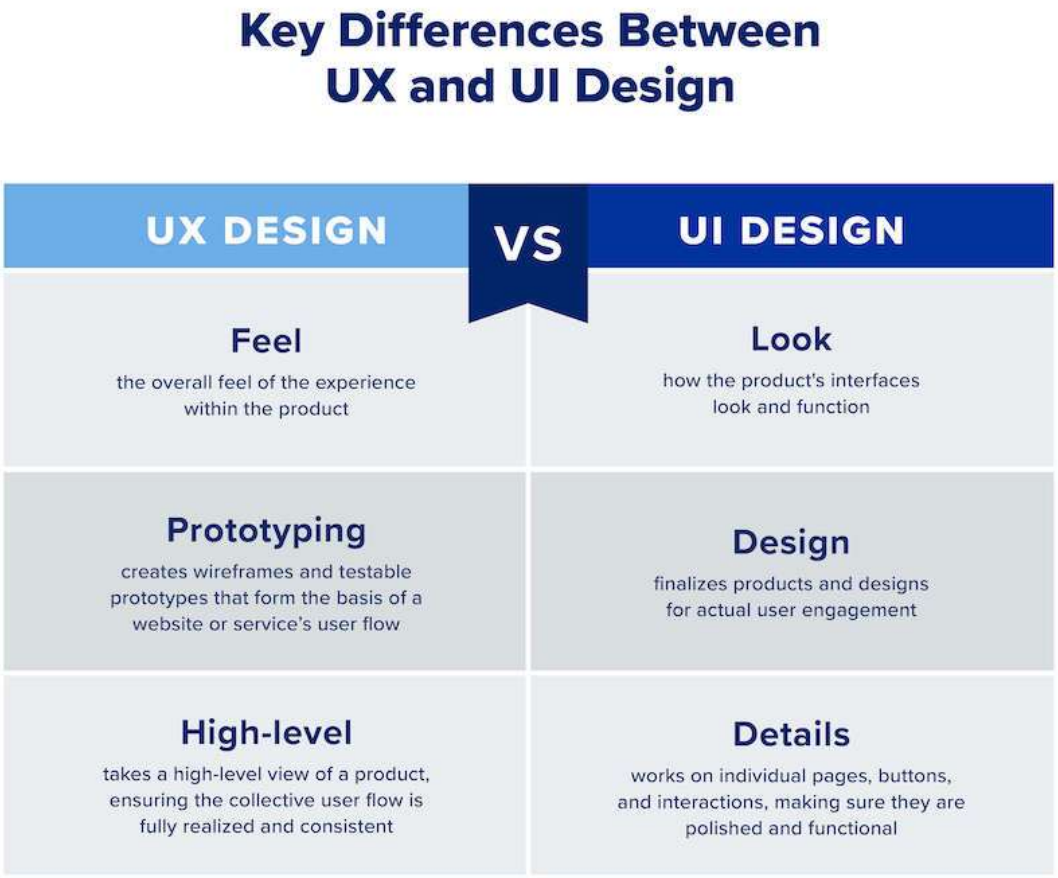
Εν κατακλείδι, αυτές οι προδιαγραφές απαιτήσεων καλύπτουν τις βασικές προδιαγραφές για ένα σύστημα κρατήσεων για καταστήματα εστίασης, εξασφαλίζοντας μια ολοκληρωμένη και ευχάριστη εμπειρία για τους χρήστες. Το σύστημα θα πρέπει να είναι φιλικό προς τον χρήστη, ασφαλές και αξιόπιστο, υποστηρίζοντας την ευκολία στην αναζήτηση και διαχείριση κρατήσεων.

2.2.1 Τι είναι η χρηστικότητα της εφαρμογής;

Η χρηστικότητα της εφαρμογής ενθαρρύνει τη δυνατότητα εκμάθησης. Μια εξαιρετική εφαρμογή θα πρέπει να είναι παρορμητική. Θα χρειαστεί πολύ μικρός χρόνος για να αποκτήσει ένας χρήστης έναν ορισμένο βαθμό συμπάθειας με τη διεπαφή.[13]

Πολλοί άνθρωποι περιπλέκουν το σχεδιασμό UX με τη χρηστικότητα και το αντίστροφο. Ωστόσο, η λειτουργικότητα της εφαρμογής είναι μια προοπτική του UX(User Experience)που συμμετέχει στη συνολική σχέση μεταξύ χρήστη και προϊόντος. Το UX ορίζει όλες τις πτυχές της στάσης ενός χρήστη για μια εφαρμογή, συμπεριλαμβανομένης της χρηστικότητας. Η χρηστικότητα της εφαρμογής είναι ανησυχητική με την αποτελεσματικότητα, την αποδοτικότητα και την απλότητα της επίτευξης στόχων εντός της εφαρμογής.[13]

«Κανένα προϊόν δεν είναι νησί. Ένα προϊόν είναι κάτι περισσότερο από το προϊόν. Είναι ένα συνεκτικό, ολοκληρωμένο σύνολο εμπειριών. Σκεφτείτε όλα τα στάδια ενός προϊόντος ή μιας υπηρεσίας – από τις αρχικές προθέσεις έως τις τελικές σκέψεις, από την πρώτη χρήση έως τη βοήθεια, το σέρβις και τη συντήρηση. Κάντε τους να συνεργάζονται απρόσκοπτα."
— Ντον Νόρμαν, εφευρέτης του όρου «Εμπειρία Χρήστη».[14]



Εικόνα 2.2.1 Βασικές διαφορές μεταξύ σχεδιασμού UX και UI

2.2.2 Ευχρηστία της εφαρμογής

Η αγορά κινητών συσκευών αναπτύσσεται ραγδαία στις μέρες μας. Οι τεχνολογικές βελτιώσεις παρέχουν μεγάλες ευκαιρίες για τη δημιουργία εφαρμογών για κινητά και ιστότοπους. Για την επιτυχία μιας εφαρμογής για κινητά ή ιστότοπου, ένα από τα κύρια ανησυχία, εκτός από θέματα ασφάλειας, είναι η χρηστικότητα.

Μία από τις κύριες ανησυχίες στις εφαρμογές για κινητές συσκευές, δίπλα σε θέματα ασφάλειας, είναι η χρηστικότητα [6], η οποία μπορεί να οριστεί ως «ο βαθμός στον οποίο οι συγκεκριμένοι χρήστες μπορούν να επιτύχουν συγκεκριμένους στόχους σε ένα συγκεκριμένο περιβάλλον, με αποτελεσματικότητα, αποδοτικότητα και ικανοποίηση» [7]. Ο Nielsen δηλώνει ότι «η χρηστικότητα είναι απαραίτητη προϋπόθεση για την επιβίωση στον Ιστό» [8].

Οι 10 γενικές αρχές του Jakob Nielsen για το σχεδιασμό αλληλεπίδρασης. Ονομάζονται "ευρετικά" επειδή είναι γενικοί εμπειρικοί κανόνες και όχι συγκεκριμένες οδηγίες χρηστικότητας.

1. Ορατότητα κατάστασης συστήματος: Η σχεδίαση θα πρέπει πάντα να ενημερώνει τους χρήστες για το τι συμβαίνει, μέσω κατάλληλων σχολίων σε εύλογο χρονικό διάστημα.

2. Ταίριασμα μεταξύ του συστήματος και του πραγματικού κόσμου: Το σχέδιο πρέπει να μιλάει τη γλώσσα των χρηστών. Χρησιμοποιήστε λέξεις, φράσεις και έννοιες οικείες στον χρήστη, αντί για εσωτερική ορολογία. Ακολουθήστε τις συμβάσεις του πραγματικού κόσμου, κάνοντας τις πληροφορίες να εμφανίζονται με φυσική και λογική σειρά.
3. Έλεγχος και ελευθερία χρήστη: Οι χρήστες συχνά εκτελούν ενέργειες κατά λάθος. Χρειάζονται μια ξεκάθαρα σημειωμένη «έξοδος έκτακτης ανάγκης» για να εγκαταλείψουν την ανεπιθύμητη ενέργεια χωρίς να χρειαστεί να περάσουν από μια εκτεταμένη διαδικασία.
4. Συνέπεια και πρότυπα: Οι χρήστες δεν πρέπει να αναρωτιούνται εάν διαφορετικές λέξεις, καταστάσεις ή πράξεις σημαίνουν το ίδιο πράγμα. Ακολουθήστε τις συμβάσεις πλατφόρμας και βιομηχανίας.
5. Πρόληψη σφαλμάτων: Τα καλά μηνύματα σφάλματος είναι σημαντικά, αλλά τα καλύτερα σχέδια αποτρέπουν προσεκτικά την εμφάνιση προβλημάτων στην αρχή. Είτε εξαλείψτε τις συνθήκες που είναι επιρρεπείς σε σφάλματα ή ελέγξτε για αυτές και παρουσιάστε στους χρήστες μια επιλογή επιβεβαίωσης προτού δεσμευτούν στην ενέργεια.
6. Αναγνώριση και όχι ανάκληση: Ελαχιστοποιήστε το φορτίο μνήμης του χρήστη κάνοντας ορατά στοιχεία, ενέργειες και επιλογές. Ο χρήστης δεν πρέπει να θυμάται πληροφορίες από το ένα μέρος της διεπαφής στο άλλο. Οι πληροφορίες που απαιτούνται για τη χρήση του σχεδίου (π.χ. ετικέτες πεδίου ή στοιχεία μενού) πρέπει να είναι ορατές ή εύκολα ανακτώσιμες όταν χρειάζεται.
7. Ευελιξία και αποτελεσματικότητα χρήσης: Οι συντομεύσεις — κρυφές από αρχάριους χρήστες — ενδέχεται να επιταχύνουν την αλληλεπίδραση για τον έμπειρο χρήστη, έτσι ώστε η σχεδίαση να μπορεί να εξυπηρετεί τόσο άπειρους όσο και έμπειρους χρήστες. Επιτρέψτε στους χρήστες να προσαρμόζουν συχνές ενέργειες.
8. Αισθητικό και Μινιμαλιστικό Σχέδιο: Οι διεπαφές δεν πρέπει να περιέχουν πληροφορίες που είναι άσχετες ή σπάνια απαραίτητες. Κάθε επιπλέον μονάδα πληροφοριών σε μια διεπαφή ανταγωνίζεται τις σχετικές μονάδες πληροφοριών και μειώνει τη σχετική ορατότητά τους.
9. Βοηθήστε τους χρήστες να αναγνωρίσουν, να διαγνώσουν και να ανακτήσουν τα σφάλματα: Τα μηνύματα σφάλματος πρέπει να εκφράζονται σε απλή γλώσσα (χωρίς κωδικούς σφαλμάτων), να υποδεικνύουν με ακρίβεια το πρόβλημα και να προτείνουν εποικοδομητικά μια λύση.
10. Βοήθεια και τεκμηρίωση: Είναι καλύτερο εάν το σύστημα δεν χρειάζεται καμία πρόσθετη εξήγηση. Ωστόσο, μπορεί να είναι απαραίτητο να παρέχετε τεκμηρίωση για να βοηθήσετε τους χρήστες να κατανοήσουν πώς να ολοκληρώσουν τις εργασίες τους.

Αυτές οι γενικές αρχές ακολουθούνται σαν κανόνας για όλες τις εφαρμογές. Παρόλα αυτά υπάρχουν και άλλες ,όπως βλέπουμε παρακάτω και αφορούν περισσότερο τις εμπορικές εφαρμογές. Ο στόχος τους είναι να προωθήσουν υπηρεσίες και προϊόντα σε μια συγκεκριμένη ομάδα καταναλωτών:

11. **Προσαρμοσμένη εμπειρία χρήστη**: Οι εμπορικές εφαρμογές συχνά ενσωματώνουν εξατομίκευση για να προσφέρουν στους χρήστες μια πιο στοχευμένη εμπειρία. Αυτό μπορεί να περιλαμβάνει προτάσεις προϊόντων ή υπηρεσιών βάσει των προηγούμενων αγορών ή της συμπεριφοράς περιήγησης του χρήστη. Η προσαρμογή αυτή μπορεί να ενισχύσει την αφοσίωση του χρήστη και να αυξήσει την πιθανότητα επαναλαμβανόμενων αγορών.
12. **Διαφάνεια στην πολιτική απορρήτου**: Είναι ζωτικής σημασίας για τις εμπορικές εφαρμογές να έχουν σαφείς και προσβάσιμες πολιτικές απορρήτου. Οι χρήστες πρέπει να γνωρίζουν πώς

χρησιμοποιούνται και προστατεύονται τα προσωπικά τους δεδομένα. Η διαφάνεια αυτή ενισχύει την εμπιστοσύνη του χρήστη και μπορεί να βελτιώσει τη φήμη της εφαρμογής.

13. **Εύκολη διαδικασία πληρωμής**: Στις εμπορικές εφαρμογές, η ευχρηστία της διαδικασίας πληρωμής είναι κρίσιμη. Η διαδικασία πρέπει να είναι απλή, γρήγορη και ασφαλής, με επιλογές για αποθήκευση στοιχείων πληρωμής για μελλοντικές αγορές. Οποιαδήποτε δυσκολία ή καθυστέρηση στη διαδικασία αυτή μπορεί να οδηγήσει σε εγκατάλειψη του καλαθιού και απώλεια εσόδων.
14. **Προώθηση ειδικών προσφορών**: Οι εμπορικές εφαρμογές συχνά χρησιμοποιούν ειδοποιήσεις ή αναδυόμενα παράθυρα για να ενημερώνουν τους χρήστες για ειδικές προσφορές, εκπτώσεις ή νέες κυκλοφορίες προϊόντων. Αυτές οι προωθήσεις πρέπει να είναι διακριτικές και σχετικές με τα ενδιαφέροντα του χρήστη, αποφεύγοντας να γίνουν ενοχλητικές ή καταπιεστικές.
15. **Ενσωμάτωση κοινωνικών δικτύων**: Η δυνατότητα να μοιράζονται οι χρήστες τα προϊόντα ή τις εμπειρίες τους στα κοινωνικά δίκτυα μπορεί να είναι ένας ισχυρός τρόπος προώθησης για τις εμπορικές εφαρμογές. Αυτό όχι μόνο ενισχύει τη διάδοση της εφαρμογής αλλά και προωθεί τα προϊόντα της σε ένα ευρύτερο κοινό μέσω της κοινωνικής απόδειξης.
16. **Αξιολόγηση και σχόλια χρηστών**: Οι εφαρμογές που επιτρέπουν στους χρήστες να αξιολογούν και να σχολιάζουν προϊόντα παρέχουν πολύτιμες πληροφορίες για άλλους πιθανούς αγοραστές και για την ίδια την επιχείρηση. Η δυνατότητα αυτή μπορεί να ενισχύσει την εμπιστοσύνη των καταναλωτών και να παρέχει άμεση ανατροφοδότηση για τη βελτίωση των προϊόντων και των υπηρεσιών.

Η επιτυχημένη ευχρηστία στις εμπορικές εφαρμογές είναι αυτή που συνδυάζει την ικανοποίηση των χρηστών με την επίτευξη των επιχειρηματικών στόχων. Με αυτόν τον τρόπο, οι εφαρμογές γίνονται πιο ελκυστικές και αποδοτικές, συμβάλλοντας στην αύξηση της πιστότητας των χρηστών και της απόδοσης της επένδυσης.

2.2.3 Material Design

Το Material Design (MD) κυκλοφορεί από το 2014. Είναι ένα σύστημα σχεδιασμού που αναπτύχθηκε από την Google. Είναι ένας οπτικός οδηγός για κίνηση και αλληλεπιδράσεις που μπορεί να προσαρμοστεί και να παραδοθεί σε διάφορες συσκευές και πλατφόρμες. Η αρχική έμπνευση για τη δημιουργία του συστήματος Material Design ήταν η διάταξη που βασίζεται σε κάρτες που χρησιμοποιεί τώρα η Google. Ο σχεδιαστής Matías Duarte εξήγησε ότι " σε αντίθεση με το πραγματικό χαρτί, το ψηφιακό υλικό μπορεί να επεκταθεί και να μορφοποιηθεί έξυπνα. Το material έχει φυσικές επιφάνειες και άκρα. Οι σκιές δίνουν νόημα σε αυτό που αγγίζεις."[2][3][4] Η Google υποστηρίζει ότι η καινούρια σχεδιαστική τους γλώσσα βασίζεται στο χαρτί και το μολύβι.

Η MD παρέχει μια ολοκληρωμένη βιβλιοθήκη στοιχείων που καλύπτουν τη χρήση, τη συμπεριφορά και τις προδιαγραφές για κάθε μοτίβο. Η δημιουργία μιας διεπαφής χρήστη είναι εύκολη και επιρρεπής σε σημαντικά κενά: όλες οι καταστάσεις του συστήματος υλοποιούνται σε βιβλιοθήκες και είναι έτοιμες για χρήση. Πολλά στοιχεία είναι διαθέσιμα, όπως εικονίδια υλικού και σύμβολα για εμφάνιση, πλοήγηση, είσοδοι, ενέργειες και επικοινωνία με τον χρήστη.

Περιτυλιγμένο σε εκτενή τεκμηρίωση, γίνεται ένας ανεκτίμητος σύντροφος τόσο για τους σχεδιαστές όσο και για τους προγραμματιστές. Σκεφτείτε το σαν τα καθιερωμένα μοτίβα σχεδίασης που βρίσκετε

στα περιοδικά – κάθε ενότητα ορίζεται μοναδικά, συνδυάζει αρμονικά διατάξεις, γραμματοσειρές και χρώματα. Τώρα, φανταστείτε αυτή την ιδέα να μεταβαίνει απρόσκοπτα στην ψηφιακή σφαίρα, όπου σχεδιαστές διαφορετικών βιομηχανιών υφαίνουν τη δημιουργικότητά τους σε βιβλιοθήκες μοτίβων μέσα σε πολύπλοκα συστήματα.

Είναι ένας δυναμικός χορός, που πλοηγείται στη λεπτή αλληλεπίδραση μεταξύ των μοτίβων σχεδιασμού λογισμικού και των βάσεων κωδικών του μπροστινού μέρους. Η εξέλιξη των μοτίβων σχεδίασης, που συχνά αναπτύσσεται σε επίπεδο εξαρτημάτων και αλληλεπιδράσεων, προσκαλεί τους σχεδιαστές να παραμείνουν ευέλικτοι και σε συγχρονισμό με τον συναρπαστικό οδικό χάρτη της ανάπτυξης προϊόντων. Εξασφαλίζει ότι η σχεδίαση παραμένει όχι μόνο λειτουργική αλλά και με επίκεντρο τον χρήστη.

Ο οδηγός Σχεδιασμού Υλικού βρίσκεται υπό συνεχή αναθεώρηση και βελτίωση. Για παράδειγμα, η νέα τεκμηρίωση για το Material Design 3 (MD3) δημοσιεύτηκε πρόσφατα με πλούσια τεκμηρίωση που υποστηρίζει γρήγορη ενσωμάτωση και εξηγεί σε βάθος τις αλληλεπιδράσεις, τις συμπεριφορές, τις εξαρτήσεις και τη χρήση που διευκολύνουν την ανάπτυξη αισθητικών, λειτουργικών προϊόντων.

Αυτή η καλά προετοιμασμένη τεκμηρίωση σημαίνει ότι οι προγραμματιστές μπορούν να χρησιμοποιήσουν αμέσως ακόμη και την πιο πρόσφατη έκδοση του Material Design, καθιστώντας το ένα από τα πιο φιλικά προς τους προγραμματιστές συστήματα του σήμερα. Μόλις κυκλοφορήσει, η μόνη εργασία είναι να διατηρείτε το προϊόν ενημερωμένο και χωρίς σφάλματα. Η επίλυση αυτών των ζητημάτων είναι μια μακροπρόθεσμη διαδικασία που συνήθως περιλαμβάνει νέες εξελίξεις που ενσωματώνονται κάθε λίγους μήνες.

Από την άποψη των προθεσμιών και της διαχείρισης, η αποτελεσματικότητα, η καλά διατηρημένη τεκμηρίωση και η υποστήριξη συντήρησης 24/7 είναι όλα πλεονεκτήματα του συστήματος σχεδιασμού υλικού.

Το Material είναι ένα σύστημα σχεδιασμού που δημιουργήθηκε για να βοηθήσει τις ομάδες να δημιουργήσουν ψηφιακές εμπειρίες υψηλής ποιότητας για Android, iOS, Flutter και τον Ιστό.

Μια καλά εφαρμοσμένη βιβλιοθήκη μοτίβων και διαδραστικών συμπεριφορών διασφαλίζει ότι τα προϊόντα εφαρμογής MD λειτουργούν γρήγορα και αποτελεσματικά. Επειδή η απόδοση είναι ένας κρίσιμος παράγοντας εμπειρίας χρήστη , τα στοιχεία MD μπορούν να βοηθήσουν στη διατήρηση των χρηστών και στη βελτίωση της συνολικής αφοσίωσης.

Εν κατακλείδι εάν μια εφαρμογή δημιουργείται κυρίως για την πλατφόρμα Android και του ιστού παράλληλα, τότε η χρήση του Material Design είναι μια εύκολη επιλογή. Λόγω της ευρείας υιοθέτησης από την Google, κάθε εφαρμογή που βασίζεται στις αρχές του Material Design θα μοιάζει με εγγενή εφαρμογή.

Κεφάλαιο 3ο: Τεχνολογίες

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο, επικεντρωθήκαμε στις κύριες τεχνολογίες που αποτέλεσαν τη βάση για την ανάπτυξη της εφαρμογής μας. Αρχικά, αξιοποιήσαμε το **Flutter**, ένα σύγχρονο framework που μας επέτρεψε να αναπτύξουμε την εφαρμογή για πολλαπλές πλατφόρμες, όπως Android και Web, χρησιμοποιώντας έναν ενιαίο κώδικα. Η χρήση των **Widgets** του Flutter μας έδωσε τη δυνατότητα να σχεδιάσουμε πλούσια και διαδραστικά περιβάλλοντα χρήστη, προσφέροντας μια εξαιρετική εμπειρία στον τελικό χρήστη.

Η εφαρμογή αναπτύχθηκε στη γλώσσα προγραμματισμού **Dart**, η οποία αποτελεί τον πυρήνα του Flutter. Η Dart επιλέχθηκε για την υψηλή της απόδοση και την ευκολία που προσφέρει στη δημιουργία και συντήρηση κώδικα. Παράλληλα, για τη διαχείριση του κώδικα και τη συνεργασία της ομάδας ανάπτυξης, χρησιμοποιήθηκαν το **Git** και το **GitHub**. Αυτά τα εργαλεία επέτρεψαν την παρακολούθηση των αλλαγών στον κώδικα, τη διασφάλιση της ακεραιότητας του έργου και την ομαλή συνεργασία μεταξύ μας.

Στη συνέχεια, αναλύσαμε τη χρήση του **Firebase** για την αυθεντικοποίηση χρηστών και τη διαχείριση της βάσης δεδομένων. Η ενσωμάτωση του **Firebase Authentication** και του **Cloud Firestore** προσέφερε μια ασφαλή και αξιόπιστη λύση για την αποθήκευση δεδομένων και την άμεση ενημέρωσή τους σε πραγματικό χρόνο, γεγονός που συνέβαλε σημαντικά στη λειτουργικότητα της εφαρμογής.

Τέλος, η ανάπτυξη του κώδικα πραγματοποιήθηκε στο **Visual Studio Code**, ένα ευέλικτο και προσαρμόσιμο περιβάλλον ανάπτυξης που υποστηρίζει ένα ευρύ φάσμα γλωσσών και εργαλείων. Η επιλογή του Visual Studio Code βοήθησε στην αποτελεσματική ανάπτυξη του έργου, διευκολύνοντας τη διαδικασία του debugging και την ολοκλήρωση του κώδικα.

Συνοψίζοντας, η πτυχιακή εργασία μας παρουσιάζει ένα παράδειγμα επιτυχημένης ενοποίησης προηγμένων τεχνολογιών για την ανάπτυξη μιας σύγχρονης και αποδοτικής εφαρμογής. Μέσα από τη χρήση του Flutter, της γλώσσας Dart, του Firebase και άλλων εργαλείων, καταφέραμε να δημιουργήσουμε μια πλατφόρμα που προσφέρει υψηλή απόδοση, ασφάλεια και ευχρηστία, ανταποκρινόμενη στις απαιτήσεις της σύγχρονης ψηφιακής εποχής.

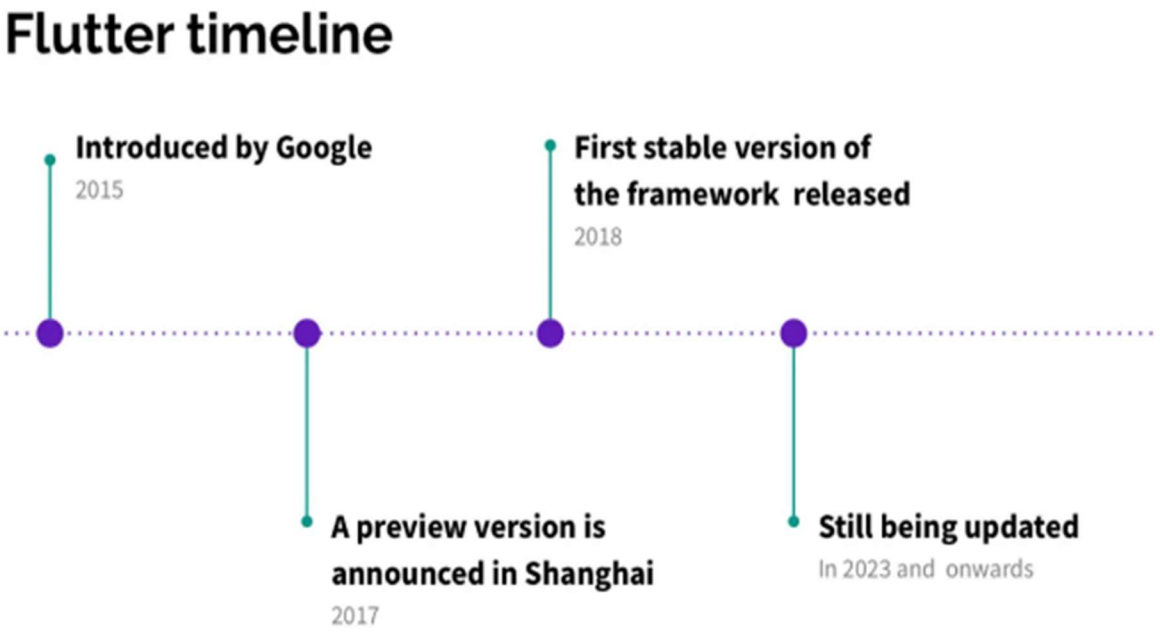
3.2 Εμβάθυνση στις Τεχνολογίες

Μετά την εισαγωγή στις τεχνολογίες που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής μας, είναι σημαντικό να εμβαθύνουμε ξεχωριστά σε κάθε μία από αυτές. Στις επόμενες ενότητες, θα αναλύσουμε τις τεχνολογίες που επιλέχθηκαν, εξετάζοντας τον ρόλο και τη συνεισφορά τους στην υλοποίηση της εφαρμογής μας.

3.3 Flutter

Το Flutter είναι ένα κιτ ανάπτυξης λογισμικού UI ανοιχτού κώδικα που δημιουργήθηκε από την Google. Ως μέρος του κιτ εργαλείων διεπαφής χρήστη της Google, αυτό το ευέλικτο πλαίσιο πολλαπλών πλατφορμών είναι ιδανικό για τη δημιουργία επεκτάσιμων και ελκυστικών εγγενών εφαρμογών και διεπαφών χρήστη. Δεδομένου ότι ένα από τα καθοριστικά χαρακτηριστικά αυτού του πλαισίου είναι η σχέση κόστους-αποτελεσματικότητας και η υψηλότερη απόδοση, ευνοείται τόσο από τους επιχειρηματίες, τις μικρές επιχειρήσεις όσο και από τις επιχειρήσεις.

Το Flutter υποστηρίζει τη σταθερή επανάληψη φόρτωσης κατά την ανάπτυξη, κάτι που θεωρείται σημαντικός παράγοντας για την ενίσχυση του κύκλου ανάπτυξης. Το Stateful hot-reload ουσιαστικά υλοποιείται με την έγχυση ενημερωμένου πηγαίου κώδικα στο τρέχον Dart VM χωρίς αλλαγή της εσωτερικής δομής της εφαρμογής, επομένως όλες οι μεταβάσεις και οι ενέργειες της εφαρμογής θα διατηρηθούν μετά θερμή επαναφόρτωση [18]



Εικόνα 3.3.1 Flutter evolution stages

ΠΛΕΟΝΕΚΤΗΜΑ ΤΟΥ FLUTTER

Α. Ανάπτυξη μεταξύ πλατφορμών: Μπορείτε να χρησιμοποιήσετε το Flutter για να δημιουργήσετε εφαρμογές για διαφορετικές συσκευές, όπως mobile/web. Αυτό σημαίνει ότι πρέπει να γράψετε μόνο μια μοναδική βάση κωδικών για όλες αυτές τις συσκευές, κάτι που είναι μεγάλη εξοικονόμηση χρόνου για τους προγραμματιστές.

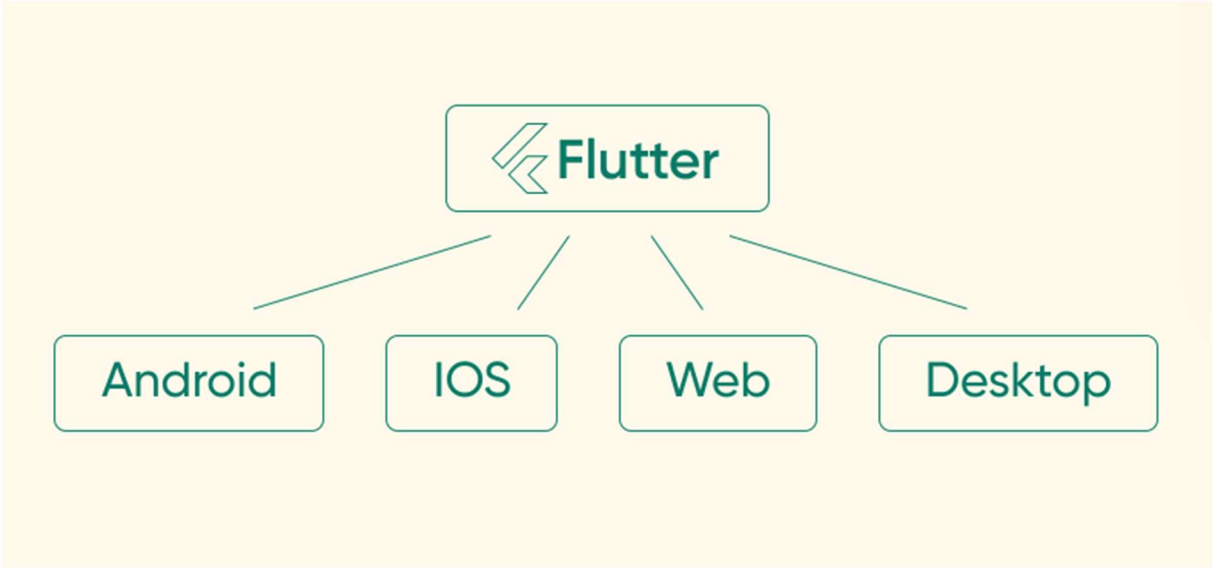
Β. Υψηλή απόδοση: Οι εφαρμογές ιστού Flutter λειτουργούν πολύ καλά και έχουν ομαλές κινήσεις. Είναι έτσι γιατί το Flutter χρησιμοποιεί τη δική του μηχανή απόδοσης που είναι φτιαγμένη για να είναι γρήγορη και αποτελεσματική.

Γ. Εγγενής εμφάνιση και αίσθηση: Οι εφαρμογές ιστού Flutter έχουν την εμφάνιση και την αίσθηση ότι ανήκουν σε κάθε συσκευή. Αυτό συμβαίνει επειδή χρησιμοποιεί δομικά στοιχεία από κάθε συσκευή για να φαίνεται σωστή η διεπαφή της.

Δ. Πλούσιο στοιχείο διεπαφής χρήστη: Το Flutter έχει πολλά εκπληκτικά κουμπιά και πράγματα που μπορείτε να χρησιμοποιήσετε για να κάνετε τις εφαρμογές ιστού σας να φαίνονται υπέροχες(γνωστό ως Widgets). Είναι εύκολο να δημιουργήσετε εκπληκτικές και φιλικές προς το χρήστη εφαρμογές ιστού με αυτές.

3.4 Cross-platform

Η κύρια έκκληση για λύσεις πολλαπλών πλατφορμών έναντι της εγγενούς ανάπτυξης είναι η ελαχιστοποίηση, το μέγεθος της ομάδας ανάπτυξης, ο χρόνος και οι πόροι. Ο άλλος βασικός λόγος είναι αυτό που παρακινεί προγραμματιστές και επιχειρήσεις να κάνουν αυτές τις εφαρμογές στην πρώτη θέση. Είναι μια επέκταση στις υπάρχουσες ιστοσελίδες τους. Οι περισσότερες εταιρείες και νεοφυείς επιχειρήσεις επεκτείνουν το προϊόν ή την υπηρεσία τους σε εφαρμογές για κινητές συσκευές λόγω της αυξανόμενης επισκεψιμότητας που λαμβάνουν από πλατφόρμες για κινητές συσκευές ιστοσελίδες. Οι λύσεις μεταξύ πλατφορμών προτιμώνται συνήθως από τους προγραμματιστές ιστού όπως οι περισσότερες από αυτές τις λύσεις χρησιμοποιούν τεχνολογίες Ιστού σε πλατφόρμες κινητής τηλεφωνίας. Ενιαία βάση κώδικα, άρα και μοναδική η ομάδα ανάπτυξης για εφαρμογές που τρέχουν σε πολλαπλές πλατφόρμες, είναι μια από τις καλύτερες επιλογές με κοινά βήματα για λύσεις μεταξύ πλατφορμών.



Εικόνα 3.4.1 A cross-platform app development framework Flutter

3.5 Dart

Το Dart είναι μια αντικειμενοστραφή , βασισμένη σε κλάσεις , γλώσσα συλλογής σκουπιδιών με σύνταξη τύπου C. [9]Βοηθά στη δημιουργία ιστού και εφαρμογές για κινητά, καθώς και εφαρμογές διακομιστή και επιτραπέζιους υπολογιστές. Το Dart έχει σχεδιαστεί για να λειτουργεί αποτελεσματικά σε συσκευές όπως ως έξυπνα τηλέφωνα και tablet. Όλες οι εφαρμογές Flutter είναι γραμμένες στη

γλώσσα προγραμματισμού Dart. Το Dart ήταν αρχικά η προσπάθεια της Google να δημιουργήσει μια εναλλακτική στην JavaScript συμπεριλαμβάνοντας το Dart VM στο Google Chrome, το οποίο θα επέτρεπε σε αυτό το πρόγραμμα περιήγησης ιστού να ερμηνεύει και να εκτελεί τον κώδικα Dart. Ωστόσο, η Google απέτυχε να το κάνει, γιατί έπρεπε ακόμα υποστήριξη διαλειτουργικότητας με JavaScript που δεν επέτρεψε τη δημιουργία της γλώσσας ήθελαν. Ωστόσο, το Dart έχει βρει τη θέση του ως γλώσσα για το Flutter, και η κύρια χρήση του στις μέρες μας είναι το Flutter. Η κεντρική σελίδα του Dart το δείχνει χρησιμοποιήστε κυρίως με Flutter παρά μόνο. (Ιστότοπος Dart 2020.) Σύμφωνα με την Προδιαγραφή Γλώσσας Προγραμματισμού Dart (2019), είναι μια γλώσσα προγραμματισμού βασισμένη σε τάξη, μεμονωμένης κληρονομικότητας, καθαρά αντικειμενοστραφής. Το Dart είναι επίσης πληκτρολογημένο προαιρετικά. Υποστηρίζει τροποποιημένα γενόσημα.

Το Dart είναι χρήσιμο για τη δημιουργία διαφορετικών ειδών εφαρμογών:

α) Εφαρμογές Ιστού: Μπορείτε να χρησιμοποιήσετε το Dart για να δημιουργήσετε γρήγορες και διαδραστικές εφαρμογές Ιστού.

β) Εφαρμογές για κινητά: Το Dart λειτουργεί για τη δημιουργία εφαρμογών σε τηλέφωνα Android και iOS.

γ) Εφαρμογές επιφάνειας εργασίας: Μπορείτε να χρησιμοποιήσετε το Dart για εφαρμογές σε υπολογιστές Windows, macOS και Linux.

δ) Εφαρμογές διακομιστή: Το Dart είναι καλό για τη δημιουργία στοιχείων όπως API και μικροϋπηρεσίες που εκτελούνται σε διακομιστές.

Widgets in Flutter

Κάθε στοιχείο στην οθόνη της εφαρμογής Flutter είναι ένα γραφικό στοιχείο. Η προβολή της οθόνης εξαρτάται πλήρως από την επιλογή και τη σειρά των γραφικών στοιχείων που χρησιμοποιούνται για τη δημιουργία των εφαρμογών. Και η δομή του κώδικα των εφαρμογών είναι ένα δέντρο γραφικών στοιχείων.

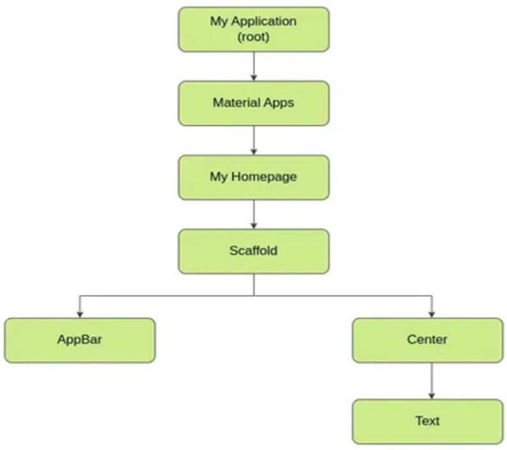
Τύποι γραφικών στοιχείων

Υπάρχουν γενικά δύο τύποι γραφικών στοιχείων στο flutter:

- Stateless Widget
- Stateful Widget

1. Stateless Widget: είναι ένας τύπος γραφικού στοιχείου που μόλις κατασκευαστεί, τότε οι ιδιότητες και η κατάστασή του δεν μπορούν να αλλάξουν. Αυτά τα γραφικά στοιχεία είναι αμετάβλητα, αφού δημιουργηθούν δεν μπορούν να τροποποιηθούν.

2. Stateful Widget: είναι ένας τύπος widget που μπορεί να αλλάξει κατάσταση. Μπορεί να διατηρήσει και να ενημερώσει την εμφάνιση στην απόκριση στην αλλαγή κατάστασης.



Εικόνα 3.5.1 Δέντρο διάταξης των Stateful και Stateless Widgets

3.6 Visual Studio Code

Το Visual Studio Code , κοινώς γνωστό και ως VS Code , [5] είναι ένα πρόγραμμα επεξεργασίας πηγαίου κώδικα που αναπτύχθηκε από τη Microsoft για Windows , Linux , macOS και προγράμματα περιήγησης ιστού . [10] [11] Οι δυνατότητες περιλαμβάνουν υποστήριξη για εντοπισμό σφαλμάτων , επισήμανση σύνταξης , έξυπνη συμπλήρωση κώδικα , αποσπάσματα , ανακατασκευή κώδικα και ενσωματωμένο έλεγχο έκδοσης με το Git . Οι χρήστες μπορούν να αλλάξουν το θέμα , τις συντομεύσεις πληκτρολογίου , τις προτιμήσεις και να εγκαταστήσουν επεκτάσεις που προσθέτουν λειτουργικότητα.

3.7 Git

Το Git είναι ένα σύστημα ελέγχου κατανεμημένων εκδόσεων ανοιχτού κώδικα. Έχει σχεδιαστεί για να χειρίζεται μικρά έως μεγάλα έργα με υψηλή ταχύτητα και αποτελεσματικότητα. Αναπτύχθηκε για να συντονίζει την εργασία μεταξύ των προγραμματιστών. Ο έλεγχος έκδοσης μας επιτρέπει να παρακολουθούμε και να συνεργαζόμαστε με τα μέλη της ομάδας μας στον ίδιο χώρο εργασίας. Το Git είναι το θεμέλιο πολλών υπηρεσιών όπως το GitHub και το GitLab, αλλά μπορούμε να χρησιμοποιήσουμε το Git χωρίς να χρησιμοποιήσουμε άλλες υπηρεσίες Git. Το Git μπορεί να χρησιμοποιηθεί ιδιωτικά και δημόσια.Το Git δημιουργήθηκε από τον Linus Torvalds το 2005 για την ανάπτυξη του Linux Kernel. Χρησιμοποιείται επίσης ως ένα σημαντικό κατανεμημένο εργαλείο ελέγχου έκδοσης για τα DevOps. Το Git μαθαίνεται εύκολα και έχει γρήγορη απόδοση. Είναι ανώτερο από άλλα εργαλεία SCM όπως το Subversion, το CVS, το Perforce και το ClearCase.

3.8 GitHub

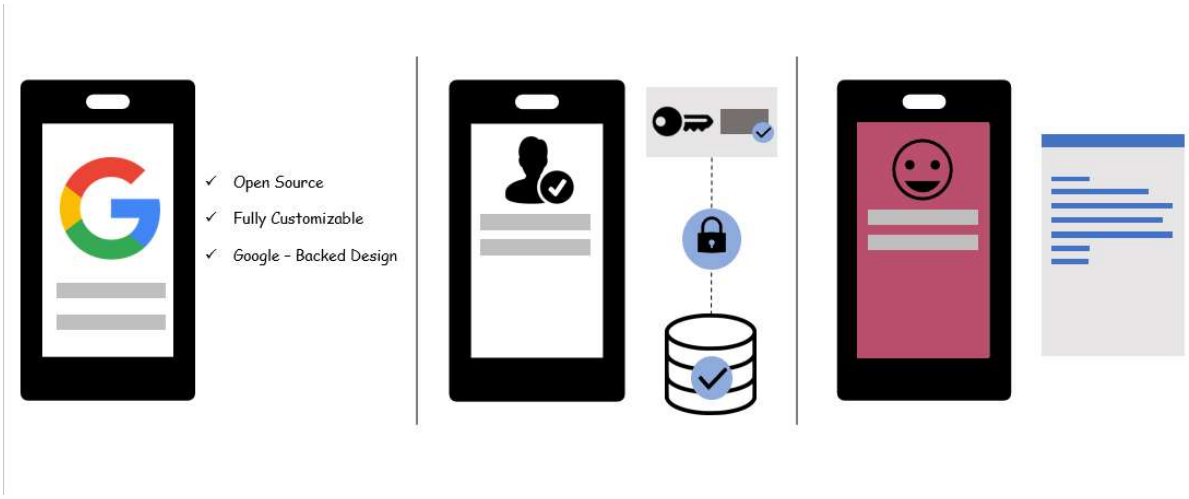
Το GitHub είναι μια πλατφόρμα προγραμματιστών που επιτρέπει στους προγραμματιστές να δημιουργούν, να αποθηκεύουν, να διαχειρίζονται και να μοιράζονται τον κώδικά τους. Χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής RESERVE-EAT καθώς υπήρχε συνεργασία και ταυτόχρονη χρήση του πηγαίου κώδικα της εφαρμογής.

3.9 Firebase

Το Firebase [19] είναι μια πλατφόρμα που αναπτύχθηκε από την Google για τη δημιουργία και την ανάπτυξη εφαρμογών για κινητές συσκευές και ιστοσελίδες. Παρέχει μια σειρά από εργαλεία και υπηρεσίες που βοηθούν τους προγραμματιστές να αναπτύξουν, να βελτιστοποιήσουν και να κλιμακώσουν τις εφαρμογές τους χωρίς να χρειάζεται να διαχειρίζονται τις υποδομές στο παρασκήνιο. Το Firebase προσφέρει υπηρεσίες όπως η αποθήκευση δεδομένων σε πραγματικό χρόνο (Firebase Realtime Database), η αποθήκευση και διαχείριση αρχείων (Firebase Storage), η ταυτοποίηση χρηστών (Firebase Authentication), οι αναλυτικές υπηρεσίες (Firebase Analytics) και πολλές άλλες δυνατότητες που διευκολύνουν την ανάπτυξη εφαρμογών. Το Firebase είναι ιδιαίτερα δημοφιλές λόγω της ευκολίας χρήσης του, της δυνατότητας για γρήγορη ανάπτυξη εφαρμογών και της βαθιάς ενσωμάτωσής του με άλλες υπηρεσίες της Google, όπως το Google Cloud.

Authentication

Επί του παρόντος, πολλές υπηρεσίες Ιστού χρησιμοποιούν έλεγχο ταυτότητας για την αναγνώριση των χρηστών και την ασφάλεια των δεδομένων μέσω ελέγχου πρόσβαση στο περιεχόμενό τους. Για παράδειγμα, η Google ζητά λογαριασμό και κωδικό πρόσβασης για να συνδεθεί σε ορισμένες εφαρμογές. Ο έλεγχος ταυτότητας είναι πιο περίπλοκος λόγω ελέγχου ταυτότητας τρίτου μέρους – API [15]. Ως εκ τούτου, είναι στόχος της ομάδας Firebase να ολοκληρώσει τον έλεγχο ταυτότητας για να υποστηρίξει τους προγραμματιστές με την παροχή ενός API που επιτρέπει στους χρήστες να συνδέονται ή να εγγράφονται από ομοσπονδιακούς παρόχους. Συνεργάζεται επίσης με τη βάση δεδομένων σε πραγματικό χρόνο για τον έλεγχο της πρόσβασης, όπως στο παρακάτω σχήμα.[34]



Εικόνα 3.9.1 Firebase Authentication

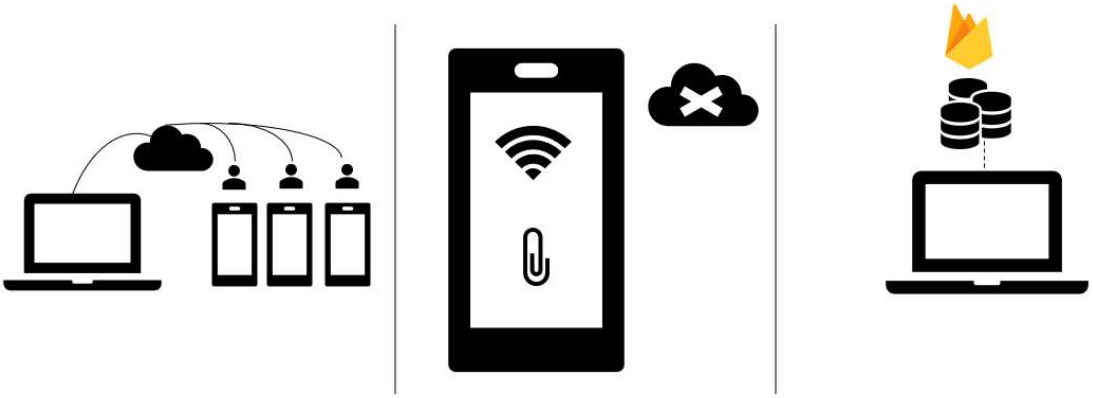
Οι κοινοί ομοσπονδιακοί πάροχοι είναι οι Google, GitHub, Facebook και Twitter. Εάν οι προγραμματιστές χρησιμοποιήσουν το Firebase, δεν είναι απαραίτητο να επαναληφθεί η διαδικασία σύνδεσης, επειδή είναι ήδη ενσωματωμένη στο σύστημα ελέγχου ταυτότητας αυτών των παρόχων. Εάν ο χρήστης έχει έναν υπάρχοντα λογαριασμό, θα συνδεθεί αμέσως στο σύστημα ελέγχου ταυτότητας[16]. Με αυτή τη διαδικασία υλοποιούνται εύκολα εφαρμογές πολλαπλών πλατφορμών (crossplatform apps) με συνεπείς μηχανισμούς αυθεντικοποίησης και εξουσιοδότησης χρηστών.

Συνολικά, η αυθεντικοποίηση της Firebase αποτελεί μια ισχυρή και ευέλικτη υπηρεσία που επιτρέπει στους προγραμματιστές να ενσωματώνουν γρήγορα και εύκολα την πιστοποίηση χρηστών στις

εφαρμογές τους, χωρίς να ανησυχούν για τις περίπλοκες διαδικασίες ταυτοποίησης και ασφάλειας που κρύβονται πίσω από τα παρασκήνια.

Realtime Database

Η κλασική βάση δεδομένων σε πραγματικό χρόνο δείχνει το σύστημα βάσης δεδομένων που ξεπερνά τον περιορισμό σε πραγματικό χρόνο για την εδραίωση ενός αξιόπιστου συστήματος βάσης δεδομένων. Αυτό είναι το βασικό προϊόν του Firebase καθώς είναι ένα απλό χαρακτηριστικό χρησιμοποιώντας το API. Συγχρονίζει νέες πληροφορίες ως JSON σε όλες τις συσκευές, στις οποίες η βάση δεδομένων σε πραγματικό χρόνο συνεργάζεται με την έκδοση ιστού και κινητής τηλεφωνίας στην ίδια συσκευή και μοιράζεται με μια άλλη συσκευή σε χιλιοστά του δευτερολέπτου. Επιπλέον, αυτό το προϊόν βελτιστοποιεί την υπηρεσία εκτός σύνδεσης για αποθήκευση των πληροφοριών χρήστη στο τοπικό cache και χρησιμοποιεί SDK βάσης δεδομένων, ώστε όταν ο χρήστης είναι συνδεδεμένος, τα δεδομένα συγχρονίζονται αυτόματα. Το Firebase σε πραγματικό χρόνο ορίζει τη δομή δεδομένων και προστατεύεται από τους κανόνες ασφαλείας της βάσης δεδομένων, που καθορίζουν άτομα που μπορούν να έχουν πρόσβαση σε συγκεκριμένες πληροφορίες [17]. Με μερικούς κωδικούς από πελάτες, περιγράφει ποιος μπορεί να δει συγκεκριμένες πληροφορίες. Δεν είναι απαραίτητο να υπάρχει διακομιστής λειτουργίας ή συντήρησης επειδή η υπηρεσία cloud φιλοξενεί μια βάση δεδομένων σε πραγματικό χρόνο (βασισμένη σε NoSQL βάση δεδομένων). Όπως και άλλα προϊόντα, η βάση δεδομένων σε πραγματικό χρόνο είναι διαθέσιμη για Android, IOS, Web, C++, Unity και JavaScript[17]. Η βάση δεδομένων σε πραγματικό χρόνο μπορεί να κλιμακώσει έως και 100.000 ταυτόχρονες συνδέσεις και 1.000/δευτερόλεπτο για κάθε βάση δεδομένων.



Εικόνα 3.9.2 The Real-time Database

Cloud Firestore

Από την άλλη πλευρά, το Cloud Firestore είναι μια ευέλικτη, επεκτάσιμη βάση δεδομένων για ανάπτυξη κινητών, ιστού και διακομιστών από το Firebase και το Google Cloud Platform. Έχει σχεδιαστεί για αποθήκευση και συγχρονισμό δεδομένων εφαρμογών σε παγκόσμια κλίμακα, προσφέροντας ενημερώσεις σε πραγματικό χρόνο και υποστήριξη εκτός σύνδεσης. Το Firestore οργανώνει δεδομένα

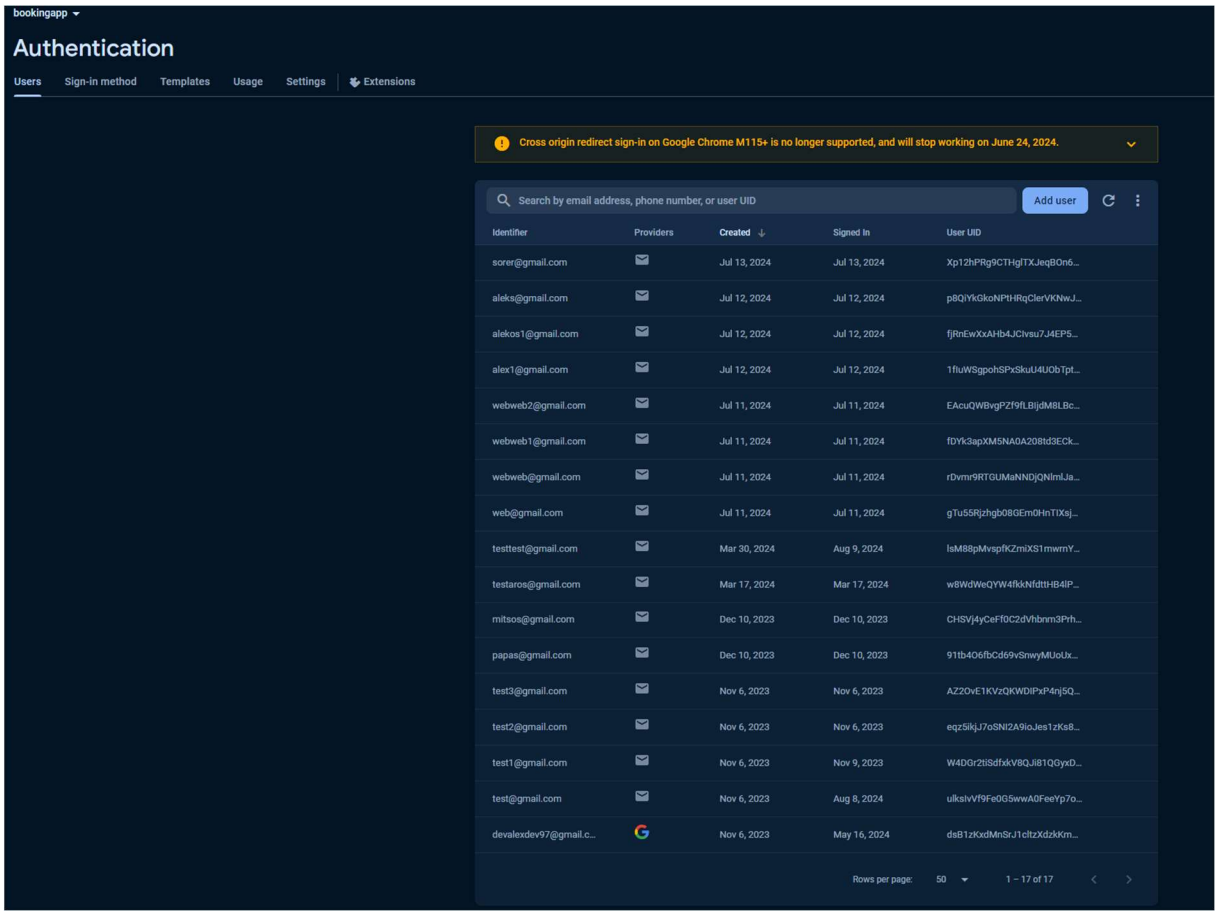
σε συλλογές και έγγραφα και επιτρέπει την αναζήτηση, την ευρετηρίαση και την ισχυρή δόμηση δεδομένων για εφαρμογές.

Κεφάλαιο 4ο: Ανάπτυξη Εφαρμογής και Περιγραφή Λειτουργίας

4.1 Ανάπτυξη Back-End

Το back-end της εφαρμογής μας αποτελεί τον θεμέλιο λίθο της λειτουργίας της, επιτρέποντας την ασφαλή διαχείριση των δεδομένων και την ομαλή αλληλεπίδραση μεταξύ του χρήστη και της εφαρμογής. Για την υλοποίηση του back-end, επιλέχθηκε η χρήση του Firebase της Google, το οποίο προσφέρει μια ολοκληρωμένη πλατφόρμα ανάπτυξης εφαρμογών, ιδιαίτερα ευέλικτη και εύκολη στη χρήση για εφαρμογές.

Το Firebase παρέχει ένα ευρύ φάσμα υπηρεσιών, όπως αυθεντικοποίηση χρηστών, αποθήκευση δεδομένων, real-time ενημερώσεις και δυνατότητα επέκτασης της εφαρμογής χωρίς περιορισμούς. Κύριος στόχος ήταν η δημιουργία ενός συστήματος που να επιτρέπει την αποθήκευση και επεξεργασία δεδομένων με ασφάλεια, ενώ παράλληλα να διασφαλίζεται η ταχύτητα και η αξιοπιστία της εφαρμογής.



Εικόνα 4.1.1 Πίνακας εγγεγραμμένων χρηστών μέσα στο Firebase Authentication της υπηρεσίας της εφαρμογής

Το παρακάτω screenshot παρουσιάζει τη διαχειριστική οθόνη χρηστών της υπηρεσίας Firebase Authentication, όπου απεικονίζονται τα στοιχεία όλων των εγγεγραμμένων χρηστών της εφαρμογής.

Στοιχεία και Πεδία της Διαχείρισης Χρηστών

Στην καρτέλα "Users" της Firebase Authentication Console, εμφανίζεται ένας πίνακας με τα εξής σημαντικά στοιχεία για κάθε χρήστη:

- **Identifier (Αναγνωριστικό):** Το email ή το τηλέφωνο που χρησιμοποίησε ο χρήστης για την εγγραφή του. Το αναγνωριστικό αυτό είναι απαραίτητο για την ταυτοποίηση και τη σύνδεση του χρήστη στην εφαρμογή.
- **Providers (Πάροχοι):** Ο πάροχος μέσω του οποίου έγινε η εγγραφή ή η σύνδεση του χρήστη. Στην εφαρμογή "RESERVE-EAT", οι περισσότεροι χρήστες έχουν επιλέξει την εγγραφή μέσω email και κωδικού πρόσβασης, ενώ υπάρχει και η δυνατότητα σύνδεσης μέσω του λογαριασμού Google.
- **Created (Ημερομηνία Δημιουργίας):** Η ημερομηνία κατά την οποία ο χρήστης δημιούργησε τον λογαριασμό του στην εφαρμογή. Αυτή η πληροφορία βοηθά στην ανάλυση της αύξησης των χρηστών με την πάροδο του χρόνου.
- **Signed In (Ημερομηνία Τελευταίας Σύνδεσης):** Η τελευταία φορά που ο χρήστης συνδέθηκε στην εφαρμογή. Η παρακολούθηση αυτής της πληροφορίας επιτρέπει την κατανόηση της δραστηριότητας των χρηστών και την αξιολόγηση της εμπειρίας χρήστη.
- **User UID (Μοναδικό Αναγνωριστικό Χρήστη):** Ένας μοναδικός κωδικός UID (User ID) αποδίδεται αυτόματα σε κάθε χρήστη κατά την εγγραφή. Αυτό το αναγνωριστικό χρησιμοποιείται για την ταυτοποίηση του χρήστη εντός της βάσης δεδομένων και για τη συσχέτιση των στοιχείων του με άλλες λειτουργίες της εφαρμογής, όπως οι κρατήσεις και τα αγαπημένα του εστιατόρια.

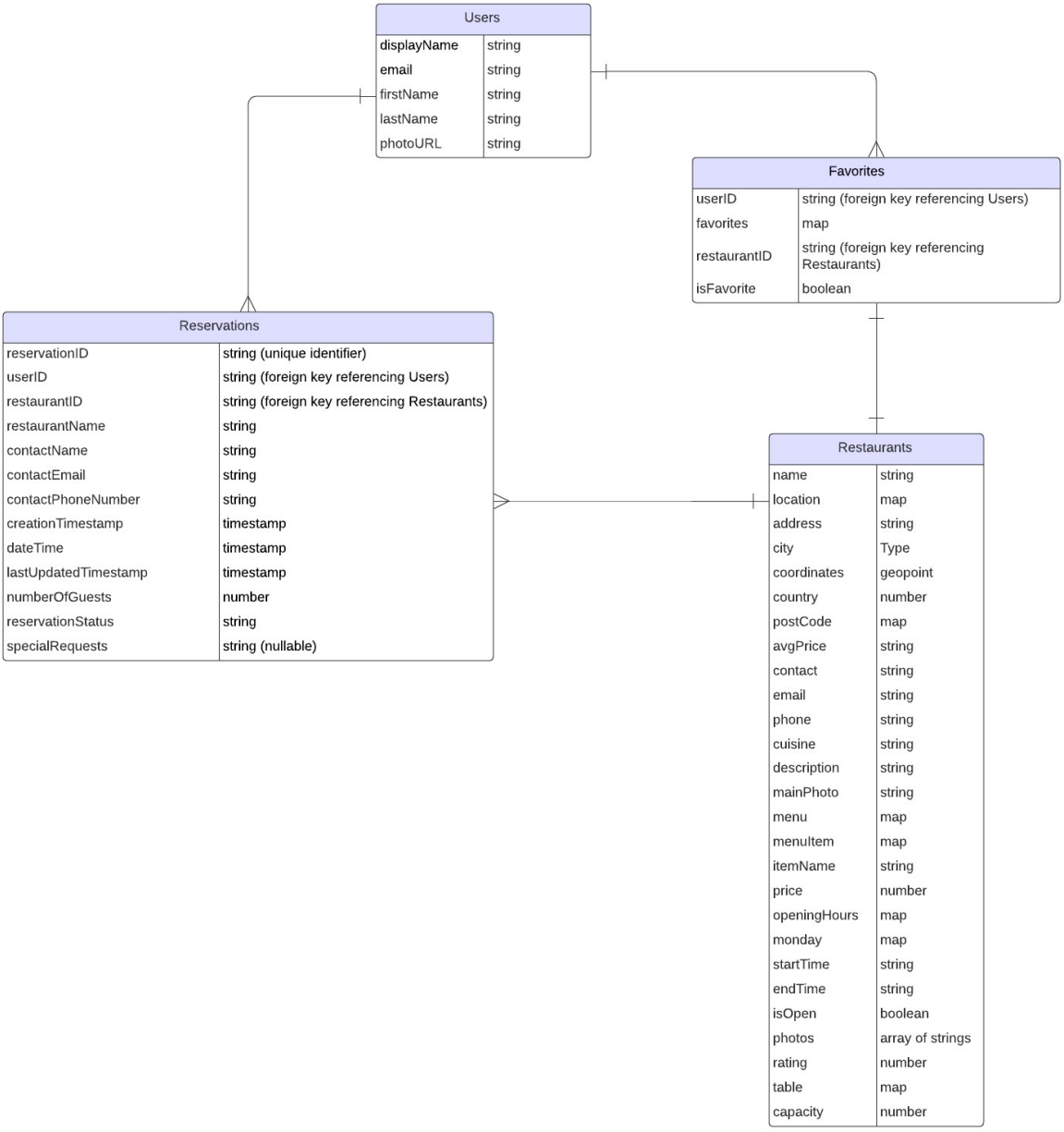
Η χρήση του Firebase Authentication στην "RESERVE-EAT" διασφαλίζει την ασφάλεια των δεδομένων των χρηστών, ενώ παράλληλα διευκολύνει τη διαχείρισή τους από τους διαχειριστές της εφαρμογής. Μέσω αυτής της υπηρεσίας, οι διαχειριστές έχουν τη δυνατότητα να παρακολουθούν τη δραστηριότητα των χρηστών, να ελέγχουν πότε και πώς συνδέονται στην εφαρμογή, και να προσθέτουν νέους χρήστες όταν είναι απαραίτητο.

Αυτή η διεπαφή παρέχει επίσης τη δυνατότητα εισαγωγής χρηστών με χειροκίνητο τρόπο μέσω της επιλογής "Add user". Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για την προσθήκη ειδικών χρηστών ή τη δημιουργία λογαριασμών που δεν απαιτούν την τυπική διαδικασία εγγραφής.

4.1.1 ER Diagram

Η σχεδίαση της βάσης δεδομένων ξεκίνησε με τη δημιουργία του ER διαγράμματος (Entity-Relationship Diagram), το οποίο αποτελεί βασικό εργαλείο για τον προσδιορισμό των οντοτήτων και των σχέσεων μεταξύ τους. Το ER διάγραμμα χρησιμοποιείται για να οπτικοποιήσει τη δομή της βάσης δεδομένων και να καθορίσει τις αλληλεπιδράσεις μεταξύ των διαφόρων στοιχείων.

4.1.2 Σχεδίαση Βάση Δεδομένων - Προσδιορισμός Οντοτήτων



Εικόνα 4.1.2 ER Διάγραμμα της εφαρμογής RESERVE-EAT

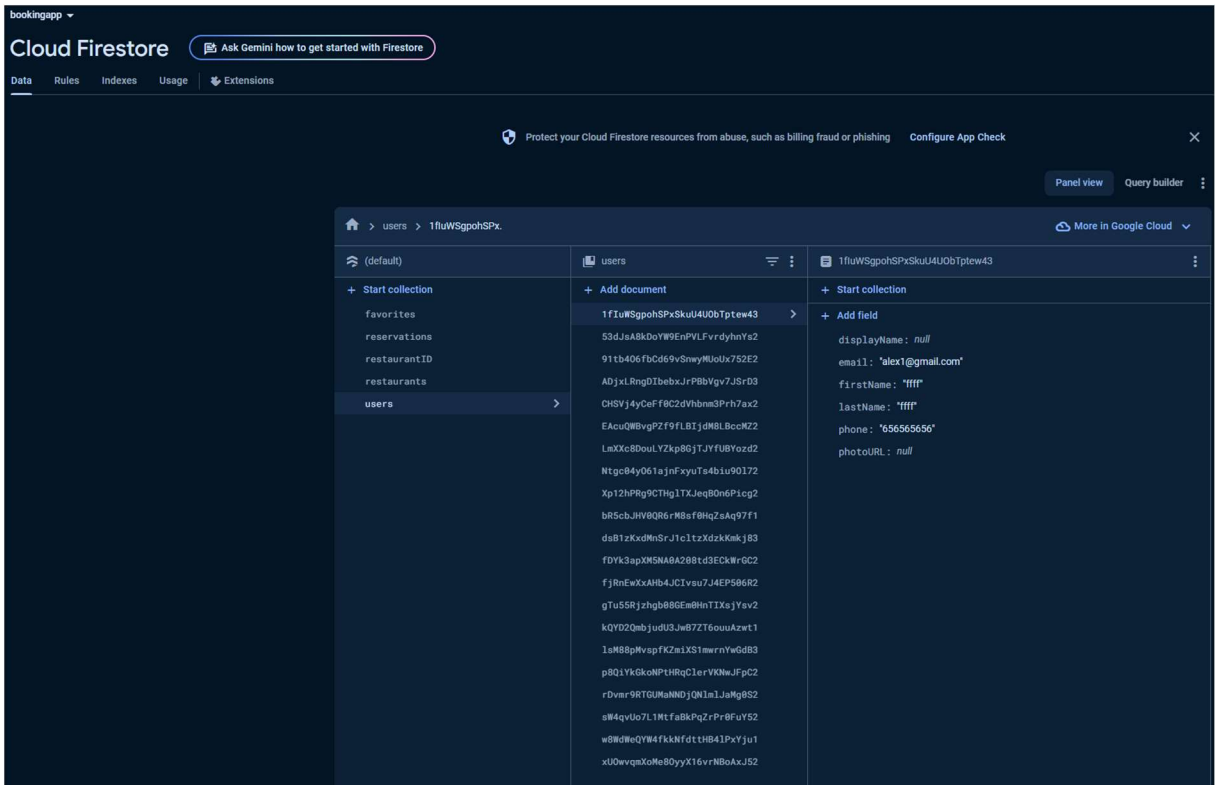
Το παραπάνω ER διάγραμμα (Entity-Relationship Diagram) αποτυπώνει την αρχιτεκτονική της βάσης δεδομένων που υποστηρίζει την εφαρμογή "RESERVE-EAT". Το διάγραμμα αυτό αποτελεί θεμελιώδες εργαλείο για την κατανόηση των οντοτήτων και των σχέσεων που υπάρχουν εντός της βάσης δεδομένων.

Περιγραφή των Κύριων Οντοτήτων και των Σχέσεών τους

1. **Users (Χρήστες):**

- **Περιγραφή:** Η οντότητα "Users" περιέχει όλες τις απαραίτητες πληροφορίες σχετικά με τους χρήστες της εφαρμογής "RESERVE-EAT". Τα δεδομένα που αποθηκεύονται περιλαμβάνουν το όνομα χρήστη, το email, και τη φωτογραφία προφίλ, τα οποία είναι απαραίτητα για την ταυτοποίηση και την εξατομίκευση της εμπειρίας χρήστη.
- **Σχέσεις:** Κάθε χρήστης μπορεί να συνδέεται με πολλές κρατήσεις (Reservations) και αγαπημένα εστιατόρια (Favorites), επιτρέποντας την καταγραφή και ανάκτηση αυτών των δεδομένων ανά πάσα στιγμή.

2. **Reservations (Κρατήσεις):**
   - **Περιγραφή:** Η οντότητα "Reservations" καταγράφει τις κρατήσεις που πραγματοποιούν οι χρήστες στην εφαρμογή. Τα δεδομένα περιλαμβάνουν την ημερομηνία και ώρα της κράτησης, τον αριθμό των επισκεπτών και ειδικές αιτήσεις του χρήστη. Αυτή η οντότητα είναι κεντρικής σημασίας για τη λειτουργία της εφαρμογής, καθώς επιτρέπει στους χρήστες να οργανώνουν τις κρατήσεις τους και να διαχειρίζονται τα γεύματά τους.
   - **Σχέσεις:** Κάθε κράτηση συνδέεται άμεσα με έναν χρήστη και ένα εστιατόριο. Αυτή η σύνδεση επιτρέπει την εύκολη ανάκτηση των σχετικών πληροφοριών, όπως το όνομα του εστιατορίου και οι λεπτομέρειες της κράτησης.
3. **Restaurants (Εστιατόρια):**
   - **Περιγραφή:** Η οντότητα "Restaurants" περιλαμβάνει λεπτομερείς πληροφορίες για τα εστιατόρια που συμμετέχουν στην πλατφόρμα "RESERVE-EAT". Τα δεδομένα περιλαμβάνουν το όνομα, τη διεύθυνση, την πόλη, την κουζίνα, το μενού και την χωρητικότητα του εστιατορίου. Αυτά τα στοιχεία είναι απαραίτητα για την παροχή πληροφοριών στους χρήστες και για τη διευκόλυνση της διαδικασίας κράτησης.
   - **Σχέσεις:** Ένα εστιατόριο μπορεί να συνδέεται με πολλές κρατήσεις και μπορεί να περιλαμβάνεται στα αγαπημένα των χρηστών, καθιστώντας το εύκολα προσβάσιμο από αυτούς.
4. **Favorites (Αγαπημένα):**
   - **Περιγραφή:** Η οντότητα "Favorites" επιτρέπει στους χρήστες να προσθέτουν εστιατόρια στα αγαπημένα τους, διευκολύνοντας την πρόσβαση σε αυτά σε μελλοντικές επισκέψεις. Αυτή η λειτουργία αυξάνει την αλληλεπίδραση των χρηστών με την εφαρμογή και βελτιώνει την εμπειρία χρήστη.
   - **Σχέσεις:** Κάθε εγγραφή στα αγαπημένα συνδέεται με έναν χρήστη και ένα εστιατόριο, επιτρέποντας την προσωποποιημένη προβολή και διαχείριση των προτιμήσεων των χρηστών.



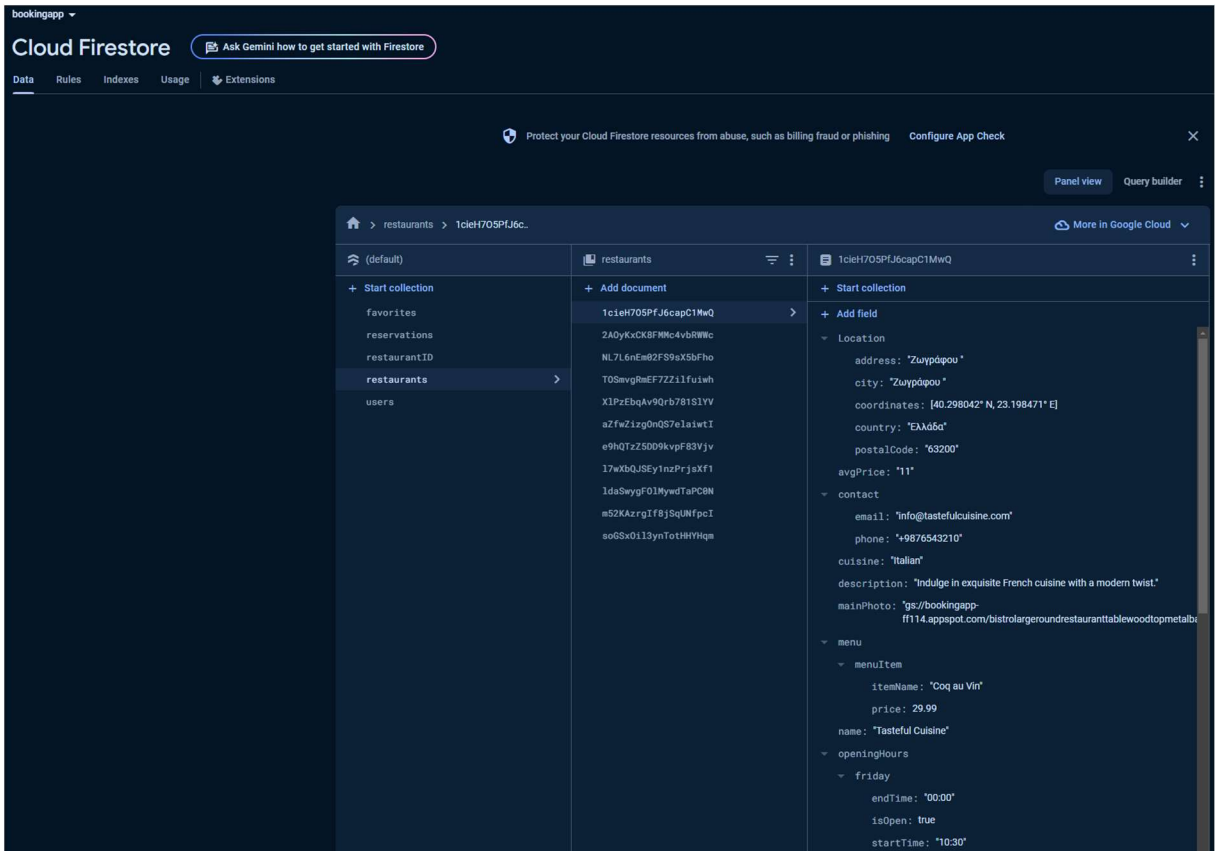
Εικόνα 4.1.3 Η συλλογή των χρηστών μέσα στην Firestore βάση, με επιλεγμένο τον πρώτο χρήστη.

Η αποθήκευση και διαχείριση των δεδομένων χρηστών στην εφαρμογή οργανώνεται μέσω του Firestore, με κάθε χρήστη να εκπροσωπείται από ένα μοναδικό αναγνωριστικό Document ID στη συλλογή "Users". Αυτό το ID εξασφαλίζει την μοναδικότητα του χρήστη και του επιτρέπει να συνδέεται με άλλες σημαντικές πληροφορίες, όπως οι κρατήσεις του και τα αγαπημένα του εστιατόρια.

Στο Firestore, η συλλογή "Users" περιέχει τα έγγραφα που αντιπροσωπεύουν κάθε χρήστη. Η οθόνη διαχείρισης επιτρέπει την προβολή αυτών των εγγράφων, όπου επιλέγοντας το Document ID ενός χρήστη, εμφανίζονται όλες οι σχετικές πληροφορίες του. Τα δεδομένα του χρήστη χωρίζονται σε δύο βασικά τμήματα: τα βασικά στοιχεία του και οι εσωτερικές συλλογές που σχετίζονται με αυτόν.

Στα βασικά στοιχεία του χρήστη περιλαμβάνονται το όνομα (displayName), η διεύθυνση email (email), το μικρό όνομα (firstName), το επώνυμο (lastName), ο αριθμός τηλεφώνου (phone) και το URL της φωτογραφίας προφίλ (photoURL). Αυτά τα πεδία βρίσκονται στο πρώτο επίπεδο της βάσης δεδομένων και είναι άμεσα διαθέσιμα κατά την ανάκτηση των δεδομένων του χρήστη.

Οι εσωτερικές συλλογές, όπως "reservations" και "favorites", αποθηκεύονται σε βαθύτερα επίπεδα και περιέχουν πληροφορίες που σχετίζονται με τις δραστηριότητες του χρήστη. Για παράδειγμα, η συλλογή "reservations" καταγράφει όλες τις κρατήσεις που έχει κάνει ο χρήστης, με λεπτομέρειες όπως η ημερομηνία, η ώρα, το όνομα του εστιατορίου και ο αριθμός των επισκεπτών. Αυτές οι πληροφορίες επιτρέπουν την παρακολούθηση των κρατήσεων του χρήστη και την παροχή εξειδικευμένων υπηρεσιών, όπως υπενθυμίσεις κρατήσεων ή προσφορές εστιατορίων.



Εικόνα 4.1.4 Σχήμα 4.4. Αποθήκευση Δεδομένων Εστιατορίων στο Firestore

Η συλλογή "restaurants" αποθηκεύει πληροφορίες σχετικά με τα εστιατόρια που είναι καταχωρημένα στην εφαρμογή. Περιλαμβάνει δεδομένα όπως το όνομα του εστιατορίου, τη διεύθυνση, την πόλη, τον τύπο της κουζίνας και την περιγραφή του εστιατορίου. Αυτά τα στοιχεία παρέχουν στους χρήστες τη δυνατότητα να αναζητήσουν και να επιλέξουν το κατάλληλο εστιατόριο σύμφωνα με τις προτιμήσεις τους.

Η σχέση μεταξύ των οντοτήτων "Users", "Reservations", και "Restaurants" διασφαλίζει ότι τα δεδομένα είναι καλά οργανωμένα και εύκολα προσβάσιμα. Οι χρήστες μπορούν να κάνουν κρατήσεις, να προσθέσουν εστιατόρια στα αγαπημένα τους, και να διαχειρίζονται αυτές τις πληροφορίες με απλό και αποτελεσματικό τρόπο. Η ευελιξία που προσφέρει το Firestore στην αποθήκευση και διαχείριση των δεδομένων διασφαλίζει την κλιμακωτή ανάπτυξη της εφαρμογής, επιτρέποντας την προσθήκη νέων λειτουργιών και την υποστήριξη αυξανόμενου αριθμού χρηστών.

Η αποδοτική χρήση των εσωτερικών συλλογών βοηθά στην οργάνωση των δεδομένων σε επίπεδα, κάτι που βελτιώνει την απόδοση και μειώνει τον φόρτο των αιτημάτων στον διακομιστή. Οι χρήστες λαμβάνουν γρήγορα τις απαραίτητες πληροφορίες για τις κρατήσεις τους και τα αγαπημένα τους εστιατόρια, ενώ η εφαρμογή διαχειρίζεται αποτελεσματικά τα δεδομένα.

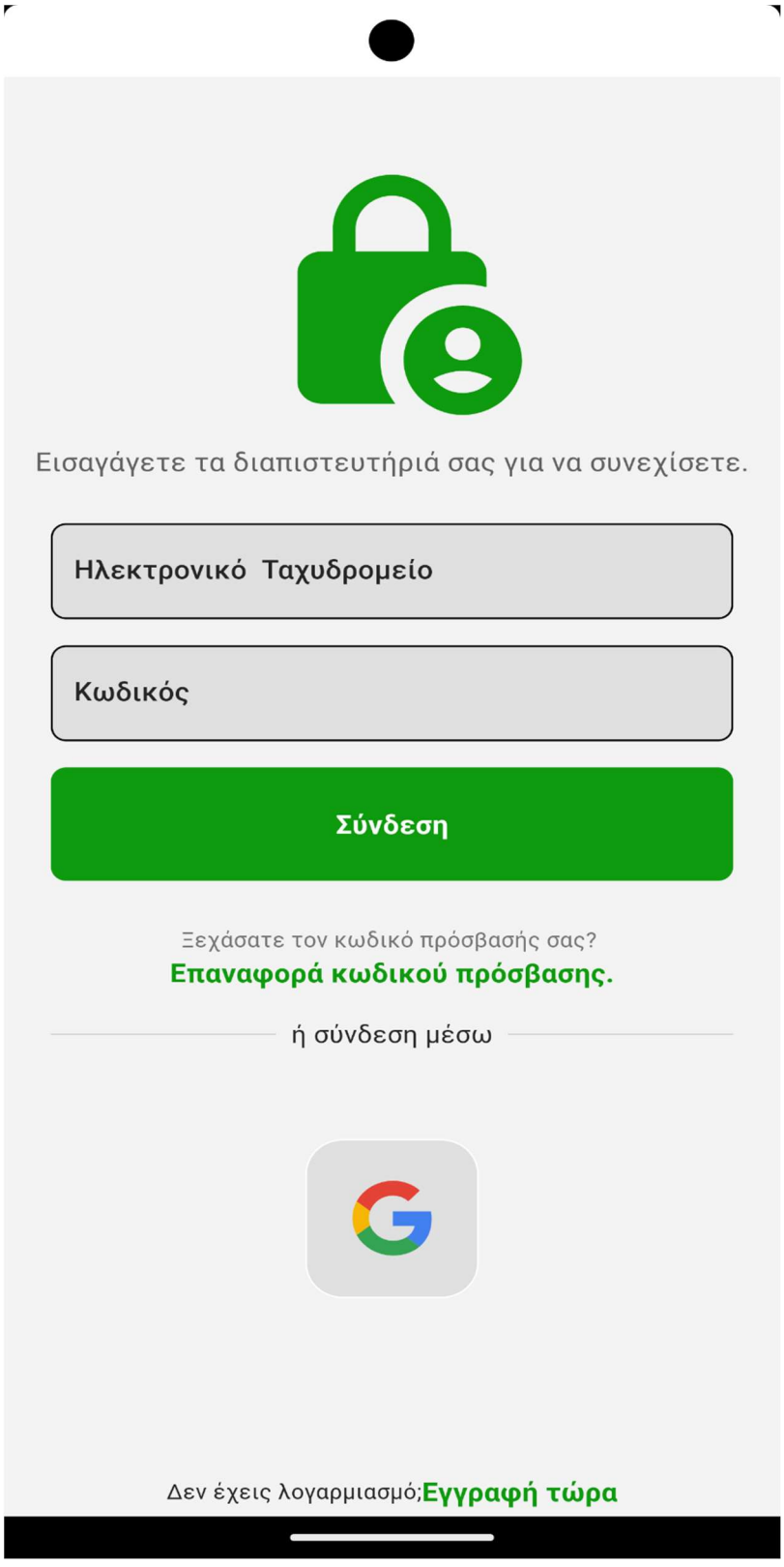
4.2 ΟΘΟΝΕΣ ΔΙΕΠΑΦΗΣ ΧΡΗΣΤΩΝ ΚΑΙ ΠΕΡΙΓΡΑΦΗ ΠΕΡΙΠΤΩΣΕΩΝ ΧΡΗΣΗΣ ΤΗΣ MOBILE ΕΚΔΟΣΗΣ

Αυτή η οθόνη είναι η πρώτη που βλέπει ο χρήστης όταν ανοίγει την εφαρμογή. Η οθόνη έναρξης (Splash Screen) χρησιμεύει για να δημιουργήσει μια ευχάριστη πρώτη εντύπωση και να παρουσιάσει το λογότυπο της εφαρμογής RESERVE-EAT, ενισχύοντας έτσι την ταυτότητα του brand. Επιπλέον, δίνει

στην εφαρμογή τον απαραίτητο χρόνο για να φορτώσει και να ετοιμαστεί πριν την εμφάνιση της κύριας λειτουργικότητας. Σε αυτό το στάδιο, ο χρήστης δεν αλληλεπιδρά με την εφαρμογή, αλλά προετοιμάζεται για την εμπειρία που ακολουθεί.

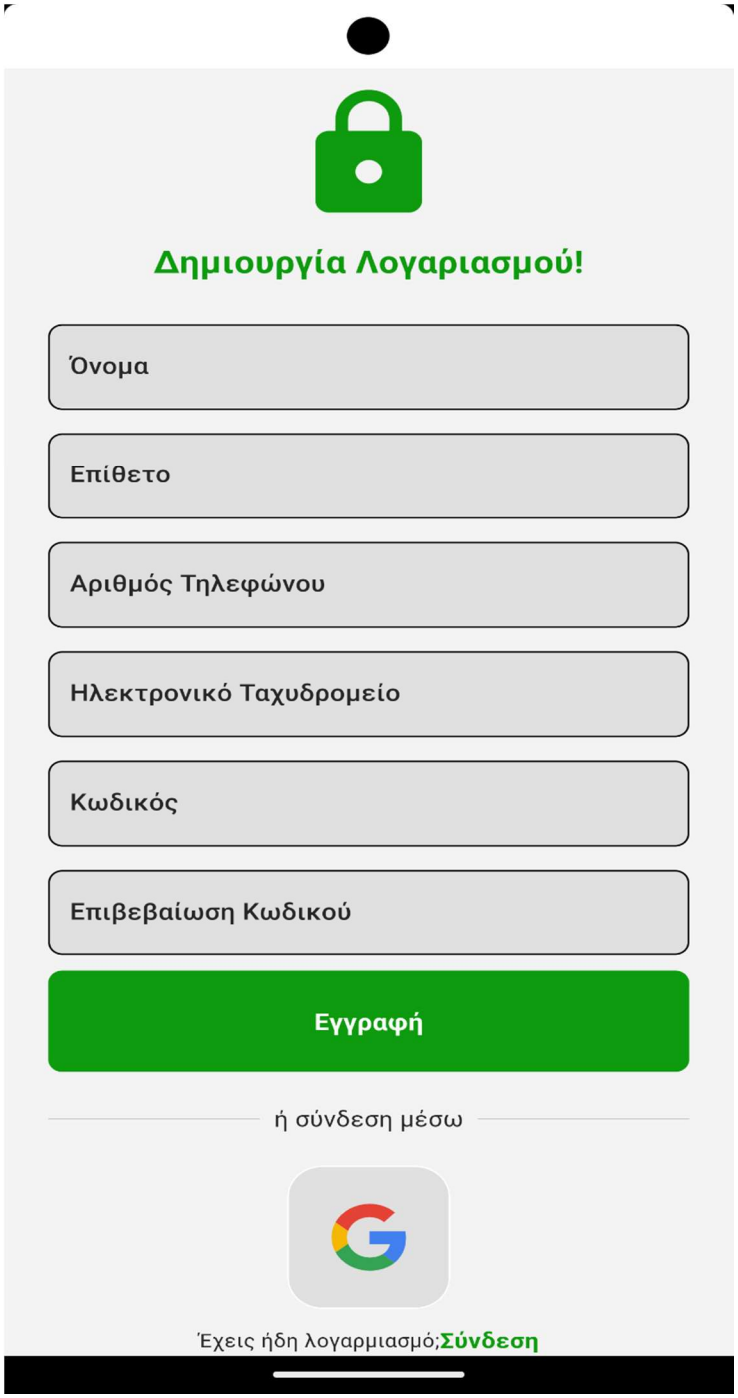
Εικόνα 4.2.1 Οθόνη Έναρξης (Splash Screen)

Αμέσως μετά είναι το σημείο εισόδου του χρήστη στην εφαρμογή. Προσφέρει δύο βασικές επιλογές σύνδεσης: μέσω ηλεκτρονικού ταχυδρομείου και κωδικού πρόσβασης ή μέσω του λογαριασμού Google. Η οθόνη περιλαμβάνει επίσης τη δυνατότητα επαναφοράς του κωδικού πρόσβασης σε περίπτωση που ο χρήστης τον έχει ξεχάσει, καθώς και την επιλογή εγγραφής για νέους χρήστες που δεν έχουν ακόμα λογαριασμό. Η παρουσία αυτής της οθόνης εξασφαλίζει την ασφαλή πρόσβαση του χρήστη στην εφαρμογή και προστατεύει τα δεδομένα του.



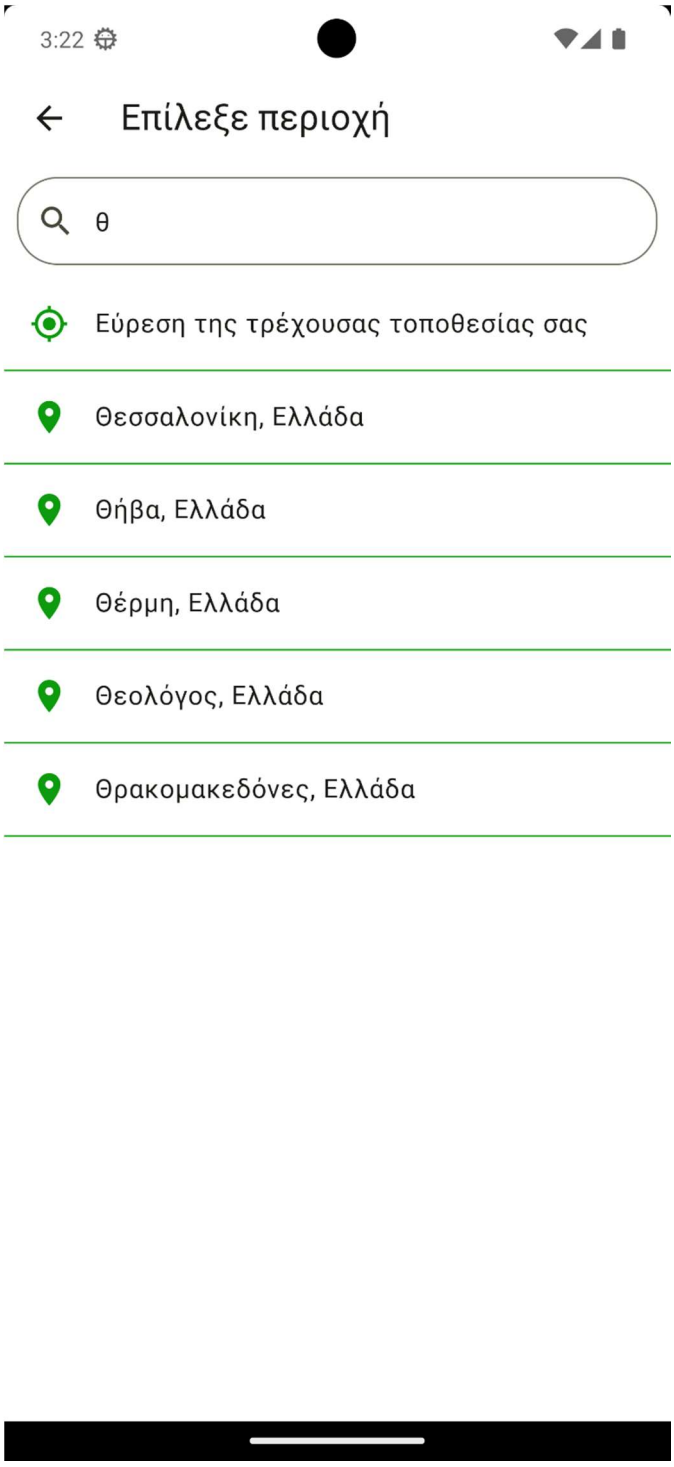
Εικόνα 4.2.2 Οθόνη Σύνδεσης (Login Screen)

Η οθόνη εγγραφής επιτρέπει στους νέους χρήστες να δημιουργήσουν έναν λογαριασμό στην εφαρμογή. Οι χρήστες καλούνται να εισάγουν τα προσωπικά τους στοιχεία, όπως όνομα, επίθετο, αριθμό τηλεφώνου, ηλεκτρονικό ταχυδρομείο και να δημιουργήσουν έναν ασφαλή κωδικό πρόσβασης, ο οποίος επιβεβαιώνεται για αποφυγή λαθών. Η οθόνη περιλαμβάνει επίσης την επιλογή εγγραφής μέσω του λογαριασμού Google, επιτρέποντας στους χρήστες να ολοκληρώσουν τη διαδικασία με ένα μόνο κλικ. Στο κάτω μέρος, υπάρχει η επιλογή για χρήστες που έχουν ήδη λογαριασμό να επιστρέψουν στην οθόνη σύνδεσης.



Εικόνα 4.2.3 Οθόνη Εγγραφής (Registration Screen)

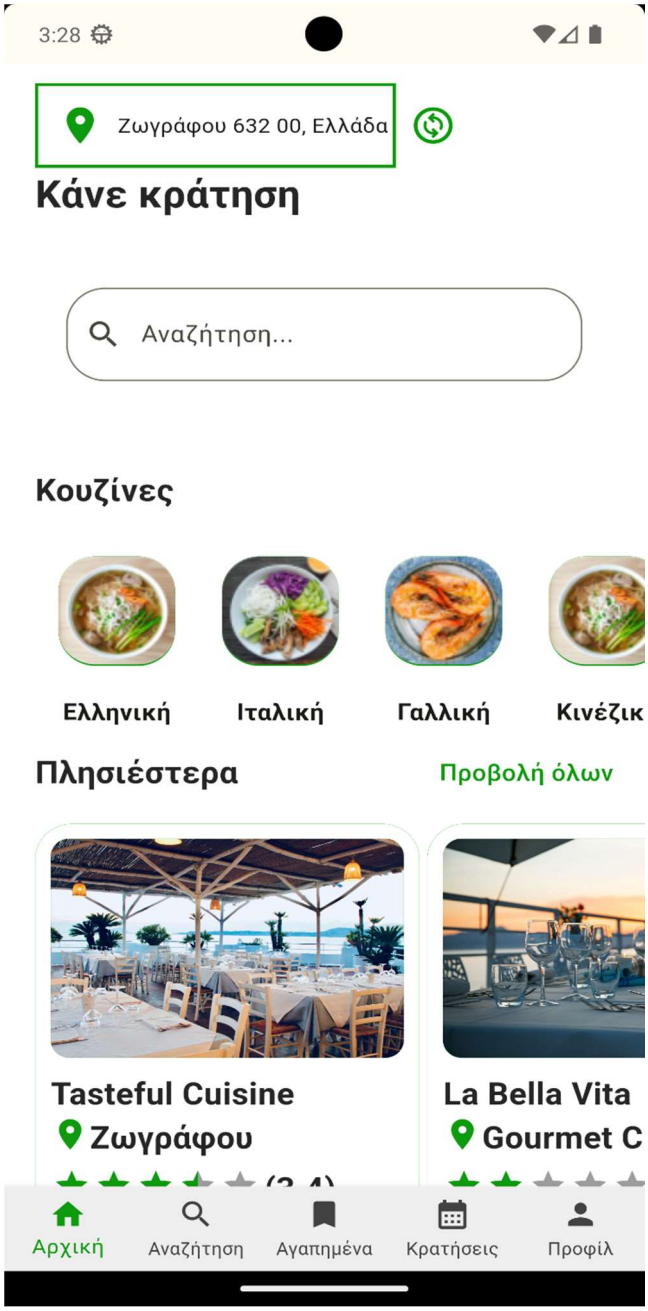
Παρακάτω βλέπουμε την οθόνη επιτρέπει στον χρήστη να επιλέξει την τοποθεσία του για να βρει εστιατόρια, καταστήματα ή υπηρεσίες στην κοντινή περιοχή. Η οθόνη παρέχει τη δυνατότητα εύρεσης της τρέχουσας τοποθεσίας του χρήστη μέσω GPS, καθώς και την επιλογή τοποθεσίας από μια λίστα προτεινόμενων περιοχών. Επίσης, υπάρχει μια λειτουργία αναζήτησης που επιτρέπει στον χρήστη να πληκτρολογήσει την επιθυμητή τοποθεσία. Η σωστή επιλογή τοποθεσίας είναι κρίσιμη για την εξατομίκευση των προτάσεων και των επιλογών που θα εμφανιστούν στη συνέχεια της εφαρμογής.



Εικόνα 4.2.4 Οθόνη Επιλογής Τοποθεσίας (Location Selection Screen)

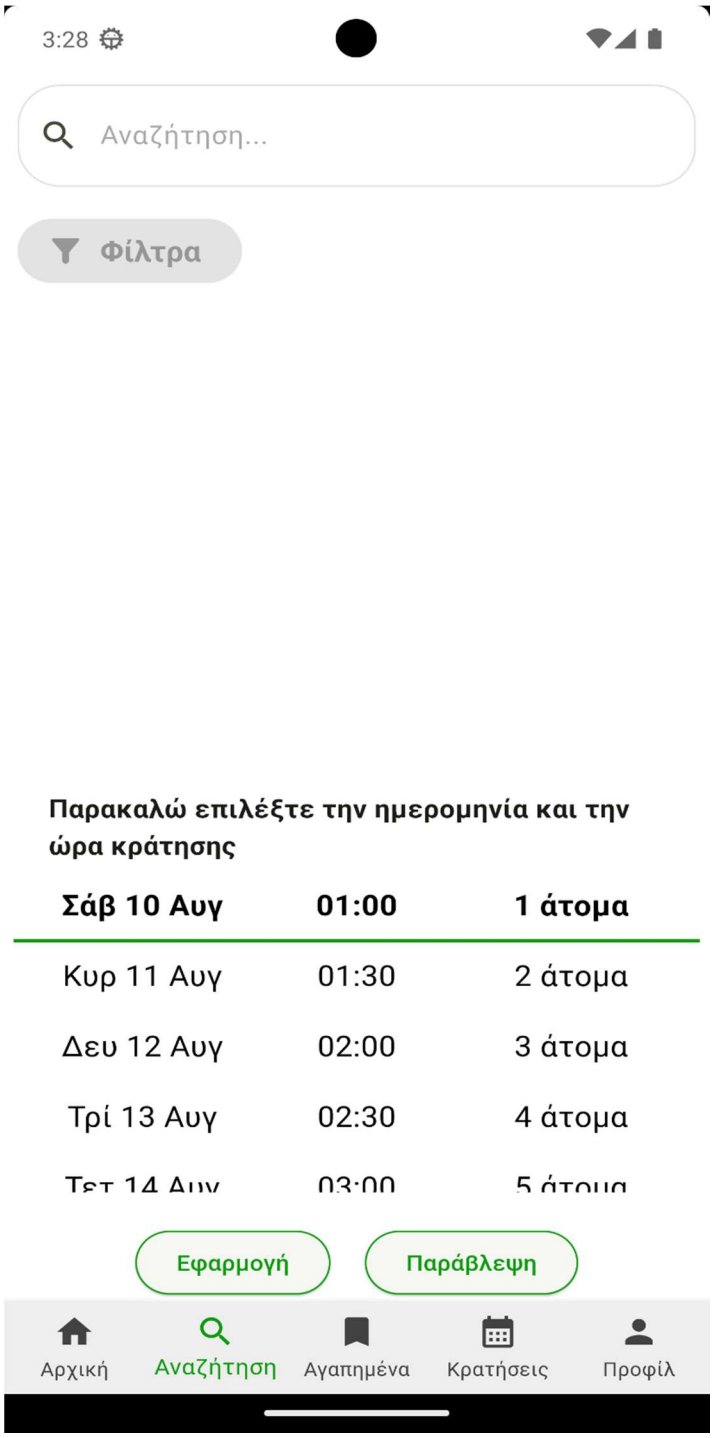
Η κεντρική οθόνη της εφαρμογής παρέχει στον χρήστη τη δυνατότητα να κάνει κράτηση σε εστιατόρια, ανάλογα με την τοποθεσία που έχει επιλέξει ή που έχει εντοπιστεί αυτόματα μέσω GPS. Στο επάνω μέρος της οθόνης, εμφανίζεται η επιλεγμένη τοποθεσία, την οποία ο χρήστης μπορεί να αλλάξει αν το επιθυμεί. Η αναζήτηση επιτρέπει στο χρήστη να βρει εστιατόρια συγκεκριμένης κουζίνας ή να περιηγηθεί σε διάφορες επιλογές, όπως ελληνική, ιταλική, γαλλική, κ.ά. Επιπλέον, η οθόνη παρέχει προτάσεις για τα πλησιέστερα εστιατόρια, με δυνατότητα προβολής όλων των επιλογών. Το κάτω μενού

επιτρέπει γρήγορη πρόσβαση σε άλλες βασικές λειτουργίες, όπως την αναζήτηση, τα αγαπημένα, τις κρατήσεις και το προφίλ του χρήστη. Αυτή η οθόνη είναι το σημείο εκκίνησης για την εμπειρία της εφαρμογής, προσφέροντας εύκολη πλοήγηση και άμεση πρόσβαση στις βασικές της λειτουργίες.



Εικόνα 4.2.5 Κεντρική Οθόνη Εφαρμογής (Home Screen)

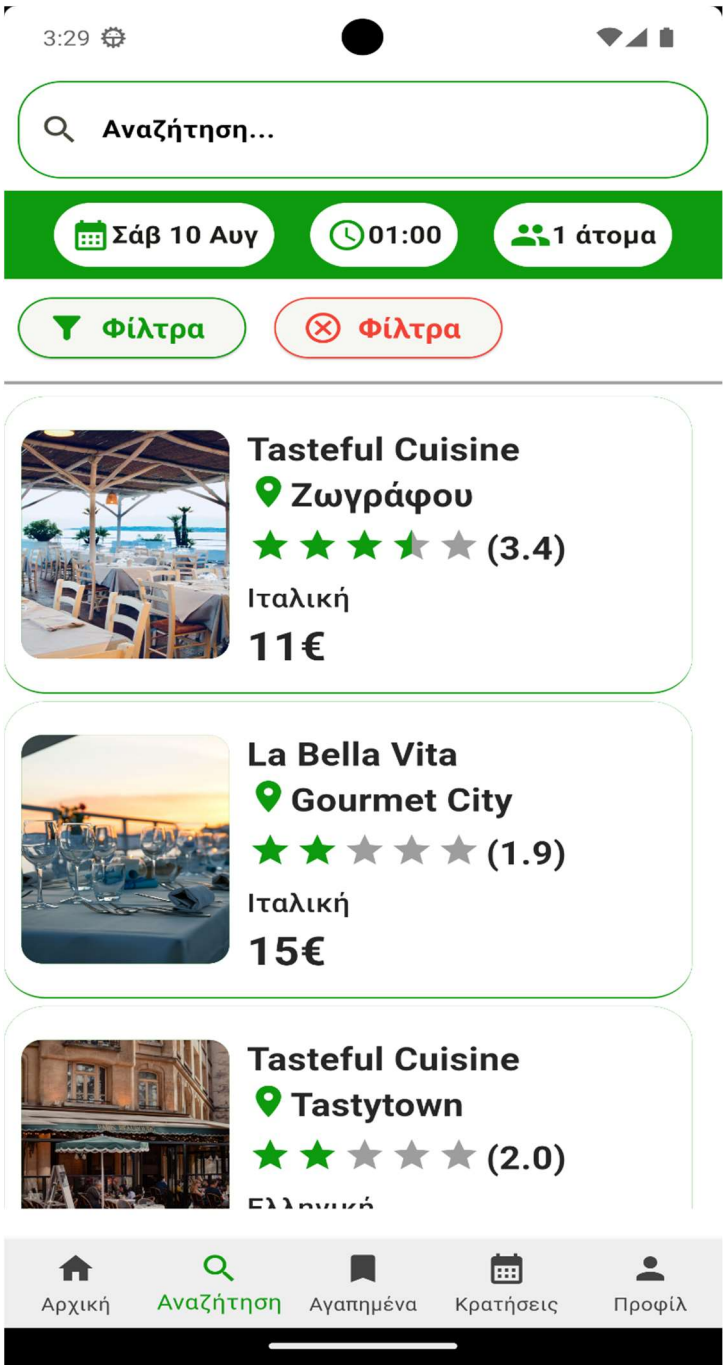
Επιπλέον, η οθόνη επιτρέπει στον χρήστη να επιλέξει την επιθυμητή ημερομηνία και ώρα για την κράτηση του σε κάποιο εστιατόριο ή άλλη υπηρεσία. Ο χρήστης μπορεί να δει διαθέσιμες επιλογές ημερομηνιών και ωρών, καθώς και να επιλέξει τον αριθμό ατόμων που θα συμμετέχουν. Μόλις επιλεγούν οι κατάλληλες πληροφορίες, ο χρήστης μπορεί να τις εφαρμόσει ή να παραβλέψει την επιλογή του, προχωρώντας στην επόμενη οθόνη. Αυτή η διαδικασία είναι σημαντική για να εξασφαλίσει ότι η κράτηση θα γίνει για την ακριβή ώρα και ημερομηνία που επιθυμεί ο χρήστης, διευκολύνοντας την εμπειρία του.



Εικόνα 4.2.6 Οθόνη Επιλογής Ημερομηνίας και Ώρας Κράτησης (Date and Time Picker Screen)

Η επόμενη οθόνη επιτρέπει στον χρήστη να αναζητήσει εστιατόρια ή άλλες υπηρεσίες με βάση τα κριτήρια που έχει επιλέξει προηγουμένως, όπως η ημερομηνία, η ώρα και ο αριθμός ατόμων για την κράτηση. Ο χρήστης μπορεί να προσθέσει επιπλέον φίλτρα για να περιορίσει τα αποτελέσματα, όπως την κουζίνα, την τιμή και την τοποθεσία. Τα αποτελέσματα της αναζήτησης εμφανίζονται με εικόνες, βαθμολογίες, είδος κουζίνας και τιμές, διευκολύνοντας τον χρήστη να κάνει τη σωστή επιλογή. Οι επιλογές φιλτραρίσματος και τα αποτελέσματα παρουσιάζονται με τρόπο που επιτρέπει γρήγορη και εύκολη σύγκριση μεταξύ των διαφόρων διαθέσιμων επιλογών. Ο χρήστης μπορεί να προχωρήσει με

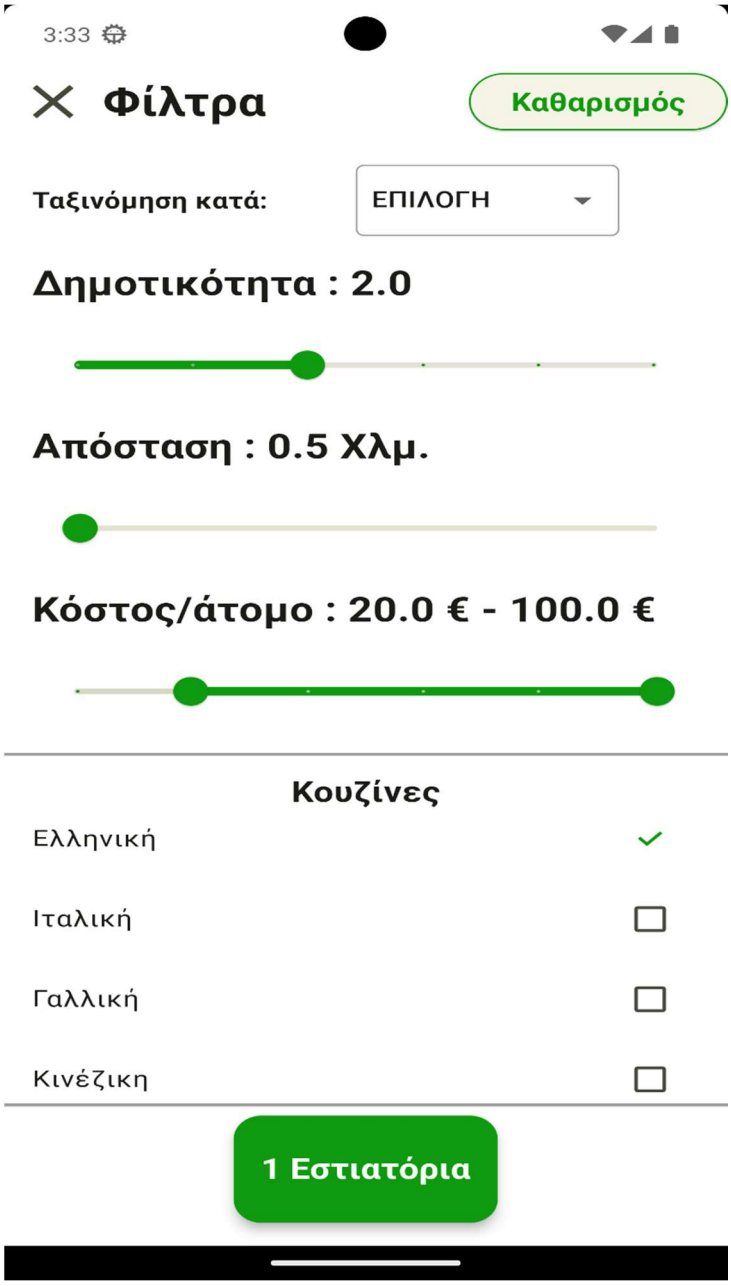
την επιλογή του κάνοντας κράτηση ή να επανεξετάσει τα φίλτρα και τα κριτήρια αναζήτησης για να βρει αυτό που ταιριάζει καλύτερα στις ανάγκες του.



Εικόνα 4.2.7 Οθόνη Αναζήτησης και Φιλτραρίσματος (Search and Filter Screen)

Αυτή η οθόνη δίνει τη δυνατότητα στον χρήστη να προσαρμόσει τα φίλτρα της αναζήτησής του για να βρει εστιατόρια ή άλλες υπηρεσίες που ανταποκρίνονται καλύτερα στις προτιμήσεις του. Οι επιλογές φιλτραρίσματος περιλαμβάνουν τη δυνατότητα ταξινόμησης με βάση τη δημοτικότητα, την απόσταση από την τοποθεσία του χρήστη και το κόστος ανά άτομο. Επιπλέον, ο χρήστης μπορεί να επιλέξει συγκεκριμένες κουζίνες, όπως ελληνική, ιταλική, γαλλική κ.ά. Η οθόνη παρέχει επίσης τη δυνατότητα καθαρισμού των φίλτρων για να ξεκινήσει μια νέα αναζήτηση χωρίς περιορισμούς. Το κουμπί στο κάτω

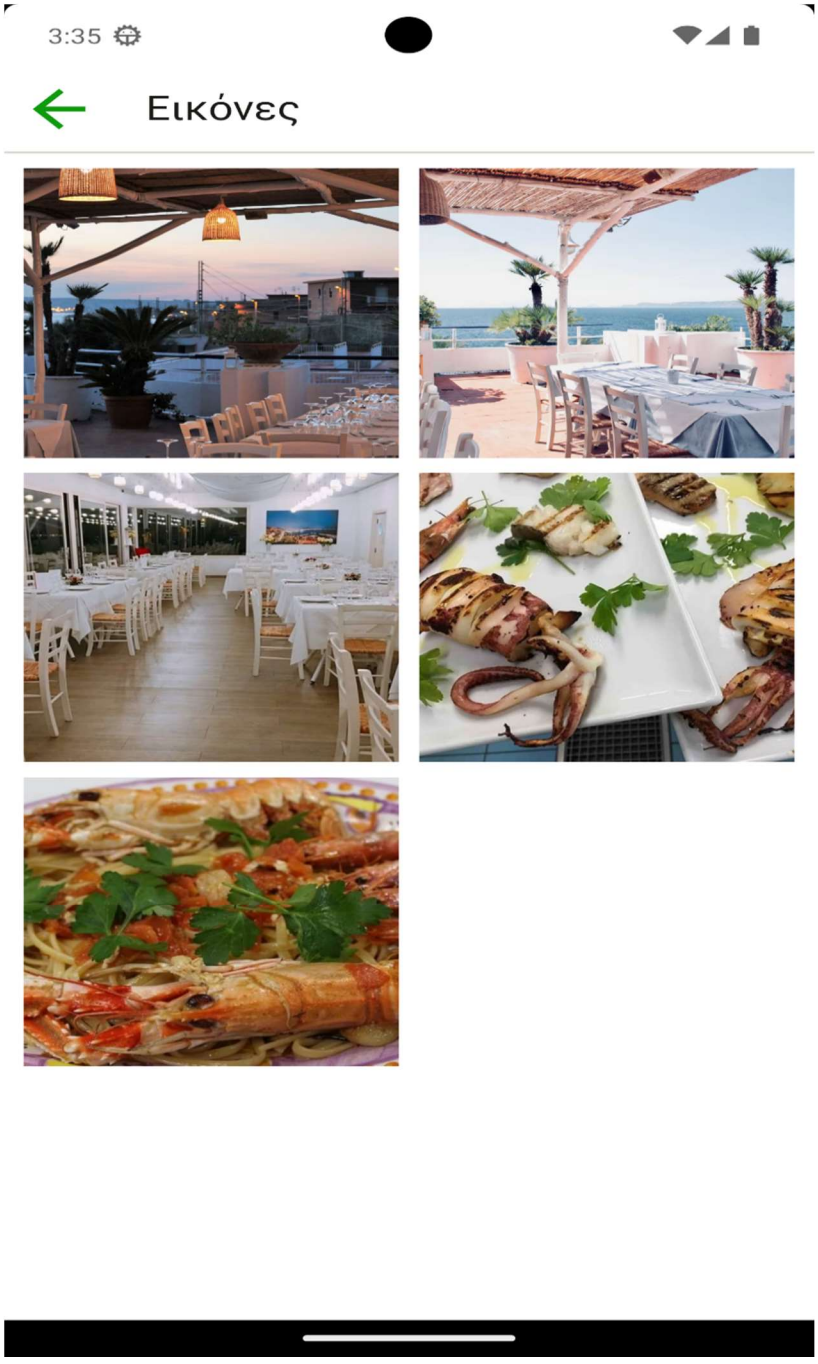
μέρος της οθόνης δείχνει τον αριθμό των εστιατορίων που ταιριάζουν με τα επιλεγμένα φίλτρα, επιτρέποντας στον χρήστη να προχωρήσει στην προβολή των αποτελεσμάτων ή να συνεχίσει να προσαρμόζει τα φίλτρα του. Αυτή η λειτουργία βοηθά τον χρήστη να εξειδικεύσει την αναζήτησή του και να βρει πιο εύκολα αυτό που αναζητά, εξοικονομώντας χρόνο και βελτιώνοντας την εμπειρία του.



Εικόνα 4.2.8 Οθόνη Ρυθμίσεων Φιλτραρίσματος (Filter Settings Screen)

Η αμέσως επόμενη οθόνη παρουσιάζει στον χρήστη μια συλλογή από φωτογραφίες του εστιατορίου που έχει επιλέξει. Οι εικόνες περιλαμβάνουν τόσο τον εσωτερικό όσο και τον εξωτερικό χώρο του εστιατορίου, καθώς και πιάτα από το μενού. Αυτή η δυνατότητα επιτρέπει στον χρήστη να πάρει μια σαφή εικόνα της ατμόσφαιρας και της ποιότητας του εστιατορίου πριν αποφασίσει να κάνει κράτηση. Οι φωτογραφίες αποτελούν σημαντικό στοιχείο της εμπειρίας, καθώς παρέχουν οπτικά ερεθίσματα που μπορούν να επηρεάσουν την απόφαση του χρήστη. Ο χρήστης μπορεί να επιστρέψει εύκολα στην

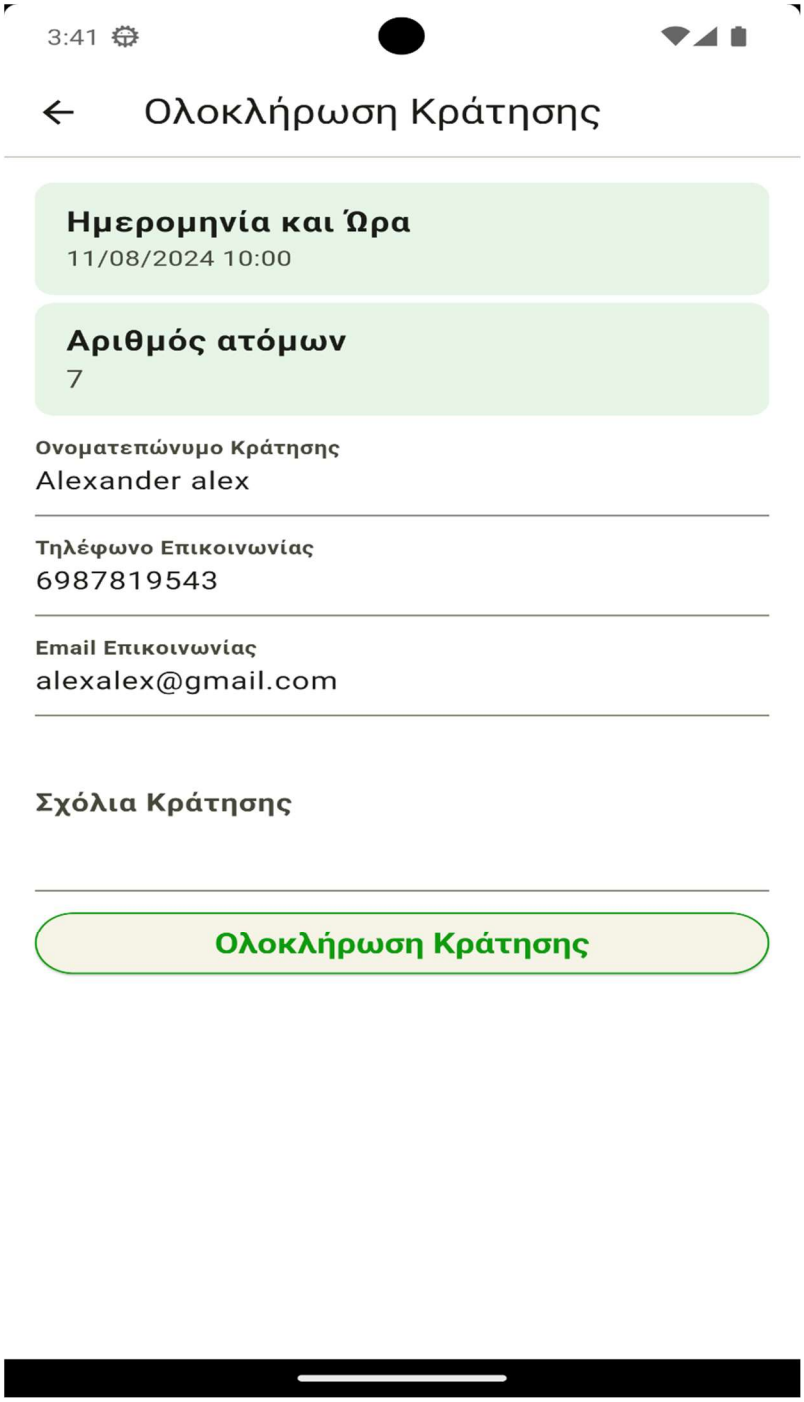
προηγούμενη οθόνη χρησιμοποιώντας το κουμπί πίσω στην κορυφή της οθόνης. Αυτή η οπτική περιήγηση βοηθά στη δημιουργία προσδοκιών και στην ενίσχυση της εμπιστοσύνης στην επιλογή του εστιατορίου.



Εικόνα 4.2.9 Οθόνη Προβολής Φωτογραφιών Εστιατορίου (Restaurant Photos View Screen)

Παρακάτω εμφανίζεται στον χρήστη ως τελευταίο βήμα για την ολοκλήρωση της κράτησης σε κάποιο εστιατόριο. Περιλαμβάνει όλες τις απαραίτητες λεπτομέρειες για την κράτηση, όπως την ημερομηνία και ώρα, τον αριθμό ατόμων, καθώς και τα στοιχεία επικοινωνίας του χρήστη (όνομα, τηλέφωνο, email). Ο χρήστης έχει επίσης τη δυνατότητα να προσθέσει σχόλια σχετικά με την κράτηση, για παράδειγμα, ειδικές απαιτήσεις ή προτιμήσεις. Με το πάτημα του κουμπιού "Ολοκλήρωση Κράτησης", η κράτηση

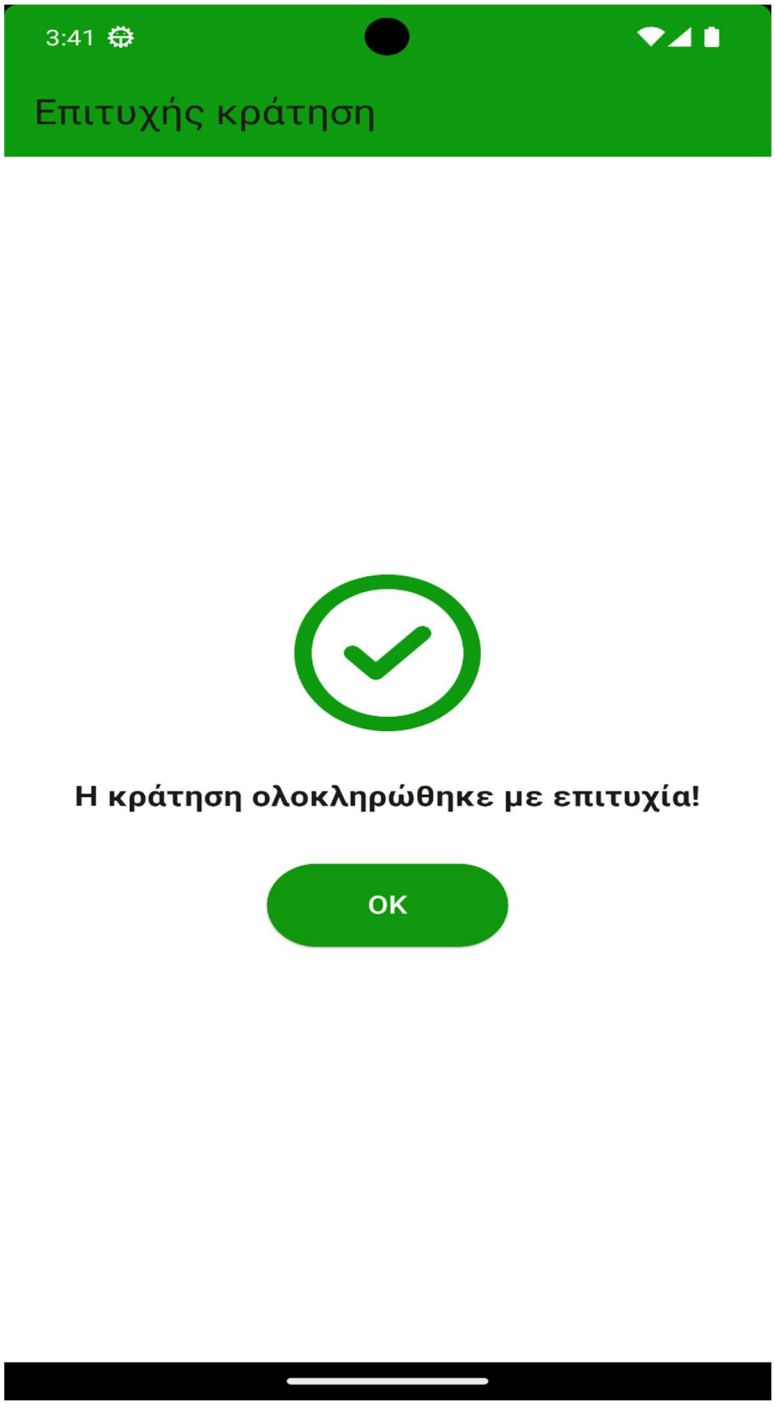
ολοκληρώνεται και ο χρήστης λαμβάνει μια επιβεβαίωση. Αυτή η οθόνη εξασφαλίζει ότι όλες οι πληροφορίες είναι σωστές πριν την τελική επιβεβαίωση, παρέχοντας μια ξεκάθαρη και εύκολη διαδικασία για τον χρήστη.



Εικόνα 4.2.10 Οθόνη Ολοκλήρωσης Κράτησης (Reservation Completion Screen)

Εδώ βλέπουμε να εμφανίζεται η οθόνη αμέσως μετά την ολοκλήρωση της κράτησης από τον χρήστη, επιβεβαιώνοντας ότι η διαδικασία πραγματοποιήθηκε με επιτυχία. Το απλό και σαφές μήνυμα «Η κράτηση ολοκληρώθηκε με επιτυχία!» παρέχει στον χρήστη την αναγκαία ανατροφοδότηση ότι η κράτησή του είναι πλέον καταχωρημένη. Το κουμπί "OK" επιτρέπει στον χρήστη να επιστρέψει στην

αρχική οθόνη ή σε κάποιο άλλο μέρος της εφαρμογής για να συνεχίσει την πλοήγηση ή να πραγματοποιήσει άλλη ενέργεια. Αυτή η οθόνη προσφέρει έναν καθαρό και αποτελεσματικό τρόπο για την ολοκλήρωση της εμπειρίας κράτησης, διασφαλίζοντας ότι ο χρήστης γνωρίζει ότι όλα πήγαν όπως τα είχε σχεδιάσει.



Εικόνα 4.2.11 Οθόνη Επιτυχούς Ολοκλήρωσης Κράτησης (Successful Reservation Completion Screen)

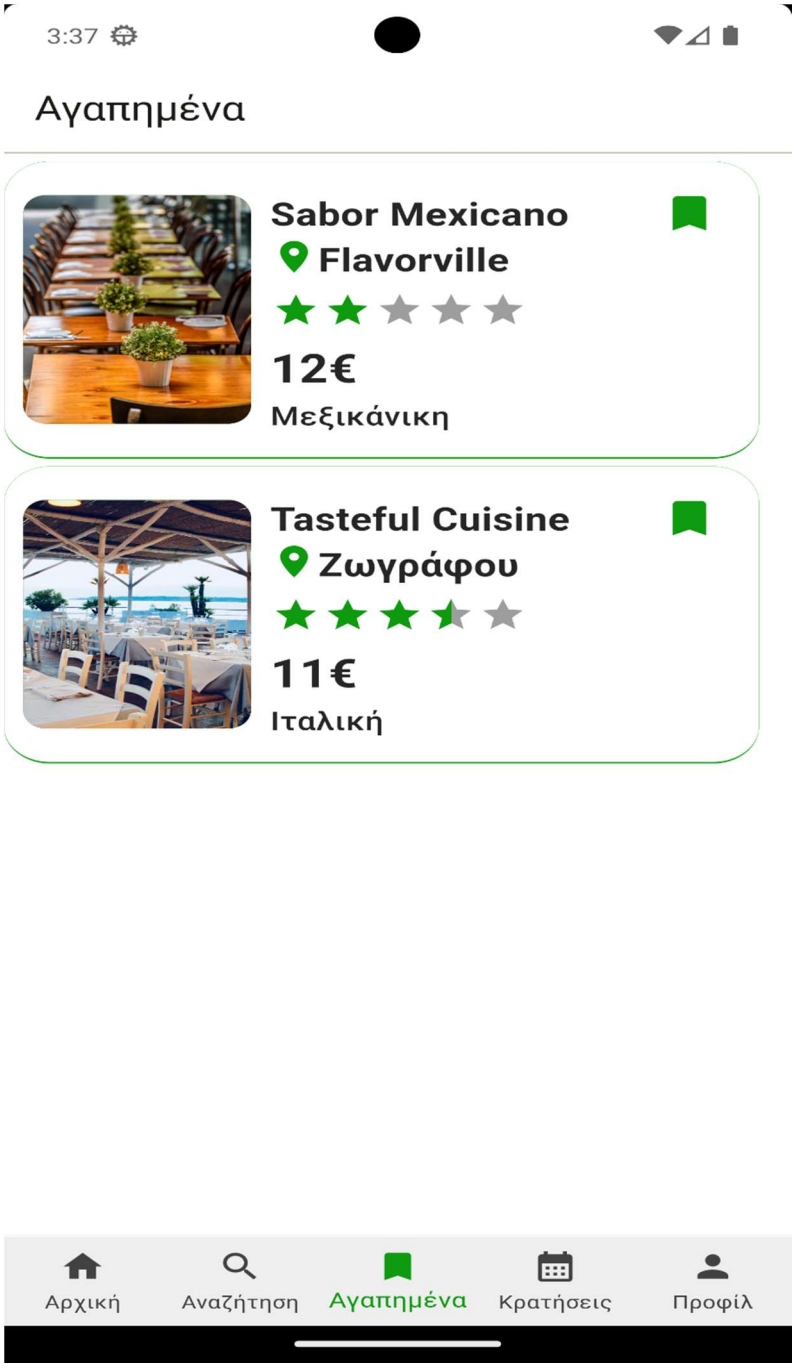
Η παρακάτω οθόνη παρέχει μια λεπτομερή παρουσίαση του εστιατορίου που έχει επιλέξει ο χρήστης. Στην κορυφή της οθόνης, εμφανίζεται μια μεγάλη εικόνα του εστιατορίου, με ένα κουμπί για την προβολή περισσότερων φωτογραφιών. Ακολουθούν βασικές πληροφορίες όπως το όνομα του

εστιατορίου, μια σύντομη περιγραφή της κουζίνας και της εμπειρίας που προσφέρει, η διεύθυνση, οι τιμές ορισμένων πιάτων από το μενού, καθώς και τα στοιχεία επικοινωνίας για τηλεφωνικές κρατήσεις. Η οθόνη περιλαμβάνει επίσης το ωράριο λειτουργίας του εστιατορίου, ώστε ο χρήστης να μπορεί να επιλέξει την κατάλληλη ώρα για την κράτησή του. Το κουμπί "Κάνε κράτηση" στο κάτω μέρος της οθόνης δίνει τη δυνατότητα στον χρήστη να προχωρήσει άμεσα με την κράτηση, κάνοντας τη διαδικασία απλή και γρήγορη. Αυτή η οθόνη είναι κρίσιμη για να δώσει στον χρήστη όλες τις απαραίτητες πληροφορίες που θα τον βοηθήσουν να αποφασίσει αν θα κάνει κράτηση στο συγκεκριμένο εστιατόριο.

Εικόνα 4.2.12 Λεπτομερής Οθόνη Εστιατορίου (Detailed Restaurant Screen)

Η επόμενη οθόνη παρουσιάζει στον χρήστη τα εστιατόρια που έχει αποθηκεύσει στα αγαπημένα του. Είναι ένας γρήγορος και εύκολος τρόπος να βρει και να έχει πρόσβαση στα εστιατόρια που προτιμά,

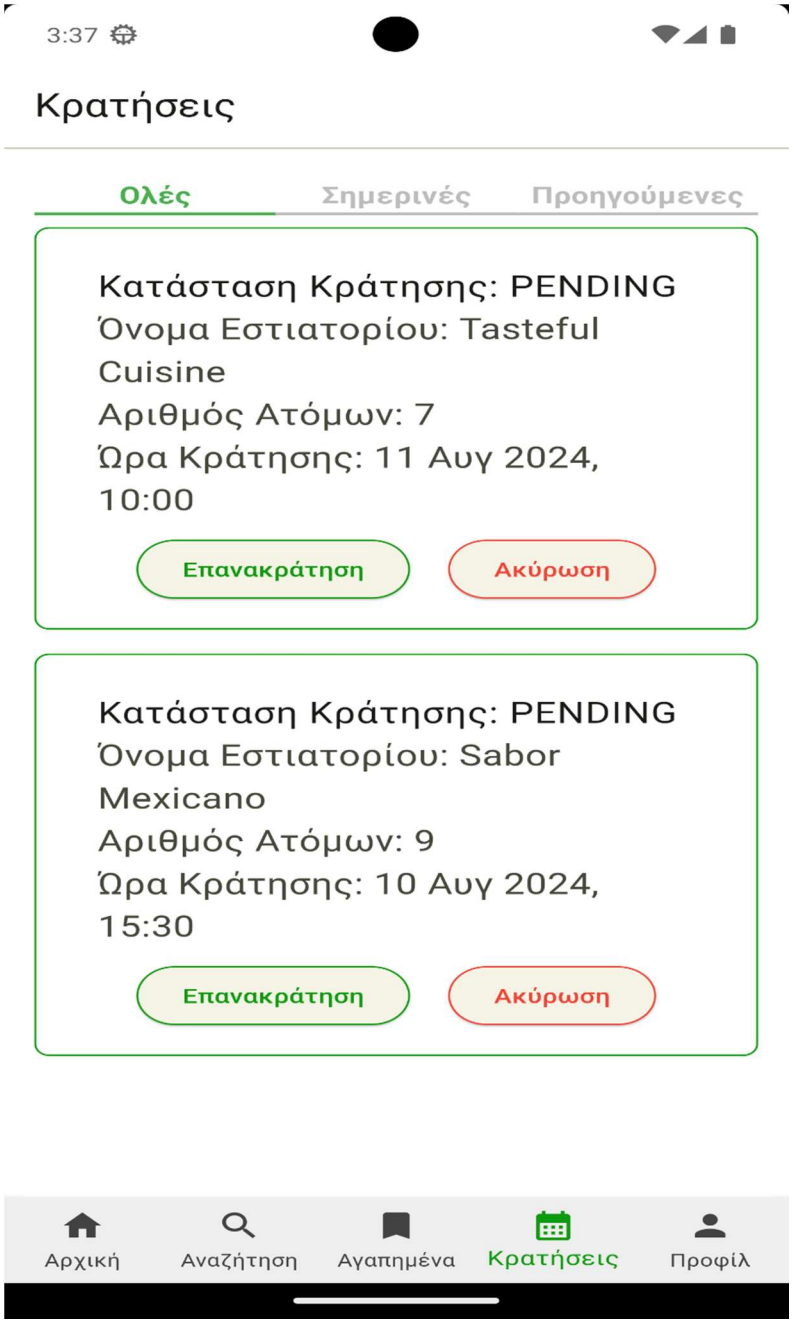
χωρίς να χρειάζεται να τα αναζητά κάθε φορά. Η λίστα περιλαμβάνει πληροφορίες όπως το όνομα του εστιατορίου, την τοποθεσία, την κατηγορία της κουζίνας, την τιμή ανά άτομο και τη βαθμολογία του εστιατορίου. Η παρουσία αυτής της οθόνης βελτιώνει την εμπειρία του χρήστη, παρέχοντας του άμεση πρόσβαση στις αγαπημένες του επιλογές για να κάνει γρήγορα μια κράτηση ή να περιηγηθεί σε αυτά τα εστιατόρια για να δει ξανά τις λεπτομέρειες. Το σύμβολο του σελιδοδείκτη υποδεικνύει ότι το εστιατόριο είναι αποθηκευμένο στα αγαπημένα.



Εικόνα 4.2.13 Οθόνη Αγαπημένων Εστιατορίων (Favorites Screen)

Παρακάτω βλέπουμε η οθόνη να παρέχει στον χρήστη μια επισκόπηση των κρατήσεων που έχει κάνει. Οι κρατήσεις εμφανίζονται με λεπτομέρειες όπως το όνομα του εστιατορίου, ο αριθμός των ατόμων, η

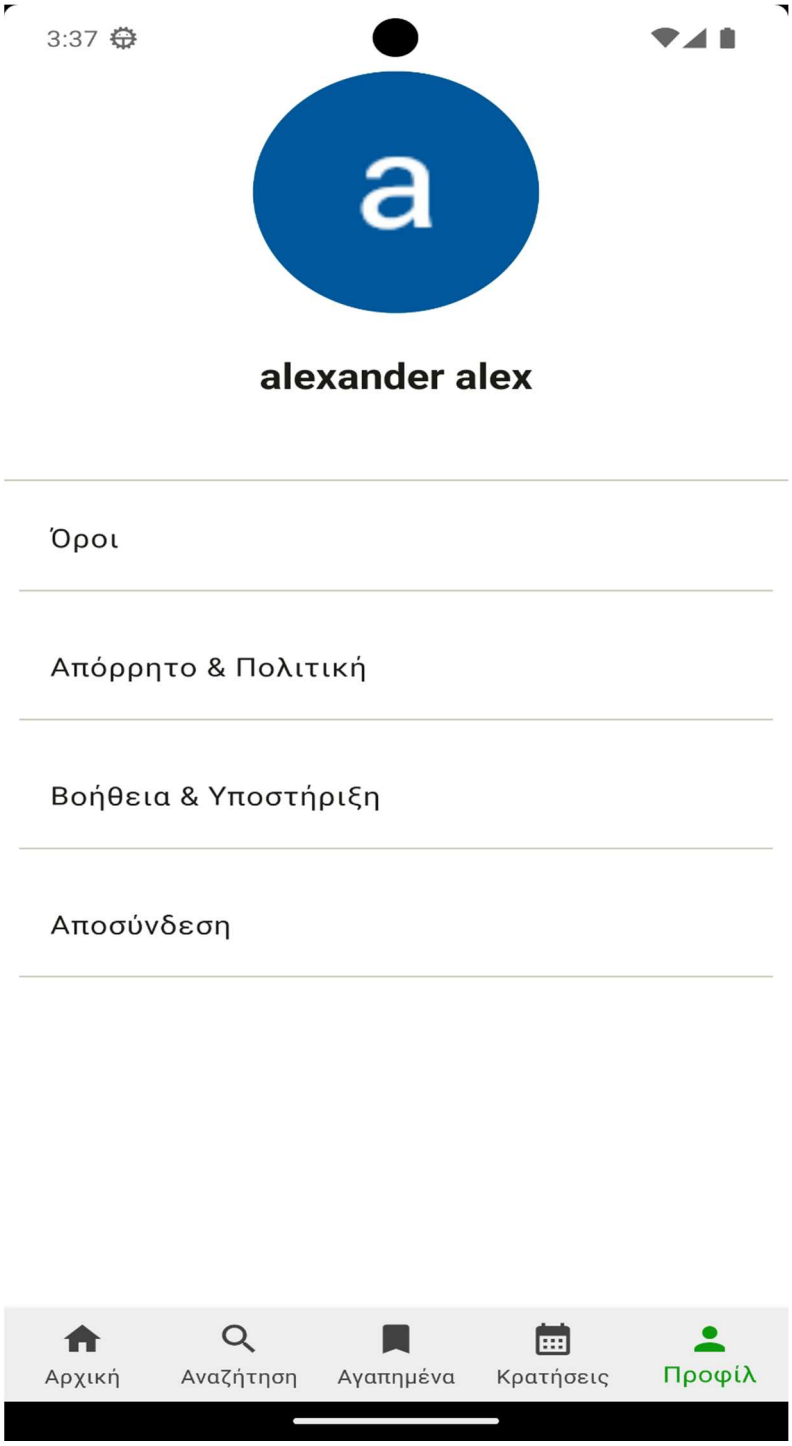
ημερομηνία και η ώρα της κράτησης, καθώς και η τρέχουσα κατάσταση της κράτησης (π.χ., PENDING - εκκρεμής). Ο χρήστης έχει τη δυνατότητα να επαναλάβει την κράτηση ή να την ακυρώσει, μέσω των κουμπιών που βρίσκονται στην κάρτα της κάθε κράτησης. Η οθόνη είναι χωρισμένη σε καρτέλες για εύκολη πλοήγηση μεταξύ όλων των κρατήσεων, των σημερινών και των προηγούμενων κρατήσεων. Αυτή η δυνατότητα επιτρέπει στον χρήστη να διαχειρίζεται εύκολα τις κρατήσεις του, να παρακολουθεί την κατάστασή τους και να αναλάβει δράση όταν είναι απαραίτητο, διασφαλίζοντας ότι η εμπειρία του με την εφαρμογή είναι ομαλή και οργανωμένη.



Εικόνα 4.2.14 Οθόνη Διαχείρισης Κρατήσεων (Reservations Management Screen)

Εν κατακλείδι η οθόνη επιτρέπει στον χρήστη να δει και να διαχειριστεί το προφίλ του στην εφαρμογή. Στην κορυφή εμφανίζεται το όνομα του χρήστη και μια εικόνα προφίλ (ή το αρχικό του, εάν δεν έχει

επιλέξει εικόνα). Ο χρήστης μπορεί να έχει πρόσβαση σε βασικές επιλογές όπως οι Όροι Χρήσης, η Πολιτική Απορρήτου, η Βοήθεια & Υποστήριξη, καθώς και η δυνατότητα αποσύνδεσης από τον λογαριασμό του. Η οθόνη αυτή παρέχει στον χρήστη τα απαραίτητα εργαλεία για να διαχειρίζεται τον λογαριασμό του με ασφάλεια και ευκολία. Η δυνατότητα αποσύνδεσης είναι επίσης διαθέσιμη για περιπτώσεις που ο χρήστης επιθυμεί να τερματίσει την περίοδο σύνδεσής του στην εφαρμογή.



Εικόνα 4.2.15 Οθόνη Προφίλ Χρήστη (User Profile Screen)

4.3 Χρήση Κώδικα για τη Διαχείριση Δεδομένων και Ενεργειών Χρηστών στην Εφαρμογή

```
void signUserIn() async {
  showDialog(
    context: context,
    builder: (context) {
      return const Center(
        child: CircularProgressIndicator(),
      ); // Center
    },
  );

  try {
    final UserCredential authResult =
        await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: emailController.text,
      password: passwordController.text,
    );
    final User? user = authResult.user;

    if (user != null) {
      widget.onLoginSuccess();
      Navigator.of(context).pop();
    } else {
      _showErrorDialog('Η σύνδεση απέτυχε. Παρακαλώ προσπαθήστε ξανά.');
    }
  } on FirebaseAuthException catch (e) {
    Navigator.of(context).pop();

    String errorMessage = firebaseErrorMessagesInGreek[e.code] ??
        'Προέκυψε ένα σφάλμα: ${e.message}';

    _showErrorDialog(errorMessage);
  } catch (e) {
    Navigator.of(context).pop();
    _showErrorDialog('Προέκυψε ένα απρόσμενο σφάλμα: $e');
  }
}
```

Εικόνα 4.3.1 Συνάρτηση Σύνδεσης Χρήστη με Firebase Authentication

Η συνάρτηση υλοποιεί τη σύνδεση του χρήστη στην εφαρμογή μέσω του Firebase Authentication. Ξεκινά με την εμφάνιση ενός κυκλικού δείκτη προόδου που ενημερώνει τον χρήστη ότι η διαδικασία σύνδεσης βρίσκεται σε εξέλιξη. Στη συνέχεια, επιχειρείται η σύνδεση με τα στοιχεία που έχουν εισαχθεί (email και κωδικό πρόσβασης). Εάν η σύνδεση είναι επιτυχής, ο χρήστης προχωρά κανονικά στην εφαρμογή, ενώ σε περίπτωση αποτυχίας εμφανίζονται κατάλληλα μηνύματα σφάλματος για να ενημερωθεί ο χρήστης σχετικά με το πρόβλημα που προέκυψε.

```
static signInWithGoogle(BuildContext context) async {
  FirebaseAuth auth = FirebaseAuth.instance;
  User? user;

  if (kIsWeb) {···
  } else {
    showDialog(···
    final GoogleSignIn googleSignIn = GoogleSignIn();

    final GoogleSignInAccount? googleSignInAccount =
        await googleSignIn.signIn();

    if (googleSignInAccount != null) {
      final GoogleSignInAuthentication googleSignInAuthentication =
          await googleSignInAccount.authentication;

      final AuthCredential credential = GoogleAuthProvider.credential(
        accessToken: googleSignInAuthentication.accessToken,
        idToken: googleSignInAuthentication.idToken,
      );
      try {
        final UserCredential userCredential =
            await auth.signInWithCredential(credential);
        user = userCredential.user;
      // Check if the user already exists
        final userSnapshot = await FirebaseFirestore.instance
            .collection('users')
            .doc(user!.uid)
            .get();
        if (!userSnapshot.exists) {···
              .set({···
        }
      } on FirebaseAuthException catch (e) {
        if (e.code == 'account-exists-with-different-credential') {···
        } else if (e.code == 'invalid-credential') {···
        }
      } catch (e) {
        return null;
      }
    }
  }
  context.go(authRoute);
}
```

Εικόνα 4.3.2 Σύνδεση Χρήστη μέσω Google (Google Sign-In Function)

Η συνάρτηση signInWithGoogle επιτρέπει τη σύνδεση του χρήστη στην εφαρμογή μέσω του λογαριασμού Google. Η διαδικασία ξεκινάει με την αρχικοποίηση της σύνδεσης Google και την απόκτηση των απαραίτητων διαπιστευτηρίων (credentials). Αν η σύνδεση είναι επιτυχής, το σύστημα

ελέγχει αν ο χρήστης υπάρχει ήδη στη βάση δεδομένων Firestore. Εάν δεν υπάρχει, δημιουργείται ένα νέο προφίλ χρήστη. Κατά τη διάρκεια αυτής της διαδικασίας, διαχειρίζονται διάφορα σφάλματα, όπως το αν υπάρχει ήδη λογαριασμός με διαφορετικά διαπιστευτήρια ή αν τα διαπιστευτήρια είναι άκυρα. Τελικά, ο χρήστης κατευθύνεται στην κύρια οθόνη της εφαρμογής μετά την επιτυχή σύνδεση.

```
void signUserUp(BuildContext dialogContext) async {
  showDialog(···
  try {
    // check if both password and confirm password are the same
    if (passwordController.text == confirmPasswordController.text) {
      UserCredential userCredential =
          await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: emailController.text,
        password: passwordController.text,
      );
      final User? user = userCredential.user;
      // Check if the user already exists in Firestore
      final userSnapshot = await FirebaseFirestore.instance
          .collection('users')
          .doc(user!.uid)
          .get();
      if (!userSnapshot.exists) {
        // Create a new user document in Firestore
        await FirebaseFirestore.instance
            .collection('users')
            .doc(user.uid)
            .set({
          'displayName': user.displayName,
          'firstName': firstNameController.text,
          'lastName': lastNameController.text,
          'phone': phoneController.text,
          'email': user.email,
          'photoURL': user.photoURL,
        });
      }
      Navigator.pop(context); // Close the loading circle
      await FirebaseAuth.instance.signOut();
      if (kIsWeb) {···
      } else {···
      }
    } else {
      Navigator.pop(context); // Close the loading circle
      genericErrorMessage("Οι κωδικοί δεν ταιριάζουν!");
    }
  } on FirebaseAuthException catch (e) {
    Navigator.pop(context); // Close the loading circle
    genericErrorMessage(e.code);
  }
}
```

Εικόνα 4.3.3 Δημιουργία Λογαριασμού Χρήστη (User Registration Function)

Η συνάρτηση signUserUp υλοποιεί τη διαδικασία δημιουργίας λογαριασμού χρήστη στην εφαρμογή. Αρχικά, ελέγχεται αν οι δύο κωδικοί που έχει εισαγάγει ο χρήστης (κωδικός και επιβεβαίωση κωδικού)

είναι ίδιοι. Εφόσον είναι, η συνάρτηση δημιουργεί έναν νέο λογαριασμό στο Firebase Authentication με τα στοιχεία του χρήστη (email και κωδικό πρόσβασης). Στη συνέχεια, ελέγχει αν ο χρήστης υπάρχει ήδη στη βάση δεδομένων Firestore, και αν όχι, δημιουργεί ένα νέο έγγραφο με τις πληροφορίες του χρήστη, όπως το όνομα, το επώνυμο, το τηλέφωνο και τη διεύθυνση email. Τέλος, η συνάρτηση διαχειρίζεται τυχόν σφάλματα και ενημερώνει τον χρήστη σε περίπτωση που οι κωδικοί δεν ταιριάζουν ή αν προκύψει άλλο πρόβλημα κατά την εγγραφή.

```
void _searchLocation(String searchTerm) async {
  if (searchTerm.isEmpty) {
    setState(() {
      _searchResults.clear();
    });
    return;
  }
  final url = 'https://maps.googleapis.com/maps/api/place/autocomplete/json?'
      'input=$searchTerm&'
      'types=(regions)&'
      'components=country:gr&'
      'language=el&'
      'key=$apiKey';

  try {
    final response = await http.get(Uri.parse(url));
    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      final List<dynamic> predictions = data['predictions'];
      setState(() {
        _searchResults = predictions;
      });
    } else {
    }
  } catch (error) {}
}
```

Εικόνα 4.3.4 Λειτουργία Αναζήτησης Τοποθεσίας (Location Search Function)

Η συνάρτηση _searchLocation υλοποιεί την αναζήτηση τοποθεσιών χρησιμοποιώντας την υπηρεσία Google Places API. Λαμβάνει ως είσοδο έναν όρο αναζήτησης (search term) και αν αυτός δεν είναι κενός, στέλνει ένα αίτημα προς την API με το συγκεκριμένο όρο. Αν το αίτημα είναι επιτυχές (με κωδικό 200), η συνάρτηση αποκωδικοποιεί την απάντηση από τη Google και ενημερώνει τη λίστα με τα αποτελέσματα αναζήτησης με τις προβλέψεις τοποθεσιών που επέστρεψε η API. Σε περίπτωση αποτυχίας, η συνάρτηση διαχειρίζεται τα σφάλματα για να διασφαλίσει ότι η εφαρμογή συνεχίζει να λειτουργεί ομαλά.

```dart
void _getPlaceDetails(String placeId) async {
  final url = 'https://maps.googleapis.com/maps/api/place/details/json?'
      'place_id=$placeId&'
      'fields=name,formatted_address,geometry,address_components&'
      'key=$apiKey&'
      'language=el';
  final response = await http.get(Uri.parse(url));
  if (response.statusCode == 200) {
    final data = json.decode(response.body);
    final Map<String, dynamic> result = data['result'];
    final List<dynamic> addressComponents = result['address_components'];
    String postalCode = '';
    for (var component in addressComponents) {
      final List<dynamic> types = component['types'];
      if (types.contains('postal_code')) {
        postalCode = component['long_name'];
        break;
      }
    }
    final Map<String, dynamic> geometry = result['geometry'];
    final Map<String, dynamic> location = geometry['location'];
    final double lat = location['lat'];
    final double lng = location['lng'];
    final String name = result['name'];
    final String formattedAddress = result['formatted_address'];
    Navigator.pop(context, {
      'name': name,
      'formatted_address': formattedAddress,
      'lat': lat,
      'lng': lng,
      'postal_code': postalCode,
    });
  }
}
```

Εικόνα 4.3.5 Λήψη Λεπτομερειών Τοποθεσίας (Get Place Details Function)

Η συνάρτηση _getPlaceDetails είναι υπεύθυνη για την απόκτηση λεπτομερειών σχετικά με μια τοποθεσία, χρησιμοποιώντας την υπηρεσία Google Places API. Με βάση το μοναδικό αναγνωριστικό της τοποθεσίας (placeId), η συνάρτηση στέλνει ένα αίτημα στην API για να λάβει πληροφορίες όπως το όνομα της τοποθεσίας, τη διεύθυνση, τα γεωμετρικά δεδομένα (γεωγραφικό πλάτος και μήκος), και τον ταχυδρομικό κώδικα. Εάν το αίτημα είναι επιτυχές, οι πληροφορίες αυτές αποκωδικοποιούνται και αποθηκεύονται σε μεταβλητές, οι οποίες στη συνέχεια επιστρέφονται στον χρήστη μέσω ενός Navigator pop, επιτρέποντας την αξιοποίηση αυτών των δεδομένων στην εφαρμογή.

```dart
Future<void> _getCurrentLocation() async {
  setState(() {
    _isLoading = true;
  });
  try {
    Position position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high,
    );
    List<Placemark> placemarks = await placemarkFromCoordinates(
      position.latitude,
      position.longitude,
    );
    if (placemarks.isNotEmpty) {
      Placemark firstPlacemark =
          placemarks.first;
      if (firstPlacemark.name != null &&
          firstPlacemark.locality != null &&
          firstPlacemark.postalCode != null) {
        Navigator.pop(context, {
          'name': firstPlacemark.name!,
          'formatted_address': firstPlacemark.locality!,
          'lat': position.latitude,
          'lng': position.longitude,
          'postal_code': firstPlacemark.postalCode!,
        });
      }
    }
  } catch (e) {
  } finally {
    if (mounted) {
      setState(() {
        _isLoading =
            false;
      });
    }
  }
}
```

Εικόνα 4.3.6 Λήψη Τρέχουσας Τοποθεσίας Χρήστη (Get Current Location Function)

Η συνάρτηση _getCurrentLocation είναι υπεύθυνη για την απόκτηση της τρέχουσας τοποθεσίας του χρήστη χρησιμοποιώντας το Geolocator. Η συνάρτηση προσπαθεί να λάβει την ακριβή γεωγραφική θέση (γεωγραφικό πλάτος και μήκος) του χρήστη. Στη συνέχεια, χρησιμοποιεί αυτά τα δεδομένα για να ανακτήσει επιπλέον πληροφορίες σχετικά με την τοποθεσία, όπως το όνομα της τοποθεσίας, τη διεύθυνση και τον ταχυδρομικό κώδικα. Εάν όλα τα απαραίτητα δεδομένα είναι διαθέσιμα, επιστρέφονται στον χρήστη μέσω ενός Navigator pop, επιτρέποντας την αξιοποίησή τους σε άλλα μέρη της εφαρμογής. Τέλος, η συνάρτηση διαχειρίζεται τυχόν σφάλματα και διασφαλίζει ότι η φόρτωση τερματίζεται σωστά.

```
static Future getRestaurantsByPostalCode(String postalCode) async {
  final result = await FirebaseFirestore.instance
      .collection("restaurants")
      .where('Location.postalCode', isEqualTo: postalCode)
      .get();

  return result.docs
      .map((e) => {
            'id': e.id, //  document ID
            ...e.data(), // document data
          })
      .toList();
}
```

Εικόνα 4.3.7 Ανάκτηση Εστιατορίων με Βάση τον Ταχυδρομικό Κώδικα (Get Restaurants by Postal Code Function)

Η συνάρτηση getRestaurantsByPostalCode ανακτά μια λίστα εστιατορίων από τη βάση δεδομένων Firestore με βάση τον ταχυδρομικό κώδικα που παρέχεται ως παράμετρος. Χρησιμοποιώντας το Firestore, η συνάρτηση πραγματοποιεί ένα ερώτημα (query) στη συλλογή "restaurants" για να βρει όλα τα έγγραφα που έχουν τον συγκεκριμένο ταχυδρομικό κώδικα. Τα αποτελέσματα επιστρέφονται ως λίστα αντικειμένων, η οποία περιλαμβάνει το αναγνωριστικό του εγγράφου και τα δεδομένα του εστιατορίου. Αυτή η συνάρτηση επιτρέπει την εύκολη αναζήτηση και εμφάνιση εστιατορίων που βρίσκονται σε μια συγκεκριμένη περιοχή.

```
List<Map<String, dynamic>> filterRestaurantsBetweenGivenPriceRanges(
    List<Map<String, dynamic>> allRestaurants, min, max) {
  List<Map<String, dynamic>> filteredRestaurants = [];
  for (var restaurant in allRestaurants) {
    double? price = double.tryParse(restaurant['avgPrice'] ?? '');
    if (price != null && price >= min && price <= max) {
      filteredRestaurants.add(restaurant);
    }
  }
  return filteredRestaurants;
}
```

Εικόνα 4.3.8 Φιλτράρισμα Εστιατορίων ανάμεσα σε Δεδομένα Εύρη Τιμών

Αυτή η συνάρτηση φιλτράρει τα εστιατόρια βάσει της μέσης τιμής τους, επιστρέφοντας εκείνα που βρίσκονται εντός ενός συγκεκριμένου εύρους τιμών.

```
List<Map<String, dynamic>> sortRestaurantsAscendingByRatings(
    List<Map<String, dynamic>> allRestaurants) {
  allRestaurants.sort((b, a) {
    double ratingA = a['rating'] ?? 0.0;
    double ratingB = b['rating'] ?? 0.0;
    return ratingA.compareTo(ratingB);
  });
  return List.from(allRestaurants);
}
```

Εικόνα 4.3.9 Ταξινόμηση Εστιατορίων κατά Αύξουσα Βαθμολογία

Η παρακάτω συνάρτηση ταξινομεί τα εστιατόρια σε αύξουσα σειρά με βάση τη βαθμολογία τους, επιτρέποντας στον χρήστη να δει πρώτα τα εστιατόρια με τις χαμηλότερες βαθμολογίες.

```
List<Map<String, dynamic>> filterRestaurantsByCuisines(
    List<Map<String, dynamic>> allRestaurants) {
  List<Map<String, dynamic>> filteredRestaurants = [];

  for (var restaurant in allRestaurants) {
    String cuisine = restaurant['cuisine'] ?? '';

    if (selectedCuisines.contains(cuisine)) {
      filteredRestaurants.add(restaurant);
    }
  }
  return filteredRestaurants;
}
```

Εικόνα 4.3.10 Φιλτράρισμα Εστιατορίων βάσει Κουζίνας

Η συνάρτηση αυτή φιλτράρει τα εστιατόρια ανάλογα με τον τύπο κουζίνας που επιλέγει ο χρήστης (π.χ. ελληνική, ιταλική).

```
List<Map<String, dynamic>> sortRestaurantsAscendingByprice(
    List<Map<String, dynamic>> allRestaurants) {
  allRestaurants.sort((a, b) {
    double? numericPriceA = double.tryParse(a['avgPrice'] ?? '');
    double? numericPriceB = double.tryParse(b['avgPrice'] ?? '');
    numericPriceA ??= double.infinity;
    numericPriceB ??= double.infinity;
    return numericPriceA.compareTo(numericPriceB);
  });
  return List.from(allRestaurants);
```

Εικόνα 4.3.11 Ταξινόμηση Εστιατορίων κατά Αύξουσα Τιμή

Αυτή η συνάρτηση ταξινομεί τα εστιατόρια σε αύξουσα σειρά με βάση τη μέση τιμή ανά άτομο, διευκολύνοντας τον χρήστη στην επιλογή του οικονομικότερου εστιατορίου.

```
List<Map<String, dynamic>> filterRestaurantsByRating(
    List<Map<String, dynamic>> allRestaurants, double sliderRatingValue) {
  List<Map<String, dynamic>> filteredRestaurants = [];
  for (var restaurant in allRestaurants) {
    double rating = (restaurant['rating'] as num).toDouble();
    if (rating >= sliderRatingValue) {
      if (!filteredRestaurants.contains(restaurant)) {
        filteredRestaurants.add(restaurant);
      }
    }
  }
  return filteredRestaurants;
}
```

Εικόνα 4.3.12 Φιλτράρισμα Εστιατορίων Βάσει Βαθμολογίας

Ο κώδικας αυτός δημιουργεί μια λίστα εστιατορίων που έχουν βαθμολογία μεγαλύτερη ή ίση με μια τιμή που ορίζεται από το χρήστη μέσω ενός slider. Για κάθε εστιατόριο στην αρχική λίστα, ελέγχεται αν η βαθμολογία του είναι μεγαλύτερη από την επιθυμητή τιμή. Εφόσον πληροί το κριτήριο, το εστιατόριο προστίθεται στη λίστα των φιλτραρισμένων αποτελεσμάτων.

```
List<Map<String, dynamic>> filterRestaurantsByDistance(
    List<Map<String, dynamic>> restaurants,
    double centerLat,
    double centerLon,
    double maxDistance) {
  List<Map<String, dynamic>> nearbyRestaurants = [];
  double restaurantLat;
  double restaurantLon;

  for (var restaurant in restaurants) {
    var location = restaurant['Location'];
    if (location != null) {
      var coordinates = location['coordinates'] as GeoPoint?;
      if (coordinates != null) {
        var coordinates = location['coordinates'] as GeoPoint;
        restaurantLat = coordinates.latitude;
        restaurantLon = coordinates.longitude;
        double distance = Geolocator.distanceBetween(
            centerLat, centerLon, restaurantLat, restaurantLon);

        if (distance <= maxDistance * 1000) {
          nearbyRestaurants.add(restaurant);
        }
      }
    }
  }
  return nearbyRestaurants;
}
```

Εικόνα 4.3.13 Φιλτράρισμα Εστιατορίων με Βάση την Απόσταση

Η συνάρτηση αυτή φιλτράρει τα εστιατόρια βάσει της απόστασης από την τρέχουσα τοποθεσία του χρήστη, επιστρέφοντας μόνο εκείνα που βρίσκονται εντός μιας καθορισμένης ακτίνας.

```
static Future getRestaurantData(String? restaurantId) async {
  try {
    DocumentSnapshot restauranSnapshot = await FirebaseFirestore.instance
        .collection('restaurants')
        .doc(restaurantId)
        .get();
    if (restauranSnapshot.exists) {
      Map<String, dynamic>? data =
          restauranSnapshot.data() as Map<String, dynamic>?;

      return data;
    } else {···
    }
  } catch (e) {
    return null;
  }
}
```

Εικόνα 4.3.14 Ανάκτηση Δεδομένων Εστιατορίου

Αυτή η συνάρτηση χρησιμοποιείται για την ανάκτηση των δεδομένων ενός συγκεκριμένου εστιατορίου από τη βάση δεδομένων του Firestore, χρησιμοποιώντας το μοναδικό αναγνωριστικό του εστιατορίου. Αρχικά, γίνεται λήψη του εγγράφου από τη συλλογή "restaurants" με βάση το restaurantId. Αν το έγγραφο υπάρχει, τα δεδομένα του εστιατορίου επιστρέφονται ως Map<String, dynamic>. Σε περίπτωση που το έγγραφο δεν βρεθεί ή υπάρξει κάποιο σφάλμα, η συνάρτηση επιστρέφει null, διασφαλίζοντας ότι η εφαρμογή μπορεί να χειριστεί περιπτώσεις αποτυχίας.

```
body: FutureBuilder<DocumentSnapshot>(
  future: FirebaseFirestore.instance
      .collection('restaurants')
      .doc(restaurantId)
      .get(),
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(child: CircularProgressIndicator());
    } else if (snapshot.hasError) {
      return Center(child: Text('Σφάλμα: ${snapshot.error}'));
    } else if (!snapshot.hasData || !snapshot.data!.exists) {
      return const Center(child: Text('Δεν υπάρχουν διαθέσιμες φωτογραφίες'));
    } else {
      List<dynamic> photos = snapshot.data!['photos'];
      return GridView.builder(
        padding: const EdgeInsets.all(10),
        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount( // SliverG
        itemCount: photos.length,
        itemBuilder: (context, index) {
          String photoUrl = photos[index];
          return GestureDetector(
            onTap: () => _showImageDialog(context, photoUrl),
            child: CachedNetworkImage( // CachedNetworkImage ···
          ); // GestureDetector
        },
      ); // GridView.builder
    }
  },
), // FutureBuilder
```

Εικόνα 4.3.15 Δημιουργία Πλέγματος Φωτογραφιών Εστιατορίου

Η συγκεκριμένη λειτουργία αναλαμβάνει την ανάκτηση και απεικόνιση φωτογραφιών ενός εστιατορίου σε μορφή πλέγματος (grid). Με τη χρήση του FutureBuilder, τα δεδομένα του εστιατορίου ανακτώνται από το Firestore και συγκεκριμένα το πεδίο photos. Ανάλογα με την κατάσταση της σύνδεσης ή την ύπαρξη σφαλμάτων, εμφανίζονται κατάλληλα μηνύματα ή ένας δείκτης φόρτωσης. Όταν τα δεδομένα είναι διαθέσιμα, οι φωτογραφίες τοποθετούνται σε πλέγμα και κάθε μία από αυτές μπορεί να επιλεχθεί για προβολή σε μεγαλύτερο μέγεθος μέσω διαλόγου (dialog).

```dart
Future<void> createReservation({
  required Timestamp? reservationDateAndTime,
  required int numberOfGuests,
  required String specialRequests,
  required String contactName,
  required String contactPhoneNumber,
  required String contactEmail,
  required String userID,
  required String? restaurantID,
  required String restaurantName,
  required String reservationStatus,
  required Timestamp? creationTimestamp,
  required Timestamp? lastUpdatedTimestamp,
}) async {
  try {
    final CollectionReference reservationsCollection =
        FirebaseFirestore.instance.collection('reservations');

    await reservationsCollection.add({
      'dateTime': reservationDateAndTime,
      'numberOfGuests': numberOfGuests,
      'specialRequests': specialRequests,
      'contactName': contactName,
      'contactPhoneNumber': contactPhoneNumber,
      'contactEmail': contactEmail,
      'userID': userID,
      'restaurantID': restaurantID,
      'restaurantName': restaurantName,
      'reservationStatus': reservationStatus,
      'creationTimestamp': creationTimestamp,
      'lastUpdatedTimestamp': lastUpdatedTimestamp
    });
  } catch (e) {
  }
}
```

Εικόνα 4.3.16 Δημιουργία Κράτησης σε Εστιατόριο

Η λειτουργία αυτή αναλαμβάνει τη δημιουργία μιας νέας κράτησης σε εστιατόριο και την αποθήκευση των σχετικών δεδομένων στο Firestore. Καταχωρούνται πληροφορίες όπως η ημερομηνία και ώρα της κράτησης, ο αριθμός των επισκεπτών, τυχόν ειδικά αιτήματα, τα στοιχεία επικοινωνίας του χρήστη, καθώς και η τρέχουσα κατάσταση της κράτησης. Όλα τα δεδομένα αποθηκεύονται σε μια συλλογή με όνομα "reservations", δημιουργώντας έτσι μια ολοκληρωμένη εγγραφή για κάθε κράτηση.

```
Future<void> addFavoriteRestaurant(
    String? userId, String restaurantId) async {
  try {
    await FirebaseFirestore.instance
        .collection('favorites')
        .doc(userId) // Using user ID as the document ID
        .set({
      'userFavorites': {
        restaurantId: {'isFavorite': true},
      },
    }, SetOptions(merge: true)); // Using merge to update existing data
  } catch (e) {}
}
```

Εικόνα 4.3.17 Προσθήκη Αγαπημένου Εστιατορίου

Η παρακάτω συνάρτηση επιτρέπει την προσθήκη ενός εστιατορίου στη λίστα αγαπημένων του χρήστη. Η συνάρτηση addFavoriteRestaurant χρησιμοποιεί το Firebase Firestore για να αποθηκεύσει το αναγνωριστικό του εστιατορίου στη συλλογή "favorites", με το userId ως το μοναδικό αναγνωριστικό εγγράφου. Σε περίπτωση που το έγγραφο υπάρχει ήδη, τα δεδομένα ενημερώνονται χρησιμοποιώντας την επιλογή SetOptions(merge: true), διασφαλίζοντας ότι δεν θα χαθούν τα υπάρχοντα δεδομένα.

```
Future<bool> isFavoriteRestaurant(String userId, String restaurantId) async {
  try {
    final DocumentSnapshot userDoc = await FirebaseFirestore.instance
        .collection('favorites')
        .doc(userId)
        .get();

    if (userDoc.exists) {
      final Map<String, dynamic>? userData =
          userDoc.data() as Map<String, dynamic>?;

      final Map<String, dynamic> userFavorites =
          (userData?['userFavorites'] as Map<String, dynamic>?) ?? {};

      final bool isFavorite = userFavorites.containsKey(restaurantId) &&
          userFavorites[restaurantId]?['isFavorite'] == true;
      return isFavorite;
    }

    return false;
  } catch (e) {
    return false;
    }
}
```

Εικόνα 4.3.18 Έλεγχος Αν Ένα Εστιατόριο Είναι Αγαπημένο

Η συνάρτηση isFavoriteRestaurant ελέγχει αν ένα συγκεκριμένο εστιατόριο βρίσκεται στη λίστα αγαπημένων του χρήστη. Χρησιμοποιεί το userId για να ανακτήσει τα δεδομένα του χρήστη από τη συλλογή "favorites" στο Firebase Firestore. Αν το έγγραφο του χρήστη υπάρχει και περιέχει το συγκεκριμένο εστιατόριο με ένδειξη ότι είναι αγαπημένο, επιστρέφει true. Αν δεν βρεθεί, επιστρέφει false, υποδεικνύοντας ότι το εστιατόριο δεν είναι στη λίστα αγαπημένων του χρήστη.

```
Future<void> removeFavoriteRestaurant(
    String userId, String restaurantId) async {
  try {
    await FirebaseFirestore.instance
        .collection('favorites')
        .doc(userId)
        .update({
      'userFavorites.$restaurantId': FieldValue.delete(),
    });
  } catch (e) {}
}
```

Εικόνα 4.3.19 Αφαίρεση Εστιατορίου από Αγαπημένα

Η συνάρτηση removeFavoriteRestaurant αφαιρεί ένα εστιατόριο από τη λίστα αγαπημένων του χρήστη. Χρησιμοποιώντας το userId, η συνάρτηση εντοπίζει το έγγραφο του χρήστη στη συλλογή "favorites" του Firebase Firestore και ενημερώνει το έγγραφο αφαιρώντας το συγκεκριμένο restaurantId από τη λίστα αγαπημένων. Η αφαίρεση γίνεται με την χρήση του FieldValue.delete(), διασφαλίζοντας ότι το εστιατόριο δεν θα εμφανίζεται πλέον στη λίστα αγαπημένων του χρήστη.

```
Future<void> _toggleFavorite(String restaurantId) async {
  try {
    final isFavorite = await db.isFavoriteRestaurant(user.uid, restaurantId);

    if (isFavorite) {
      await db.removeFavoriteRestaurant(user.uid, restaurantId);
    } else {
      await db.addFavoriteRestaurant(user.uid, restaurantId);
    }

    setState(() {
      // Update the list of favorites
      _favorites = db.getAllFavoriteRestaurants(user.uid);
      _favoriteRestData = db.getAllFavoriteRestaurantsData(_favorites!);
    });
  } catch (e) {}
}
```

Εικόνα 4.3.20 Εναλλαγή Κατάστασης Αγαπημένου Εστιατορίου

Η συνάρτηση toggleFavorite ελέγχει αν ένα εστιατόριο βρίσκεται ήδη στη λίστα αγαπημένων του χρήστη και εναλλάσσει την κατάστασή του ανάλογα. Εάν το εστιατόριο είναι ήδη αγαπημένο, το αφαιρεί από τη λίστα αγαπημένων. Διαφορετικά, το προσθέτει στη λίστα αγαπημένων του χρήστη. Τέλος, ενημερώνει την κατάσταση της εφαρμογής με την πιο πρόσφατη λίστα αγαπημένων εστιατορίων, διασφαλίζοντας ότι η διεπαφή χρήστη αντικατοπτρίζει την τρέχουσα κατάσταση.

```
Future<List<String>>? getAllFavoriteRestaurants(String userId) async {
  try {
    final DocumentSnapshot userDoc = await FirebaseFirestore.instance
        .collection('favorites')
        .doc(userId)
        .get();

    if (userDoc.exists) {
      final Map<String, dynamic>? userData =
          userDoc.data() as Map<String, dynamic>?;

      final Map<String, dynamic> userFavorites =
          (userData?['userFavorites'] as Map<String, dynamic>?) ?? {};

      List<String> favorites = userFavorites.keys.toList();

      return favorites;
    }
    return [];
  } catch (e) {
    return [];
  }
}
```

Εικόνα 4.3.21 Ανάκτηση Όλων των Αγαπημένων Εστιατορίων

Αυτή η συνάρτηση, getAllFavoriteRestaurants, είναι υπεύθυνη για την ανάκτηση της λίστας των αγαπημένων εστιατορίων ενός χρήστη από το Firestore. Εάν τα δεδομένα του χρήστη υπάρχουν στη συλλογή "favorites", η συνάρτηση επιστρέφει τη λίστα των εστιατορίων που έχει επιλέξει ο χρήστης ως αγαπημένα. Σε περίπτωση σφάλματος ή αν δεν υπάρχουν δεδομένα, η συνάρτηση επιστρέφει μια κενή λίστα, εξασφαλίζοντας ότι η εφαρμογή μπορεί να χειριστεί την κατάσταση χωρίς σφάλματα.

```dart
Future<List<Map<String, dynamic>>> getAllReservationseExceptToday(
    String userId) async {
  try {
    DateTime now = DateTime.now();
    DateTime startOfDay = DateTime(now.year, now.month, now.day);

    final QuerySnapshot reservationsBeforeTodaySnapshot =
        await FirebaseFirestore.instance
            .collection('reservations')
            .where('userID', isEqualTo: userId)
            .where('creationTimestamp',
                isLessThan: Timestamp.fromDate(startOfDay))
            .orderBy('creationTimestamp', descending: true)
            .get();

    return reservationsBeforeTodaySnapshot.docs.map((doc) {
      final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
      return {'id': doc.id, ...data};
    }).toList();
  } catch (e) {
    return [];
  }
}
```

Εικόνα 4.3.22 Ανάκτηση Κρατήσεων Χρήστη Εξαιρουμένης της Σημερινής Ημέρας

Η συνάρτηση getAllReservationsExceptToday έχει ως στόχο να ανακτήσει όλες τις κρατήσεις ενός συγκεκριμένου χρήστη από το Firestore, εξαιρώντας τις κρατήσεις που αφορούν τη σημερινή ημέρα. Η συνάρτηση πραγματοποιεί ερώτημα στο Firestore με βάση το userID και το creationTimestamp, φιλτράροντας τις κρατήσεις που έγιναν πριν από την αρχή της τρέχουσας ημέρας. Στη συνέχεια, τα δεδομένα ταξινομούνται κατά φθίνουσα σειρά δημιουργίας και επιστρέφονται ως λίστα για εμφάνιση στην εφαρμογή.

```dart
Future<List<Map<String, dynamic>>> getAllReservations(String userId) async {
  try {
    final QuerySnapshot reservationSnapshot = await FirebaseFirestore.instance
        .collection('reservations')
        .where('userID', isEqualTo: userId)
        .orderBy('creationTimestamp', descending: true)
        .get();

    return reservationSnapshot.docs.map((doc) {
      final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
      return {'id': doc.id, ...data};
    }).toList();
  } catch (e) {
    return [];
  }
}
```

Εικόνα 4.3.23 Ανάκτηση Όλων των Κρατήσεων Χρήστη

Η συνάρτηση getAllReservations ανακτά όλες τις κρατήσεις ενός συγκεκριμένου χρήστη από το Firestore. Η συνάρτηση κάνει ένα αίτημα στη συλλογή reservations, φιλτράροντας τα αποτελέσματα με βάση το userID και ταξινομώντας τα κατά φθίνουσα σειρά με βάση το creationTimestamp. Στη συνέχεια, τα δεδομένα των κρατήσεων μετατρέπονται σε λίστα και επιστρέφονται για περαιτέρω επεξεργασία ή εμφάνιση στην εφαρμογή. Σε περίπτωση αποτυχίας, η συνάρτηση επιστρέφει μια κενή λίστα.

```dart
Future<List<Map<String, dynamic>>> getAllReservationsToday(
    String userId) async {
  try {
    DateTime now = DateTime.now();
    DateTime startOfDay = DateTime(now.year, now.month, now.day);
    DateTime endOfDay =
        startOfDay.add(Duration(days: 1)).subtract(Duration(milliseconds: 1));

    final QuerySnapshot reservationSnapshot = await FirebaseFirestore.instance
        .collection('reservations')
        .where('userID', isEqualTo: userId)
        .where('creationTimestamp',
            isGreaterThanOrEqualTo: Timestamp.fromDate(startOfDay))
        .where('creationTimestamp',
            isLessThanOrEqualTo: Timestamp.fromDate(endOfDay))
        .orderBy('creationTimestamp', descending: true)
        .get();

    return reservationSnapshot.docs.map((doc) {
      final Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
      return {'id': doc.id, ...data};
    }).toList();
  } catch (e) {
    return [];
  }
}
```

Εικόνα 4.3.24 Ανάκτηση Κρατήσεων Χρήστη για την Τρέχουσα Ημέρα

Η συνάρτηση getAllReservationsToday ανακτά όλες τις κρατήσεις ενός χρήστη που έχουν δημιουργηθεί κατά την τρέχουσα ημέρα. Για να επιτύχει αυτό, η συνάρτηση καθορίζει την έναρξη και τη λήξη της ημέρας, στη συνέχεια υποβάλλει ένα ερώτημα στο Firestore, φιλτράροντας τα δεδομένα με βάση το userID και το creationTimestamp, έτσι ώστε να περιλαμβάνονται μόνο οι κρατήσεις που δημιουργήθηκαν εντός της τρέχουσας ημέρας. Τα αποτελέσματα επιστρέφονται σε λίστα για περαιτέρω επεξεργασία ή εμφάνιση στην εφαρμογή. Σε περίπτωση αποτυχίας, επιστρέφεται μια κενή λίστα.

```
Future<void> updateReservationStatus(
    String userId, String reservationId, String newStatus) async {
  try {
    DocumentReference reservationDoc = FirebaseFirestore.instance
        .collection('reservations')
        .doc(reservationId);

    final docSnapshot = await reservationDoc.get();
    if (docSnapshot.exists) {
      await reservationDoc.update({'reservationStatus': newStatus});
    }
  } catch (e) {}
}
```

Εικόνα 4.3.25 Ενημέρωση Κατάστασης Κράτησης

Η συνάρτηση updateReservationStatus αναλαμβάνει να ενημερώσει την κατάσταση μιας υπάρχουσας κράτησης στο σύστημα. Χρησιμοποιώντας το reservationId, η συνάρτηση προσπελάζει το σχετικό έγγραφο στη συλλογή "reservations" του Firebase Firestore. Εφόσον το έγγραφο υπάρχει, ενημερώνει το πεδίο reservationStatus με τη νέα κατάσταση που παρέχεται ως παράμετρος. Αν το έγγραφο δεν υπάρχει ή προκύψει κάποιο σφάλμα, η ενημέρωση αποτυγχάνει σιωπηλά. Αυτή η λειτουργία είναι κρίσιμη για τη διαχείριση και παρακολούθηση της πορείας των κρατήσεων από την εφαρμογή.

```
Future<Map<String, dynamic>?> getUserData(String userId) async {
  try {
    DocumentSnapshot userSnapshot = await FirebaseFirestore.instance
        .collection('users')
        .doc(userId)
        .get();

    if (userSnapshot.exists) {
      Map<String, dynamic> data = userSnapshot.data() as Map<String, dynamic>;
      return data;
    } else {
      return null;
    }
  } catch (e) {
    return null;
  }
}
```

Εικόνα 4.3.26 Ανάκτηση Δεδομένων Χρήστη

Η συνάρτηση getUserData είναι υπεύθυνη για την ανάκτηση των δεδομένων ενός χρήστη από το Firebase Firestore, χρησιμοποιώντας το μοναδικό userId. Με την πρόσβαση στο έγγραφο του χρήστη από τη συλλογή "users", η συνάρτηση επιστρέφει τα δεδομένα ως έναν χάρτη (Map<String, dynamic>).

Εάν το έγγραφο υπάρχει, τα δεδομένα επιστρέφονται, διαφορετικά επιστρέφεται null. Αυτή η διαδικασία χρησιμοποιείται συχνά για την απόκτηση βασικών πληροφοριών του χρήστη που είναι αποθηκευμένες στο σύστημα.

Υλοποίηση Εφαρμογής σε Πολλαπλές Πλατφόρμες / mobile-website

Συνοψίζοντας, η παρούσα εφαρμογή έχει υλοποιηθεί τόσο για mobile συσκευές όσο και για περιβάλλον web. Και στις δύο πλατφόρμες, η βασική αρχιτεκτονική, ο κώδικας και οι λειτουργίες παραμένουν κοινές. Αυτό επιτεύχθηκε μέσω της χρήσης κοινών τεχνολογιών, οι οποίες επιτρέπουν την επαναχρησιμοποίηση κώδικα μεταξύ των διαφορετικών πλατφορμών. Το μεγαλύτερο μέρος της λειτουργικότητας, όπως η διαχείριση δεδομένων, οι συναρτήσεις και η λογική των ερωτημάτων, παραμένει αμετάβλητο, διασφαλίζοντας την ενιαία εμπειρία χρήστη ανεξαρτήτως πλατφόρμας.

Ωστόσο, η διαφορά έγκειται στην παρουσίαση του περιβάλλοντος χρήστη (UI), το οποίο έχει προσαρμοστεί ανάλογα με τις απαιτήσεις κάθε πλατφόρμας. Για παράδειγμα, ενώ οι οθόνες και τα λειτουργικά στοιχεία παραμένουν ίδια στη mobile και στη web έκδοση, η διάταξη και το μέγεθος των στοιχείων του UI έχουν τροποποιηθεί για να προσφέρουν βέλτιστη εμπειρία χρήστη σε κάθε πλατφόρμα.

Επιπλέον, αξίζει να σημειωθεί ότι οι οθόνες και τα λειτουργικά χαρακτηριστικά που έχουν ήδη παρουσιαστεί στη mobile έκδοση της εφαρμογής, ισχύουν επίσης και για την έκδοση web. Οι τροποποιήσεις που έγιναν αφορούν κυρίως τη βελτιστοποίηση της διάταξης για μεγαλύτερες οθόνες και τη βελτιστοποίηση της αλληλεπίδρασης μέσω του browser, χωρίς να αλλάξει η βασική λειτουργικότητα της εφαρμογής.

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Εξέλιξη

Κατά τη διάρκεια της υλοποίησης της παρούσας πτυχιακής εργασίας, το κύριο αντικείμενο της μελέτης ήταν η ανάπτυξη μιας ολοκληρωμένης εφαρμογής κρατήσεων για εστιατόρια, διαθέσιμη τόσο σε mobile όσο και σε web πλατφόρμες. Η εργασία αυτή περιελάμβανε όλα τα στάδια ανάπτυξης, από τη σύλληψη της αρχικής ιδέας μέχρι την τελική υλοποίηση και αξιολόγηση της εφαρμογής. Στόχος ήταν να δημιουργηθεί ένα εργαλείο που θα προσέφερε ευκολία στους χρήστες, βελτιώνοντας παράλληλα την αποδοτικότητα των εστιατορίων στη διαχείριση των κρατήσεων.

Η διαδικασία ξεκίνησε με μια εκτενή ανάλυση των αναγκών τόσο των χρηστών όσο και των εστιατορίων. Προκειμένου να εξασφαλιστεί ότι η εφαρμογή θα ανταποκρινόταν πλήρως στις προσδοκίες όλων των εμπλεκομένων, αναλύθηκαν οι υφιστάμενες λύσεις στην αγορά και εντοπίστηκαν οι τομείς όπου μπορούσε να προστεθεί αξία. Η ανάλυση αυτή οδήγησε στον καθορισμό των βασικών χαρακτηριστικών της εφαρμογής, όπως η ευκολία στη χρήση, η ταχύτητα στις κρατήσεις, η διαχείριση των δεδομένων κρατήσεων και η βελτίωση της επικοινωνίας μεταξύ πελατών και εστιατορίων.

Στη φάση του σχεδιασμού, δόθηκε έμφαση στην ανάπτυξη ενός φιλικού προς τον χρήστη περιβάλλοντος (UI/UX). Οι διαδικασίες που σχεδιάστηκαν ήταν απλές και αποδοτικές, επιτρέποντας στους χρήστες να ολοκληρώνουν τις κρατήσεις τους με ελάχιστα βήματα. Το σύστημα διαχείρισης κρατήσεων που αναπτύχθηκε για τα εστιατόρια ήταν εύχρηστο και επεκτάσιμο, επιτρέποντας την ανάλυση δεδομένων σε πραγματικό χρόνο και την εύκολη προσαρμογή στις ανάγκες των επιχειρήσεων.

Η υλοποίηση της εφαρμογής περιελάμβανε τη χρήση σύγχρονων τεχνολογιών ανάπτυξης λογισμικού και μεθόδων, εξασφαλίζοντας ότι το έργο θα ήταν ευέλικτο και ικανό να προσαρμοστεί σε τυχόν αλλαγές στις απαιτήσεις. Κατά τη διάρκεια της ανάπτυξης, πραγματοποιήθηκαν συνεχείς δοκιμές, τόσο για την αποσφαλμάτωση του κώδικα όσο και για τη βελτίωση της εμπειρίας χρήστη. Η ενσωμάτωση ανατροφοδότησης από δοκιμαστικούς χρήστες έπαιξε καθοριστικό ρόλο στη βελτίωση της τελικής έκδοσης της εφαρμογής.

Η παρούσα πτυχιακή εργασία απέδειξε ότι μια καλοσχεδιασμένη και εύχρηστη εφαρμογή κρατήσεων μπορεί να προσφέρει σημαντικά οφέλη τόσο στους χρήστες όσο και στα εστιατόρια. Οι χρήστες εκτιμούν την ευκολία και την ταχύτητα με την οποία μπορούν να πραγματοποιούν κρατήσεις, ενώ τα εστιατόρια απολαμβάνουν την καλύτερη διαχείριση των κρατήσεων και την αξιοποίηση δεδομένων για τη βελτίωση των υπηρεσιών τους. Η εφαρμογή αυτή επιβεβαίωσε επίσης ότι οι σύγχρονες τεχνολογίες και οι καλές πρακτικές ανάπτυξης λογισμικού μπορούν να συμβάλουν καθοριστικά στην επιτυχία ενός έργου αυτού του είδους. Η ενσωμάτωση ανατροφοδότησης και η συνεχής βελτίωση του προϊόντος κατά τη διάρκεια της ανάπτυξης εξασφάλισαν ότι η τελική εφαρμογή θα ανταποκρινόταν στις ανάγκες της αγοράς και θα προσέφερε μια εξαιρετική εμπειρία χρήστη.

Παρά τα σημαντικά αυτά επιτεύγματα, η πτυχιακή εργασία κατέδειξε επίσης τις δυνατότητες για περαιτέρω ανάπτυξη και βελτίωση της εφαρμογής. Ένας τομέας ιδιαίτερου ενδιαφέροντος είναι η ενσωμάτωση τεχνητής νοημοσύνης (AI) στην πλατφόρμα, με σκοπό την παροχή εξατομικευμένων προτάσεων στους χρήστες, βασισμένων στις προσωπικές τους προτιμήσεις και στην ιστορικότητα των κρατήσεών τους. Η AI μπορεί επίσης να συνεισφέρει στη βελτιστοποίηση της διαχείρισης των κρατήσεων, προβλέποντας τις ώρες αιχμής και προτείνοντας εναλλακτικές επιλογές όταν οι διαθέσιμες επιλογές είναι περιορισμένες.

Η επέκταση της εφαρμογής σε νέες αγορές, τόσο σε εθνικό όσο και σε διεθνές επίπεδο, αποτελεί μια ακόμη σημαντική προοπτική. Η υλοποίηση της εφαρμογής σε περιοχές με περιορισμένη παρουσία

παρόμοιων λύσεων θα μπορούσε να διευρύνει την πελατειακή βάση και να ενισχύσει τη θέση της εφαρμογής στην αγορά.

Η προσθήκη επιπλέον λειτουργιών, όπως η δυνατότητα ολοκλήρωσης πληρωμών εντός της εφαρμογής, θα μπορούσε να προσφέρει μεγαλύτερη ευκολία και ολοκληρωμένη εμπειρία στους χρήστες. Επιπλέον, η ενσωμάτωση συστημάτων αξιολόγησης και σχολιασμού από τους χρήστες θα παρείχε στα εστιατόρια πολύτιμη ανατροφοδότηση, συμβάλλοντας στη διαρκή βελτίωση των υπηρεσιών τους.

Η ανάπτυξη στρατηγικών συνεργασιών με άλλες επιχειρήσεις, όπως υπηρεσίες delivery ή τουριστικούς οδηγούς, θα μπορούσε να εμπλουτίσει περαιτέρω τις δυνατότητες της εφαρμογής, καθιστώντας την έναν ολοκληρωμένο προορισμό για κρατήσεις και άλλες σχετικές υπηρεσίες.

Ένας επιπλέον τομέας μελλοντικής εξέλιξης αφορά την ανάπτυξη ενός admin panel. Το admin panel θα επιτρέπει στους διαχειριστές των εστιατορίων να διαχειρίζονται την πλατφόρμα, να παραμετροποιούν τις λειτουργίες της εφαρμογής και να έχουν πλήρη έλεγχο των δεδομένων και των κρατήσεων. Αυτή η λειτουργία θα ενισχύσει την ευελιξία και τη λειτουργικότητα της εφαρμογής, παρέχοντας στους διαχειριστές τα απαραίτητα εργαλεία για την καλύτερη οργάνωση και διαχείριση των κρατήσεων.

Τέλος, η συνεχής βελτίωση της εμπειρίας χρήστη (UI/UX) είναι απαραίτητη για τη διατήρηση της ανταγωνιστικότητας της εφαρμογής. Η προσαρμογή του σχεδιασμού στις ανάγκες των χρηστών και η ενσωμάτωση των τελευταίων τεχνολογικών εξελίξεων θα συμβάλλουν στη διατήρηση υψηλών επιπέδων ικανοποίησης και στη συνεχή ανάπτυξη της εφαρμογής.

Συμπερασματικά, η παρούσα πτυχιακή εργασία αναδεικνύει τις δυνατότητες μιας καλοσχεδιασμένης εφαρμογής κρατήσεων να βελτιώσει ουσιαστικά την εμπειρία των χρηστών και την αποδοτικότητα των επιχειρήσεων. Η υλοποίηση της εφαρμογής απέδειξε την αξία της στην πράξη, ενώ οι προοπτικές για περαιτέρω ανάπτυξη προσφέρουν μια ισχυρή βάση για μελλοντικές βελτιώσεις και καινοτομίες. Με την υλοποίηση των προτεινόμενων βελτιώσεων και την επέκταση σε νέες αγορές, η εφαρμογή έχει τη δυνατότητα να καταστεί ένας ηγετικός παίκτης στον τομέα των κρατήσεων εστιατορίων, προσφέροντας ανεκτίμητη αξία τόσο στους χρήστες όσο και στους συνεργάτες της.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] HafizaMahrukhShahzadi, “Restaurant Table Reservation System Using Android Mobile Application” in International Journal of Advanced Research in Science, Engineering and Technology Vol. 5, Issue 9, September 2018

[2]«Exclusive: Quantum Paper And Google's Upcoming Effort To Make Consistent UI Simple». Engadget. 25 Ιουνίου 2014.

[3]«Google Reveals Details About Android L at Google IO». Anandtech. 25 Ιουνίου 2014.

[4]«Google's New, Improved Android Will Deliver A Unified Design Language». Co.Design. 25 Ιουνίου 2014.

[5] Stanton, Lee (2021-08-17). "How to Run Code in VS Code" . Alphr. Archived from the original on 2022-06-02 . Retrieved 2022-04-03.

[6] Buranatrived, J., Vickers, P.: An investigation of the impact of mobile phone and PDA interfaces on the usability of mobile-commerce applications. IEEE 5th International Workshop on Networked Appliances. pp. 90–95. Liverpool (2002).

[7] ISO: International Standard Ergonomic requirements for office work with visual display terminals (VDTs)-Part 11: Guidance on Usability, (1998).

[8] Nielsen, J.: Usability 101: Introduction to Usability, www.nngroup.com/articles/usability-101-introduction-to-usability/ (Accessed: 14.11.2014)

[9] "A Tour of the Dart Language". dart.dev . Retrieved 2018-08-09.visual code

[5] Stanton, Lee (2021-08-17). "How to Run Code in VS Code" . Alphr. Archived from the original on 2022-06-02 . Retrieved 2022-04-03.

[10] Lardinois, Frederic (29 April 2015). "Microsoft launches Visual Studio Code, a free cross-platform code editor for OS X, Linux and Windows" . TechCrunch. Archived from the original on October 28, 2017. Retrieved 15 April 2018.

[11] Devine, Richard (22 December 2022). "How to use Visual Studio Code in a web browser" . Windows Central. Retrieved 11 April 2024.'

[12] Joseph, Scott (September 8, 2006). "Table for 2 Is a Click Away". Orlando Sentinel. p. E3. Archived from the original on October 9, 2016. Retrieved April 12, 2007.

[13]https://www.ifourtechnolab.com/blog/how-to-improve-usability-and-user-friendliness-for-software-web-and-mobile-applications

[14] https://www.interaction-design.org/literature/topics/ux-design

[15]S myth, Neil. 2017. Firebase Essential - Android Edition. eBookFrenzy. Accessed: 02.12.2018

[16] Moroney, Laurence. 2017. The definitive guide to Firebase. Washington, USA. Accessed: 03.11.2018

[17] Google. 2019. Firebase. California: USA. Available: https://firebase.google.com/docs/ Accessed:04.11.2018

[18] https://flutter.dev/

[19] https://firebase.google.com/docs