ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

# «Σχεδιασμός και Ανάπτυξη Mobile Εφαρμογής Ενημέρωσης για την Διαθεσιμότητα Χιονοδρομικών Μονάδων στην Ελλάδα»


**Του φοιτητή**
**Παναγιωτόπουλου Ευάγγελου**
**Αρ. Μητρώου: 174912**

**Επιβλέπων**
**Μπράτσας Χαράλαμπος**
**Επίκουρος καθηγητής**

**Ημερομηνία 30/05/2025**

Τίτλος Δ.Ε. **Σχεδιασμός και Ανάπτυξη Mobile Εφαρμογής Ενημέρωσης για την Διαθεσιμότητα Χιονοδρομικών Μονάδων στην Ελλάδα**

Κωδικός Δ.Ε. **25106**

Ονοματεπώνυμο φοιτητή: **Παναγιωτόπουλος Ευάγγελος**

Ονοματεπώνυμο εισηγητή: **Μπράτσας Χαράλαμπος**

Ημερομηνία ανάληψης Δ.Ε. **22-01-2025**

Ημερομηνία περάτωσης Δ.Ε. **30-05-2025**

*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.*

*«Στην οικογένειά μου και σε όλους μου τους φίλους»*

# Πρόλογος

Η επιλογή αυτής της πτυχιακής εργασίας αποτελεί φυσικό επακόλουθο της ακαδημαϊκής και επαγγελματικής μου πορείας στην ανάπτυξη εφαρμογών για κινητές συσκευές, καθώς και του προσωπικού μου ενδιαφέροντος για τα χιονοδρομικά κέντρα στην Ελλάδα. Έχοντας αποκτήσει σημαντική εμπειρία στον προγραμματισμό, ειδικά στο Flutter framework, είδα τη δημιουργία μιας εφαρμογής για τα χιονοδρομικά κέντρα ως μια μοναδική ευκαιρία να συνδυάσω την τεχνογνωσία μου με το πάθος μου για το snowboard.

Η έμπνευση για την ανάπτυξη της εφαρμογής προέκυψε από την παρατήρηση της απουσίας ψηφιακών εργαλείων που να προσφέρουν ολοκληρωμένες πληροφορίες και υπηρεσίες στους λάτρεις των χειμερινών δραστηριοτήτων στην Ελλάδα. Μέσα από αυτήν την πτυχιακή εργασία, φιλοδοξώ να εξελιχθώ ακόμη περισσότερο στον τομέα της ανάπτυξης mobile εφαρμογών και να συνεισφέρω στη βελτίωση της εμπειρίας των χρηστών που θέλουν να ανακαλύψουν τα χιονοδρομικά κέντρα της χώρας μας.

Παράλληλα, μέσα από αυτήν την προσπάθεια, επιδιώκω να αναδείξω τη σημασία της τεχνολογίας στον τουριστικό τομέα, ενώ θα διευρύνω τις δεξιότητές μου στον προγραμματισμό και στη σχεδίαση εφαρμογών που απευθύνονται σε ένα ευρύτερο κοινό, ξεπερνώντας τα όρια της προσωπικής χρήσης, βελτιώνοντας την κατανόησή μου σε πραγματικά έργα.

# Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής για κινητές συσκευές, η οποία παρέχει ολοκληρωμένες πληροφορίες για τα χιονοδρομικά κέντρα της Ελλάδας. Η εφαρμογή αναπτύχθηκε χρησιμοποιώντας το Flutter framework, αξιοποιώντας τις δυνατότητες RESTful API για τη διαχείριση και παρουσίαση δεδομένων. Στόχος της εφαρμογής είναι η διευκόλυνση των χρηστών στην αναζήτηση πληροφοριών σχετικά με τις συνθήκες, τις υπηρεσίες και τις παροχές των χιονοδρομικών κέντρων, προσφέροντας παράλληλα μια σύγχρονη και εύχρηστη εμπειρία χρήστη.

Η επιλογή του θέματος βασίστηκε στην ανάγκη για καινοτόμες ψηφιακές λύσεις στον τομέα του χειμερινού τουρισμού, καθώς και στο προσωπικό ενδιαφέρον του δημιουργού για τον χειμερινό αθλητισμό. Παράλληλα, η εργασία αυτή φιλοδοξεί να αναδείξει τη σημασία της τεχνολογίας στην ενίσχυση του τουριστικού κλάδου και την παροχή αξιόπιστων εργαλείων στους χρήστες.

Η εφαρμογή σχεδιάστηκε με γνώμονα την επεκτασιμότητα, την απόδοση και τη φιλικότητα προς τον χρήστη, ενσωματώνοντας σύγχρονες αρχές ανάπτυξης mobile εφαρμογών. Το έργο αυτό συμβάλλει στη βελτίωση της εμπειρίας των χρηστών που ενδιαφέρονται για τα χιονοδρομικά κέντρα και παράλληλα αποτελεί ένα σημαντικό βήμα για την προσωπική εξέλιξη του δημιουργού στον τομέα του mobile development.

# «Design & Development of a Mobile Application for Ski Resorts in Greece»

«Evangelos Panagiotopoulos»

## Abstract

This thesis focuses on the development of a mobile application that provides comprehensive information about ski resorts in Greece. The application was developed using the Flutter framework, leveraging the capabilities of a RESTful API for data management and presentation. Its primary objective is to facilitate users in accessing information about conditions, services, and amenities at ski resorts, while delivering a modern and user-friendly experience.

The selection of this topic was motivated by the identified need for innovative digital solutions in the field of winter tourism and the creator's personal interest in winter sports. Additionally, this project aims to highlight the significance of technology in enhancing the tourism sector and offering reliable tools for users.

The application is designed with scalability, performance, and user-friendliness as key principles, incorporating modern practices in mobile application development. This project contributes to improving the experience of users interested in ski resorts and serves as a valuable step in the creator's personal growth in the field of mobile development.

# Ευχαριστίες

Ξεκινώντας θα ήθελα να ευχαριστήσω την οικογένεια μου, η οποία με στηρίζει σε κάθε βήμα της ζωής όπως και σε αυτό της πτυχιακής εργασίας και το κλείσιμο αυτού του ακαδημαϊκού ταξιδιου.

Στην συνέχεια θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου που με βοήθησαν και με στήριξαν, είτε δανείζοντας μου εξοπλισμό είτε δίνοντας μου μία γνώμη ή συμβουλή είτε απλά να ακούσουν κάποιο παράπονο ή να δώσουν υποστηρικτικά λόγια.

Επίσης θα ήθελα να ευχαριστήσω τους καθηγητές Χαράλαμπο Μπράτσα για την εμπιστοσύνη του σε μένα και την καθοδήγηση του κατά την ανάπτυξη της εργασίας, με εποικοδομητική κριτική, προτάσεις και σχόλια. Τον κ. Γουλιάνα Κωνσταντίνο ο οποίος στηρίζει και βοηθάει τους φοιτητές από την πρώτη στιγμή που εντασσόμαστε στην σχολή μέχρι το τέλος την αποφοίτησης.

Τέλος θα ήθελα να ευχαριστήσω τους συμφοιτητές Πάντσο Ξενοφών, Βενιαμίν Σεφερίδη και Μεϊμάρογλου Ανδρέα οι οποίοι δημιούργησαν αντίστοιχη πτυχιακή εργασία σε iOS, Web και Backend, δίνοντας έτσι την δυνατότητα να δημιουργήσουμε μια ολοκληρωμένη εφαρμογή.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Συντομογραφίες

| | |
|---|---|
| Δ.Ε. | Διπλωματική Εργασία |
| ΔΙΠΑΕ | Διεθνές Πανεπιστήμιο Ελλάδος |
| Π.Ε. | Πτυχιακή Εργασία |

# Κεφάλαιο 1ο: Εισαγωγή

## 1.1 Εισαγωγή

Οι εφαρμογές για κινητές συσκευές έχουν καταστεί αναπόσπαστο κομμάτι της καθημερινότητας, επηρεάζοντας σημαντικά τον τρόπο που επικοινωνούμε, εργαζόμαστε, διασκεδάζουμε και αναζητούμε πληροφορίες. Με την αλματώδη αύξηση της χρήσης των smartphones, οι εφαρμογές αποτελούν πλέον βασικό μέσο για την παροχή υπηρεσιών και την πρόσβαση σε πληροφορίες, ενώ καλύπτουν ένα ευρύ φάσμα αναγκών, από την ψυχαγωγία και την εκπαίδευση μέχρι τον τουρισμό και την υγεία.

Η ανάπτυξη εφαρμογών για κινητές συσκευές βασίζεται σε σύγχρονες τεχνολογίες, όπως τα frameworks Flutter, React Native και SwiftUI, που προσφέρουν τη δυνατότητα δημιουργίας λειτουργικών, αποδοτικών και ελκυστικών εφαρμογών. Το Flutter, ειδικότερα, έχει κερδίσει δημοτικότητα λόγω της ευελιξίας και της αποτελεσματικότητάς του στην ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες με κοινό κώδικα.

Σε έναν κόσμο όπου οι ανάγκες των χρηστών αλλάζουν συνεχώς, οι εφαρμογές για κινητές συσκευές αποτελούν ένα δυναμικό εργαλείο που συνδυάζει την τεχνολογία με την καθημερινότητα. Μέσα σε αυτό το πλαίσιο, η ανάπτυξη μιας εφαρμογής για τα χιονοδρομικά κέντρα της Ελλάδας αντιπροσωπεύει μια καινοτόμο προσέγγιση στον τουριστικό τομέα, καλύπτοντας την έλλειψη ψηφιακών εργαλείων που παρέχουν ολοκληρωμένες πληροφορίες για τους λάτρεις των χειμερινών σπορ.

## 1.2 Στόχοι και Αντικείμενο Έρευνας

Η παρούσα πτυχιακή εργασία στοχεύει στην ανάπτυξη μιας εφαρμογής για κινητές συσκευές που θα παρέχει ολοκληρωμένες πληροφορίες σχετικά με τα χιονοδρομικά κέντρα της Ελλάδας. Στο πλαίσιο αυτό, η εργασία επιδιώκει να καλύψει την ανάγκη για σύγχρονα ψηφιακά εργαλεία που θα διευκολύνουν τους λάτρεις των χειμερινών σπορ, τόσο στην ενημέρωση όσο και στον προγραμματισμό των δραστηριοτήτων τους.

**Κύριοι Στόχοι:**

1. **Παροχή πληροφοριών για τα χιονοδρομικά κέντρα:**

   Η εφαρμογή θα προσφέρει πληροφορίες όπως οι καιρικές συνθήκες, η κατάσταση των πιστών, οι διαθέσιμες υπηρεσίες και οι τιμές.

2. **Ενίσχυση της εμπειρίας χρήστη:**

   Μέσω μιας εύχρηστης και καλαίσθητης διεπαφής, η εφαρμογή θα στοχεύει στη δημιουργία μιας θετικής εμπειρίας για τους χρήστες, με έμφαση στη φιλικότητα και τη λειτουργικότητα.

3. **Χρήση σύγχρονων τεχνολογιών:**

Η αξιοποίηση του Flutter framework και μιας RESTful API για τη διαχείριση δεδομένων θα επιτρέψει τη δημιουργία μιας επεκτάσιμης και αποδοτικής εφαρμογής.

4. **Συμβολή στον τουριστικό κλάδο:**

   Η εφαρμογή φιλοδοξεί να λειτουργήσει ως εργαλείο υποστήριξης για τον χειμερινό τουρισμό, ενισχύοντας την ορατότητα των χιονοδρομικών κέντρων της Ελλάδας.

**Αντικείμενο Έρευνας:**

Το αντικείμενο της παρούσας έρευνας εστιάζει στη μελέτη και ανάλυση των αναγκών των χρηστών σε σχέση με τα χιονοδρομικά κέντρα και στην ανάπτυξη λύσεων που ανταποκρίνονται στις προσδοκίες τους. Η έρευνα περιλαμβάνει:

- Την ανάλυση της υπάρχουσας κατάστασης στον τομέα των εφαρμογών για κινητές συσκευές που σχετίζονται με χειμερινά σπορ.
- Την καταγραφή των απαιτήσεων χρηστών μέσω ανάλυσης απαιτήσεων και προτύπων διεπαφής
  .
- Την εφαρμογή σύγχρονων μεθοδολογιών ανάπτυξης λογισμικού για τη σχεδίαση και την υλοποίηση μιας αξιόπιστης εφαρμογής.

Η επίτευξη των παραπάνω στόχων αναμένεται να αναδείξει τη σημασία της τεχνολογίας στην υποστήριξη του χειμερινού τουρισμού, ενώ παράλληλα να προσφέρει νέες δυνατότητες στους επισκέπτες των χιονοδρομικών κέντρων της Ελλάδας.

## 1.3 Δομή της Πτυχιακής Εργασίας

Η παρούσα πτυχιακή εργασία είναι οργανωμένη σε πέντε κεφάλαια, τα οποία περιγράφονται αναλυτικά παρακάτω:

**Κεφάλαιο 1ο: Εισαγωγή**

Στο πρώτο κεφάλαιο παρουσιάζονται το γενικό πλαίσιο της πτυχιακής εργασίας, οι στόχοι και το αντικείμενο της έρευνας, καθώς και η δομή της εργασίας.

**Κεφάλαιο 2ο: Τεχνολογίες και Εργαλεία**

Στο δεύτερο κεφάλαιο αναλύονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Συγκεκριμένα, γίνεται αναφορά στο Flutter framework, στις RESTful APIs, καθώς και σε άλλα εργαλεία που υποστήριξαν τη διαδικασία σχεδιασμού και υλοποίησης.

**Κεφάλαιο 3ο: Σχεδιασμός και Υλοποίηση της Εφαρμογής**

Το τρίτο κεφάλαιο περιγράφει τη διαδικασία σχεδιασμού και υλοποίησης της εφαρμογής. Περιλαμβάνει την ανάλυση απαιτήσεων, τον καθορισμό της αρχιτεκτονικής της εφαρμογής, καθώς και την ανάπτυξη των κύριων λειτουργιών της.

**Κεφάλαιο 4ο: Παρουσίαση της Εφαρμογής και Ροή Χρήστη**

Στο τέταρτο κεφάλαιο γίνεται παρουσίαση της εφαρμογής, περιλαμβάνοντας τη διεπαφή χρήστη, τις κύριες λειτουργίες της και τη ροή χρήστη. Επίσης, παρουσιάζονται στιγμιότυπα οθόνης για την καλύτερη κατανόηση της λειτουργικότητας.

**Κεφάλαιο 5ο: Συμπεράσματα**

Το τελευταίο κεφάλαιο συνοψίζει τα αποτελέσματα της πτυχιακής εργασίας, αναλύει τα συμπεράσματα που προκύπτουν από την ανάπτυξη της εφαρμογής και προτείνει πιθανές κατευθύνσεις για μελλοντική εξέλιξη.

Αυτή η δομή έχει ως στόχο να παρουσιάσει με σαφήνεια και λογική ακολουθία όλα τα στάδια της πτυχιακής εργασίας, από την αρχική ιδέα έως την τελική υλοποίηση και αξιολόγηση.

# Κεφάλαιο 2ο: Τεχνολογίες και Εργαλεία

## 2.1 Flutter

Το Flutter αποτελεί ένα λογισμικό ανάπτυξης διεπαφών χρήστη (UI Software Development Kit) ανοιχτού κώδικα, το οποίο αναπτύχθηκε από την Google. Η συγκεκριμένη πλατφόρμα προσφέρει τη δυνατότητα δημιουργίας εφαρμογών πολλαπλών πλατφορμών (cross-platform) από έναν ενιαίο κώδικα (single codebase), υποστηρίζοντας πλήθος λειτουργικών συστημάτων, όπως τα Web, Android, iOS, Linux, macOS, Windows, καθώς και το λειτουργικό Fuchsia. Παρουσιάστηκε για πρώτη φορά το 2015 και κυκλοφόρησε επίσημα τον Μάιο του 2017, σηματοδοτώντας μια σημαντική εξέλιξη στον τομέα της ανάπτυξης λογισμικού.

Η χρήση του Flutter δεν περιορίζεται μόνο στους ανεξάρτητους προγραμματιστές, αλλά υιοθετείται ευρέως από μεγάλους οργανισμούς. Εσωτερικά, η Google χρησιμοποιεί το Flutter για την ανάπτυξη εφαρμογών, όπως το Google Pay και το Google Earth. Επιπλέον, οργανισμοί όπως η ByteDance και η Alibaba έχουν αξιοποιήσει το Flutter για την ανάπτυξη καινοτόμων λύσεων λογισμικού.


Σχήμα 2.1: Λογότυπο Flutter

### 2.1.1 Καινοτομίες και Τεχνολογικά Χαρακτηριστικά

Ένα από τα πλέον σημαντικά και διαφοροποιητικά χαρακτηριστικά του Flutter είναι η ενσωμάτωση μιας αποκλειστικής μηχανής απόδοσης (rendering engine). Αυτή η μηχανή παρέχει τη δυνατότητα απευθείας εξαγωγής των δεδομένων pixel στην οθόνη, παρακάμπτοντας την εξάρτηση από το εγγενές περιβάλλον απόδοσης της κάθε πλατφόρμας.

Σε αντίθεση με άλλα frameworks, όπως οι εγγενείς εφαρμογές Android που βασίζονται στο Android SDK ή το React Native που αξιοποιεί δυναμικά τη στοίβα διεπαφής χρήστη (UI stack) της πλατφόρμας, το Flutter επιτυγχάνει πλήρη έλεγχο του rendering pipeline. Αυτή η αρχιτεκτονική επιλογή παρέχει σημαντικά πλεονεκτήματα, όπως η συνέπεια στην εμφάνιση και τη λειτουργικότητα των εφαρμογών, ανεξαρτήτως της πλατφόρμας στόχου.

### 2.1.2 Εφαρμογές και Πλεονεκτήματα στη Σύγχρονη Ανάπτυξη Λογισμικού

Η αρχιτεκτονική και οι τεχνολογικές δυνατότητες του Flutter το καθιστούν ιδανική επιλογή για την ανάπτυξη σύγχρονων εφαρμογών. Η ικανότητα δημιουργίας αποδοτικών, επεκτάσιμων και καλαίσθητων διεπαφών χρήστη από έναν κοινό κώδικα μειώνει το κόστος ανάπτυξης και συντήρησης λογισμικού, ενώ επιταχύνει τους χρόνους παράδοσης (time-to-market).

Συνολικά, το Flutter αποτελεί ένα πρωτοποριακό εργαλείο που ενσωματώνει σύγχρονες τεχνολογικές πρακτικές, προσφέροντας αξιόπιστες λύσεις για προγραμματιστές και οργανισμούς που επιδιώκουν τη δημιουργία υψηλής ποιότητας λογισμικού για πολλαπλές πλατφόρμες.

## 2.2 Βασικά Στοιχεία του Flutter

Κάποια βασικά στοιχεία του Flutter που αποτελούν τα θεμέλια της ανάπτυξης εφαρμογών:

1. Widgets:

   Τα widgets είναι τα βασικά δομικά στοιχεία της Flutter εφαρμογής. Κάθε στοιχείο της διεπαφής χρήστη (UI), όπως κείμενο, κουμπιά ή ακόμα και διατάξεις, αντιπροσωπεύεται από ένα widget. Υπάρχουν δύο κύριες κατηγορίες widgets: τα StatefulWidgets, που μπορούν να αλλάξουν την κατάσταση τους (state), και τα StatelessWidgets, που είναι στατικά και δεν αλλάζουν κατάσταση.

2. Dart Language και Flutter Framework:

   Το Flutter χρησιμοποιεί τη γλώσσα προγραμματισμού Dart, η οποία είναι απλή και βελτιστοποιημένη για την ανάπτυξη UI. Το Flutter framework παρέχει εργαλεία για τη διαχείριση του state, τη σχεδίαση layout και τη διασύνδεση με APIs.

3. State Management:

   Η διαχείριση της κατάστασης (state management) είναι ένας από τους πιο σημαντικούς τομείς στο Flutter. Εργαλεία όπως το Provider, το Riverpod και το Bloc παρέχουν μεθοδολογίες για την αποδοτική διαχείριση του state της εφαρμογής, διευκολύνοντας τη δημιουργία σύνθετων UI.

4. Hot Reload και Hot Restart:

   Το Flutter διαθέτει χαρακτηριστικά όπως το Hot Reload, που επιτρέπει στους προγραμματιστές να βλέπουν άμεσα τις αλλαγές στον κώδικα χωρίς να χρειάζεται πλήρης επανεκκίνηση της εφαρμογής. Αυτό βελτιώνει την παραγωγικότητα και μειώνει τον χρόνο ανάπτυξης.

5. Rendering και Custom Painting:

Το Flutter παρέχει έναν πλήρη έλεγχο της διαδικασίας απόδοσης (rendering pipeline) και υποστηρίζει την προσαρμοσμένη σχεδίαση (custom painting). Οι προγραμματιστές μπορούν να δημιουργούν μοναδικά UI που ξεφεύγουν από τα παραδοσιακά πρότυπα.

6. Navigation και Routing:

   Το σύστημα πλοήγησης (navigation) του Flutter υποστηρίζει τη δημιουργία εφαρμογών πολλαπλών οθονών (multi-screen applications). Παρέχει εργαλεία για την πλοήγηση, όπως το Navigator και το Material Page Route, ενώ υποστηρίζει και την έννοια των nested navigators.

7. Platform Channels:

   Το Flutter επιτρέπει την επικοινωνία με τον εγγενή (native) κώδικα μέσω των platform channels. Αυτό καθιστά δυνατή την αξιοποίηση εγγενών (native) λειτουργιών των πλατφορμών, όπως αισθητήρες, κάμερες και βάσεις δεδομένων.

8. Packages και Plugins:

   Το Flutter διαθέτει ένα εκτενές οικοσύστημα πακέτων και plugins, τα οποία παρέχουν έτοιμες λύσεις για κοινές λειτουργίες, όπως η διαχείριση δικτύου, η ενσωμάτωση με Firebase και η πρόσβαση σε APIs τρίτων.

Η δομή του Flutter βασίζεται σε έναν μοντέρνο σχεδιασμό και αρχιτεκτονική, που προσφέρει στους προγραμματιστές ευελιξία, ταχύτητα και αποδοτικότητα κατά την ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες.

## 2.3 Dart

Η Dart είναι μια σύγχρονη γλώσσα προγραμματισμού, σχεδιασμένη από την Google με στόχο τη βελτιστοποίηση της ανάπτυξης εφαρμογών για πολλαπλές πλατφόρμες. Έχοντας ως βασική της αρχή την απλότητα και την απόδοση, υποστηρίζει τόσο στατική όσο και δυναμική τυποποίηση, προσφέροντας ευελιξία στους προγραμματιστές ανάλογα με τις απαιτήσεις της εφαρμογής.

Η γλώσσα ενσωματώνει χαρακτηριστικά όπως κλάσεις, mixins και async/await, καθιστώντας την ιδανική για τη δημιουργία εφαρμογών που απαιτούν ταυτόχρονη εκτέλεση εργασιών και ευκολία στον προγραμματισμό. Επιπλέον, με τη δυνατότητα της Dart να μεταγλωττίζεται σε εγγενή ή JavaScript κώδικα, αποτελεί μια ολοκληρωμένη λύση για ανάπτυξη εφαρμογών σε web, κινητές συσκευές και desktop περιβάλλοντα.

Συνδυάζοντας απλότητα σύνταξης και υψηλή απόδοση, η Dart παρέχει έναν ενιαίο τρόπο για την ανάπτυξη σύνθετων διεπαφών χρήστη, χωρίς να εισάγει επιπλέον πολυπλοκότητα. Σε συνδυασμό με

το Flutter, αποτελεί τη βάση για την ανάπτυξη προηγμένων, επεκτάσιμων και γρήγορων εφαρμογών που καλύπτουν τις σύγχρονες ανάγκες των χρηστών.

## 2.4 Firebase

Το Firebase είναι μια ολοκληρωμένη πλατφόρμα ανάπτυξης εφαρμογών, η οποία παρέχεται από την Google, με στόχο να διευκολύνει τη διαδικασία δημιουργίας, διαχείρισης και βελτιστοποίησης εφαρμογών για κινητές συσκευές και ιστό. Προσφέρει πλήθος εργαλείων και υπηρεσιών, όπως βάσεις δεδομένων σε πραγματικό χρόνο, αυθεντικοποίηση χρηστών, φιλοξενία αρχείων, analytics, cloud messaging και machine learning. Το Firebase είναι γνωστό για την ευκολία χρήσης, τη δυνατότητα γρήγορης ανάπτυξης (rapid development) και την εξαιρετική κλιμάκωση (scalability), επιτρέποντας στους προγραμματιστές να επικεντρωθούν στη λογική της εφαρμογής αντί στην υποδομή.

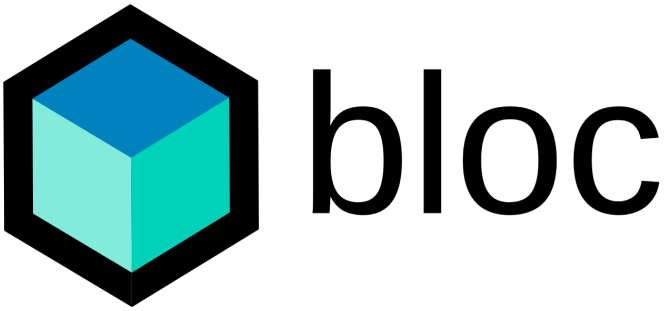
Οι δύο υπηρεσίες του Firebase που χρησιμοποιήθηκαν είναι το Cloud Firestore και το Firebase Authentication. Το Cloud Firestore είναι μια σύγχρονη, NoSQL βάση δεδομένων που αποθηκεύει και συγχρονίζει δεδομένα σε πραγματικό χρόνο μεταξύ των χρηστών και της εφαρμογής, ενώ είναι πλήρως cloud-hosted και προσφέρει ευελιξία και υψηλή απόδοση. Το Firebase Authentication είναι μια εύχρηστη υπηρεσία αυθεντικοποίησης, η οποία υποστηρίζει διάφορες μεθόδους σύνδεσης, όπως email/password, social media (Google, Facebook, Twitter), καθώς και ανώνυμη σύνδεση. Αυτές οι υπηρεσίες συνδυάζονται άριστα, παρέχοντας ένα ασφαλές, αποτελεσματικό και ολοκληρωμένο σύστημα διαχείρισης χρηστών και δεδομένων για κάθε εφαρμογή.


Σχήμα 2.2: Λογότυπο Firebase

## 2.5 Βιβλιοθήκες

Για την υλοποίηση της παρούσας εργασίας αξιοποιήθηκαν διάφορες βιβλιοθήκες ανοιχτού κώδικα, οι οποίες έπαιξαν καθοριστικό ρόλο στην ανάπτυξη της εφαρμογής. Αυτές οι βιβλιοθήκες συνέβαλαν στην ευκολότερη υλοποίηση συγκεκριμένων λειτουργιών, μειώνοντας τον όγκο του απαιτούμενου κώδικα και επιταχύνοντας τη διαδικασία ανάπτυξης. Κυρίως βασισμένες στις γλώσσες JavaScript και TypeScript, υποστήριξαν τη λειτουργικότητα του frontend της εφαρμογής, ενώ παράλληλα επέτρεψαν την ενσωμάτωση πρόσθετων εργαλείων και τεχνολογιών.

### 2.5.1 BLoC




Σχήμα 2.3: Λογότυπο bloc

Το BLoC (Business Logic Component) είναι ένα δημοφιλές σχεδιαστικό πρότυπο (design pattern) για τη διαχείριση της κατάστασης (state management) σε εφαρμογές Flutter. Βασίζεται στην ιδέα του διαχωρισμού της επιχειρηματικής λογικής (business logic) από τη διεπαφή χρήστη (UI), παρέχοντας μια σαφή αρχιτεκτονική δομή. Στο BLoC, η κατάσταση της εφαρμογής διαχειρίζεται μέσω "events" (γεγονότα) που ενεργοποιούνται από τον χρήστη ή το σύστημα, και "streams" (ροές δεδομένων) που ενημερώνουν το UI για τις αλλαγές στην κατάσταση. Αυτή η προσέγγιση προσφέρει επεκτασιμότητα, επαναχρησιμοποίηση κώδικα και καθαρό διαχωρισμό των αρμοδιοτήτων, κάνοντας την ανάπτυξη, τη συντήρηση και τον έλεγχο του κώδικα πιο αποτελεσματική.

### 2.5.2 Flutter Maps

Η βιβλιοθήκη flutter_map είναι ένα ευέλικτο και ισχυρό εργαλείο για την ενσωμάτωση διαδραστικών χαρτών σε εφαρμογές Flutter. Βασίζεται στο Leaflet, μια δημοφιλή βιβλιοθήκη JavaScript, και επιτρέπει στους προγραμματιστές να ενσωματώνουν διάφορους παρόχους χαρτών, όπως το OpenStreetMap, εντελώς δωρεάν—σε αντίθεση με άλλες υπηρεσίες όπως οι χάρτες της Google, που ενδέχεται να έχουν κόστος χρήσης. Η flutter_map υποστηρίζει πολλαπλά χαρακτηριστικά, όπως επίπεδα (layers), markers, polygons, και αλληλεπίδραση του χρήστη με τον χάρτη μέσω zoom, pan και tap events. Χάρη στην ευκολία χρήσης, την εκτενή παραμετροποίηση και την υποστήριξη plugins, αποτελεί ιδανική επιλογή για εφαρμογές που απαιτούν αξιόπιστη, οικονομική και διαδραστική εμφάνιση χαρτών.

### 2.5.3 Maps launcher for Flutter

Η βιβλιοθήκη **maps_launcher** για Flutter παρέχει έναν εύκολο και αποτελεσματικό τρόπο για την εκκίνηση εφαρμογών χαρτών από τη δική σας εφαρμογή. Επιτρέπει στους προγραμματιστές να ανοίγουν γρήγορα την προεγκατεστημένη εφαρμογή χαρτών της συσκευής του χρήστη (π.χ. Google Maps ή Apple Maps), παρέχοντας τη δυνατότητα εμφάνισης τοποθεσιών ή οδηγιών πλοήγησης. Χάρη στην απλότητα και την ευκολία ενσωμάτωσης, το **maps_launcher** αποτελεί ιδανική λύση για εφαρμογές που απαιτούν βασικές λειτουργίες χαρτών και πλοήγησης χωρίς την ανάγκη για περίπλοκη ρύθμιση ή πρόσθετες άδειες.

### 2.5.4 JSON Serializable και JSON Annotation

Οι βιβλιοθήκες json_annotation και json_serializable παρέχουν έναν αποτελεσματικό και αξιόπιστο τρόπο διαχείρισης JSON δεδομένων σε εφαρμογές Flutter. Η json_annotation προσφέρει annotations (σημειώσεις) που επιτρέπουν τον εύκολο ορισμό της δομής των μοντέλων δεδομένων, ενώ η json_serializable είναι υπεύθυνη για την αυτόματη δημιουργία κώδικα μετατροπής από και προς JSON μέσω code generation. Συνδυαστικά, οι δύο αυτές βιβλιοθήκες επιταχύνουν τη διαδικασία δημιουργίας και διαχείρισης μοντέλων, μειώνοντας τον χρόνο ανάπτυξης και τα λάθη που συχνά προκύπτουν κατά τη χειροκίνητη διαχείριση JSON δεδομένων.

### 2.5.5 Firebase Auth for Flutter

Η βιβλιοθήκη firebase_auth είναι ένα ισχυρό και ευέλικτο plugin για την ενσωμάτωση της υπηρεσίας αυθεντικοποίησης χρηστών του Firebase σε εφαρμογές Flutter. Παρέχει εύκολη πρόσβαση σε διάφορες μεθόδους σύνδεσης, όπως email και κωδικό πρόσβασης, σύνδεση μέσω λογαριασμών Google, Facebook, Apple, αλλά και ανώνυμη σύνδεση. Η firebase_auth διαχειρίζεται αποτελεσματικά την κατάσταση του χρήστη (user state) και παρέχει ασφαλείς μεθόδους επαλήθευσης ταυτότητας. Με την ενσωμάτωση αυτής της βιβλιοθήκης, οι προγραμματιστές μπορούν να υλοποιήσουν γρήγορα και αξιόπιστα συστήματα αυθεντικοποίησης χωρίς να ανησυχούν για την πολυπλοκότητα και την ασφάλεια, καθώς όλα διαχειρίζονται μέσω της ισχυρής υποδομής του Firebase.


Σχήμα 2.4: Λογότυπο Firebase Authentication

### 2.5.6 Cloud Firestore Plugin for Flutter

Η βιβλιοθήκη cloud_firestore είναι ένα ισχυρό και αξιόπιστο plugin για την ενσωμάτωση του Cloud Firestore, της NoSQL βάσης δεδομένων του Firebase, σε εφαρμογές Flutter. Επιτρέπει στους προγραμματιστές να αποθηκεύουν, να ανακτούν και να συγχρονίζουν δεδομένα σε πραγματικό χρόνο, χρησιμοποιώντας συλλογές (collections) και έγγραφα (documents). Η cloud_firestore προσφέρει ευέλικτα queries, δυνατότητες offline πρόσβασης και αυτόματο συγχρονισμό, καθιστώντας τη ιδανική επιλογή για εφαρμογές που απαιτούν άμεση ενημέρωση δεδομένων σε πολλαπλούς χρήστες. Χάρη στην ενσωμάτωση της βιβλιοθήκης αυτής, οι προγραμματιστές μπορούν να επικεντρωθούν στη λειτουργικότητα της εφαρμογής τους, αφήνοντας στο Firestore τη διαχείριση δεδομένων και την εξασφάλιση υψηλής απόδοσης και ασφάλειας.


Σχήμα 2.5: Λογότυπο Cloud Firestore

## 2.6 Εργαλεία

Σε αυτό το υποκεφάλαιο παρουσιάζονται τα εργαλεία που αξιοποιήθηκαν για τον σχεδιασμό, την ανάπτυξη, τον έλεγχο και τη διαχείριση του κώδικα της εφαρμογής. Αναφέρονται οι επεξεργαστές κώδικα, καθώς και οι πλατφόρμες που συνέβαλαν σημαντικά στη διαδικασία ανάπτυξης του frontend. Επιπλέον, γίνεται αναφορά στα εργαλεία που χρησιμοποιήθηκαν για την αποθήκευση, τον έλεγχο εκδόσεων και την αυτοματοποίηση διαδικασιών, όπως και σε εκείνα που επιλέχθηκαν για τη δοκιμή και τον έλεγχο της ορθότητας και λειτουργικότητας της εφαρμογής.

### 2.6.1 Visual Studio Code (VS code)

Το Visual Studio Code (VSCode) είναι ένα εξαιρετικά δημοφιλές και ελαφρύ ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), το οποίο χρησιμοποιείται ευρέως στην ανάπτυξη εφαρμογών Flutter. Διαθέτει πλούσιο οικοσύστημα πρόσθετων (plugins), μεταξύ των οποίων ξεχωρίζουν εκείνα για το Flutter και τη γλώσσα Dart, προσφέροντας ολοκληρωμένη υποστήριξη για syntax highlighting, hot reload, debugging και εύκολη διαχείριση project. Μαζί με το Android Studio, αποτελούν τις δύο κυριότερες επιλογές IDE για προγραμματιστές Flutter, με το VSCode να είναι προτιμητέο για την ταχύτητα, την ευελιξία και την απλότητα που παρέχει.


Σχήμα 2.6: Λογότυπο Visual Studio Code

### 2.6.2 Git και Github

Το Git είναι ένα δημοφιλές σύστημα διαχείρισης εκδόσεων (version control system) ανοιχτού κώδικα, το οποίο αξιοποιείται για την παρακολούθηση και διαχείριση αλλαγών στον κώδικα κατά τη διάρκεια της ανάπτυξης ενός έργου. Παρέχει στους προγραμματιστές τη δυνατότητα να διαχειρίζονται πολλαπλές εκδόσεις του κώδικα, να συνεργάζονται αποτελεσματικά εντός ομάδων, καθώς και να αναθεωρούν και να επαναφέρουν οποιαδήποτε αλλαγή κριθεί απαραίτητο. Χρησιμοποιώντας τη δυνατότητα δημιουργίας και συγχώνευσης κλάδων (branches), το Git επιτρέπει στους προγραμματιστές να αναπτύσσουν νέα χαρακτηριστικά και να πειραματίζονται χωρίς να επηρεάζεται η βασική λειτουργικότητα της εφαρμογής.

Το GitHub είναι μια διαδικτυακή πλατφόρμα που βασίζεται στο Git, η οποία παρέχει αποθήκευση και διαχείριση αποθετηρίων κώδικα στο cloud. Εκτός από τις βασικές λειτουργίες του Git, το GitHub προσφέρει πρόσθετα εργαλεία συνεργασίας όπως pull requests και issues, μέσω των οποίων οι προγραμματιστές μπορούν να συζητούν, να αναθεωρούν κώδικα και να εντοπίζουν προβλήματα με διαφανή και οργανωμένο τρόπο. Επιπλέον, το GitHub διαθέτει το GitHub Projects, ένα ολοκληρωμένο εργαλείο για τη διαχείριση εργασιών (tasks) και του κύκλου ζωής τους, από τον σχεδιασμό μέχρι την ολοκλήρωση. Τέλος, η πλατφόρμα υποστηρίζει συνεχή ενσωμάτωση (CI) και αυτοματοποίηση διαδικασιών μέσω εργαλείων όπως το GitHub Actions, βελτιώνοντας την αποδοτικότητα και τη διαχείριση της ανάπτυξης.


Σχήμα 2.7: Λογότυπο GitHub

### 2.6.3 Postman

Το Postman είναι ένα ιδιαίτερα δημοφιλές εργαλείο ανάπτυξης που αξιοποιείται κυρίως για τη δοκιμή και την ανίχνευση προβλημάτων σε APIs (Application Programming Interfaces). Προσφέρει μια εύχρηστη διεπαφής χρήστη για την εκτέλεση αιτημάτων HTTP, την ανάλυση των αποκρίσεων και τη διαχείριση των endpoints του API. Μέσω του Postman, οι προγραμματιστές μπορούν εύκολα να ελέγχουν τη σωστή επικοινωνία μεταξύ frontend και backend, να επαληθεύουν την ακρίβεια των δεδομένων και να αυτοματοποιούν επαναλαμβανόμενες δοκιμές. Επιπρόσθετα, η δυνατότητα αποθήκευσης και κοινής χρήσης των αιτημάτων το καθιστά ένα εξαιρετικά χρήσιμο εργαλείο για την ανάπτυξη, τον έλεγχο και τη συντήρηση σύγχρονων εφαρμογών.


Σχήμα 2.8: Λογότυπο Postman

### 2.6.4 Lucidchart

Το Lucidchart είναι ένα ευέλικτο διαδικτυακό εργαλείο δημιουργίας διαγραμμάτων, που προσφέρει εκτεταμένες δυνατότητες σχεδίασης διαγραμμάτων UML (Unified Modeling Language), ιδιαίτερα χρήσιμες στην ανάπτυξη λογισμικού. Μια από τις βασικές δυνατότητες του είναι η δημιουργία διαγραμμάτων περιπτώσεων χρήσης (use case diagrams), τα οποία βοηθούν στην οπτικοποίηση των αλληλεπιδράσεων μεταξύ χρηστών (actors) και συστημάτων. Μέσω του Lucidchart, οι χρήστες μπορούν να σχεδιάσουν και να τροποποιήσουν εύκολα use cases, ενσωματώνοντας προδιαγεγραμμένα σύμβολα UML, connectors, και διαδραστικά εργαλεία, διευκολύνοντας έτσι τη σαφή αποτύπωση των λειτουργικών απαιτήσεων μιας εφαρμογής. Η απλότητα, η διαδραστικότητα και η συνεργατική φύση του Lucidchart το καθιστούν ιδανική λύση για ομάδες ανάπτυξης που επιθυμούν να οπτικοποιήσουν, να επικοινωνήσουν και να διαχειριστούν αποτελεσματικά τις περιπτώσεις χρήσης των εφαρμογών τους.


Σχήμα 2.9: Λογότυπο Lucidchart

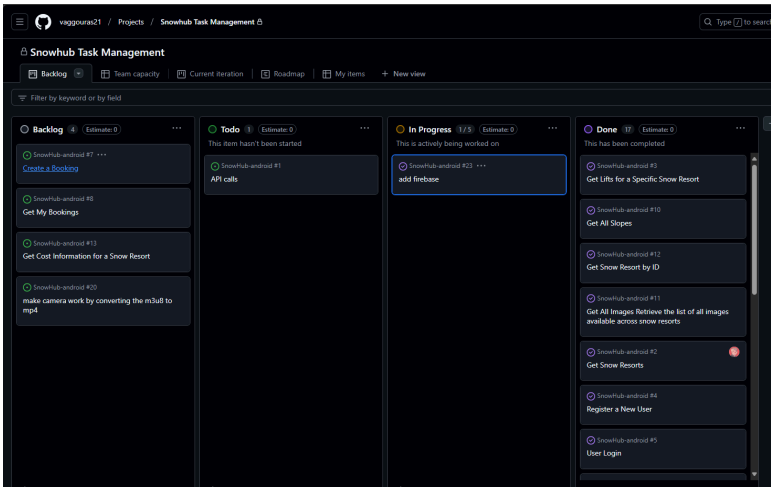
# Κεφάλαιο 3ο: Σχεδιασμός και Υλοποίηση της Εφαρμογής

## 3.1 Εισαγωγή

Για τον σχεδιασμό της εφαρμογής και τον καθορισμό των στόχων της, πραγματοποιήθηκε αρχικά διεξοδική ανάλυση των RESTful APIs, τα οποία έχουν αναπτυχθεί από τον Ξενοφώντα Πάντσο. Η διερεύνηση αυτή υλοποιήθηκε μέσω του Postman, επιτρέποντας την καταγραφή των διαθέσιμων endpoints, των δομών δεδομένων που επιστρέφονται, καθώς και των πιθανών μοντέλων που απαιτούνται για την αποθήκευση και παρουσίαση των πληροφοριών εντός της εφαρμογής. Παράλληλα, εντοπίστηκαν λειτουργίες και χαρακτηριστικά που προκύπτουν έμμεσα από τις αποκρίσεις των APIs, συμβάλλοντας στη διαμόρφωση της τελικής αρχιτεκτονικής της εφαρμογής.

Επιπλέον, πραγματοποιήθηκε ανάλυση της εφαρμογής που αναπτύχθηκε για συσκευές iOS από τον Βενιαμίν Σεφερίδη, προκειμένου να καταγραφεί το user flow, οι διαθέσιμες λειτουργίες και οι δυνατότητες της εφαρμογής. Αυτή η διαδικασία παρείχε πολύτιμες πληροφορίες σχετικά με τη συνολική εμπειρία χρήστη, την αλληλεπίδραση με τα δεδομένα και τον τρόπο παρουσίασής τους. Παράλληλα, προσέφερε μια αρχική εικόνα του σχεδιασμού της διεπαφής χρήστη (UI), η οποία αποτέλεσε τη βάση για την υλοποίηση της Android εφαρμογής, διασφαλίζοντας συνέπεια και ομοιομορφία μεταξύ των διαφορετικών εκδόσεων της εφαρμογής.

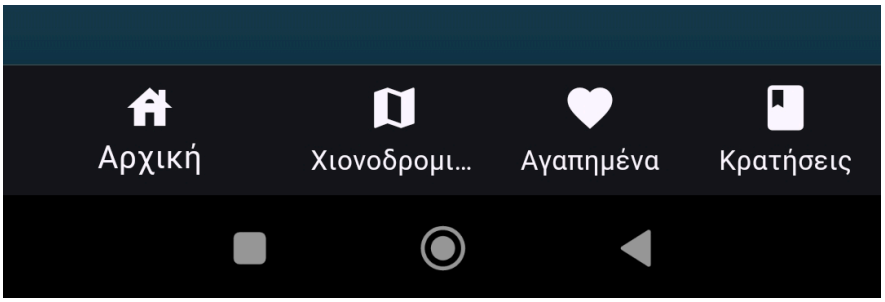
Όλες οι απαιτήσεις της εφαρμογής, αφού καταγράφηκαν και αναλύθηκαν, μεταφέρθηκαν από το αρχικό σχεδιαστικό στάδιο σε GitHub Projects με τη μορφή tasks, επιτρέποντας την αποτελεσματική παρακολούθηση της προόδου ανάπτυξης. Η χρήση του GitHub Projects διευκόλυνε τη διαχείριση των εργασιών και τη δομημένη παρακολούθηση της υλοποίησης, εξασφαλίζοντας έναν οργανωμένο και αποδοτικό κύκλο ανάπτυξης της εφαρμογής.




Σχήμα 3.1: SnowHub GitHub Project

## 3.2 UI και Δομή των Σελίδων Πλοήγησης

Η αρχική φάση του σχεδιασμού επικεντρώθηκε στην οικοδόμηση της θεμελιώδους αρχιτεκτονικής – του "σκελετού" – της εφαρμογής, με κύριο άξονα την εμπειρία χρήστη (UI) και την υλοποίηση βασικών λειτουργιών πλοήγησης. Συγκεκριμένα, υλοποιήθηκαν μηχανισμοί κάθετης και οριζόντιας κύλισης (vertical και horizontal scrolling), καθώς και μια μπάρα πλοήγησης στο κάτω μέρος της οθόνης (bottom navigation bar), η οποία παρέχει άμεση πρόσβαση στις τέσσερις βασικές ενότητες της εφαρμογής: Αρχική, Χιονοδρομικά Κέντρα, Αγαπημένα και Κρατήσεις.



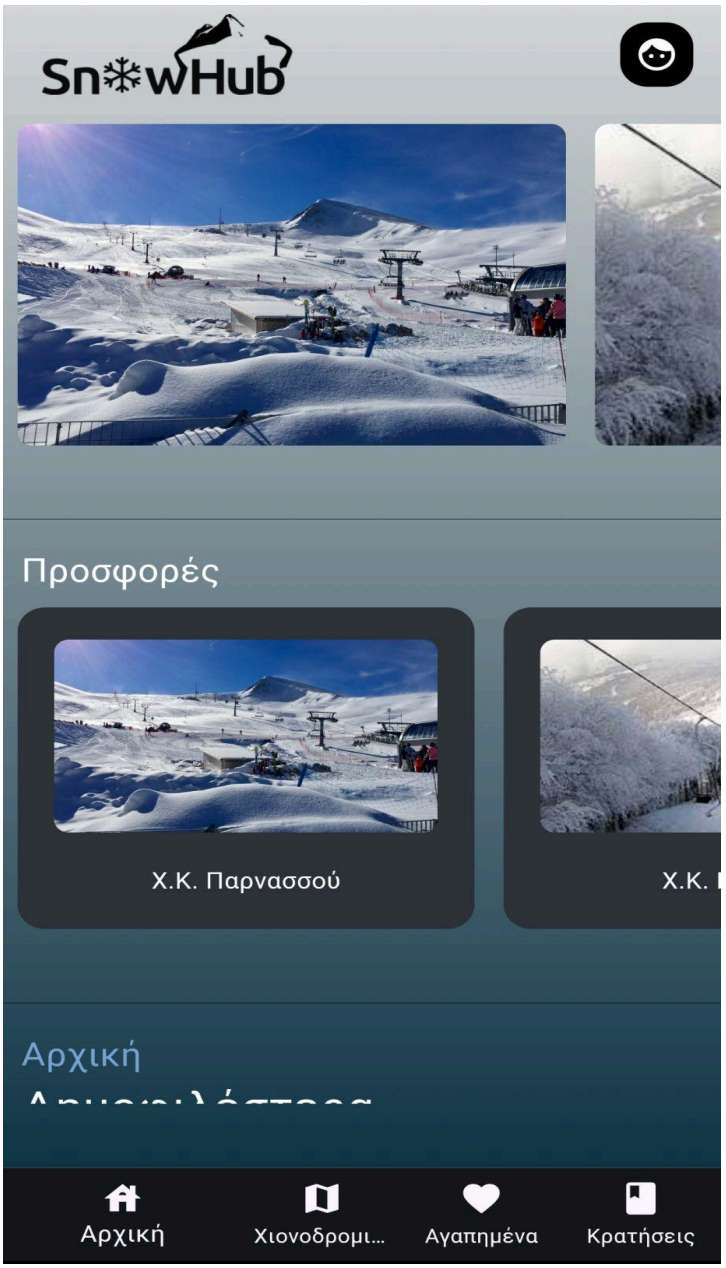

Σχήμα 3.2: Bottom Navigation Bar

### 3.2.1 Αρχική Σελίδα

Στην παρούσα σελίδα υλοποιείται ένα περιβάλλον παρουσίασης πληροφοριών σχετικών με χιονοδρομικά κέντρα, οργανωμένο μέσω ενός CustomScrollView με χρήση Slivers. Η δομή της σελίδας:

1. Μια οριζόντια λίστα με εικόνες χιονοδρομικών κέντρων
2. Μια δεύτερη οριζόντια λίστα υπό τον τίτλο "Προσφορές", η οποία απεικονίζει compact κάρτες χιονοδρομικών
3. Μια κάθετη λίστα με αναλυτικές πληροφορίες για τα χιονοδρομικά

Η επιλογή της χρήσης Slivers ενισχύει την αποδοτικότητα στην απόδοση (rendering) των δεδομένων, επιτρέποντας τη δυναμική φόρτωση των στοιχείων, κάτι που είναι ιδιαίτερα κρίσιμο σε περιπτώσεις όπου υπάρχουν πολλαπλές και πολυπλοκότερες λίστες, ή όταν απαιτείται υποστήριξη αμφίπλευρης κύλισης (scrolling τόσο κάθετα όσο και οριζόντια) στην ίδια οθόνη.




Σχήμα 3.3: Αρχική Σελίδα

Για τη βελτιστοποίηση της εμπειρίας του χρήστη, ενσωματώνονται μηχανισμοί φόρτωσης (loaders) έως ότου ολοκληρωθεί η ανάκτηση και απεικόνιση των δεδομένων. Επιπλέον, ακολουθείται διαδικασία επαλήθευσης της διαθεσιμότητας των URLs των εικόνων μετά την ανάκτησή τους. Σε περιπτώσεις που κάποια εικόνα δεν μπορεί να προβληθεί επιτυχώς, εμφανίζεται προκαθορισμένο placeholder, προκειμένου να διατηρηθεί η αισθητική συνοχή και η ευχρηστία της εφαρμογής.

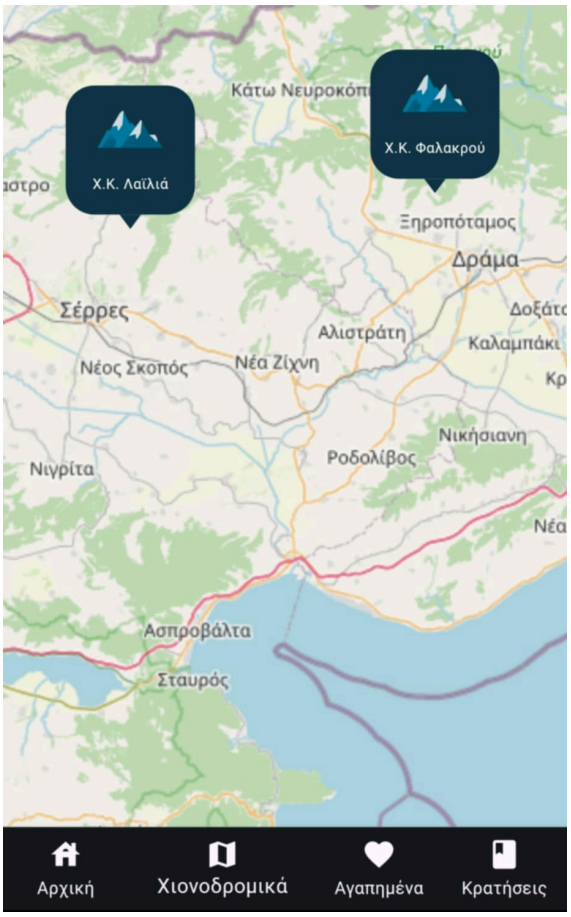
```dart
  Future<List<String>> filterValidImages(List<String> urls) async {
    var validUrls = <String>[];

    for (var url in urls) {
      final response = await http.get(Uri.parse(url));
      if (response.statusCode == 200 &&
          !response.body.contains('<!DOCTYPE html>')) {
        validUrls.add(url);
      } else {
        validUrls.add(
            'https://i.etsystatic.com/21635101/r/il/5409b8/2790397743/il_794xN.2790397743_3gcg.jpg');
      }
    }

    return validUrls;
  }
}
```

Σχήμα 3.4: Μέθοδος filterValidImages

### 3.2.2 Σελίδα Χιονοδρομικά

Στη δεύτερη σελίδα της εφαρμογής, με τίτλο "Χιονοδρομικά", παρουσιάζεται ένας διαδραστικός χάρτης, στον οποίο απεικονίζονται τα χιονοδρομικά κέντρα μέσω προσαρμοσμένων σημάνσεων (custom markers) τοποθετημένων στις αντίστοιχες γεωγραφικές συντεταγμένες τους. Για την υλοποίηση του χάρτη χρησιμοποιήθηκε το πακέτο flutter_map, το οποίο επιλέχθηκε ως λύση λόγω της δυνατότητάς του να λειτουργεί χωρίς ανάγκη σύνδεσης σε διαδικτυακές υπηρεσίες API όπως το Google Maps, προσφέροντας έτσι μεγαλύτερη αυτονομία, μηδενικό κόστος χρήσης και μεγαλύτερο έλεγχο επί των δεδομένων.

Κατά τη μετάβαση του χρήστη στην καρτέλα "Χιονοδρομικά", ενεργοποιείται η απεικόνιση του χάρτη, με κάθε χιονοδρομικό κέντρο να αναπαρίσταται μέσω marker ακριβώς στη θέση των γεωγραφικών του συντεταγμένων. Η χρήση προσαρμοσμένων markers επιτρέπει την καλύτερη οπτική διάκριση και προσφέρει μια πιο εστιασμένη και ευχάριστη εμπειρία πλοήγησης για τον χρήστη.




Σχήμα 3.5: Σελίδα Χάρτη

```
return FlutterMap(
  options: const MapOptions(
    interactionOptions: InteractionOptions(
        flags: InteractiveFlag.pinchZoom | InteractiveFlag.drag),
    keepAlive: false,
    initialCenter:
        LatLng(40.629269, 22.947412), // Center the map over London
    initialZoom: 7,
  ),
  children: [
    TileLayer(
      // Display map tiles from any source
      urlTemplate:
          'https://tile.openstreetmap.org/{z}/{x}/{y}.png', // OSMF'
      userAgentPackageName: 'com.example.app',
      // And many more recommended properties!
    ),
    MarkerLayer(
      markers: [
        ...resorts.map(
          (resort) => resortMarker(resort, context),
        ),
      ],
    ),
  ],
);
});
```

Σχήμα 3.6: FlutterMap χρήση

Για τα στιγματα των χιονοδρομικων δημιουργηθηκε ενα custom widget resortMarker. Το resortMarker δεχεται ως παραμετρο ενα αντικειμενο resort το οποιο ειναι το αντικειμενο που χρησιμοποιειται σε ολη την εφαρμογη και περιεχη τα στοιχεια ενος χιονοδρομικου οπως ονομα, συντεταγμενες, αριθμο lifts κτλ. Το resortMarker αποτελειται απο μια στηλη οπου περιεχει ενα container με μια εμφωλευμενη στηλη που περιεχει μια εικονα και το ονομα του χιονοδρομικου, κατω απο το container δημιουργηθηκε με custom painter ενα τριγωνο για να φαινεται καλυτερα στον χαρτη που βρισκεται το χιονοδρομικο. Το marker αν επιλεχθει απο τον χρηστη θα μετακινηθει στην σελιδα του χιονοδρομικου που περιεχει αναλυτικες πληροφοριες.


Σχήμα 3.7: resortMarker

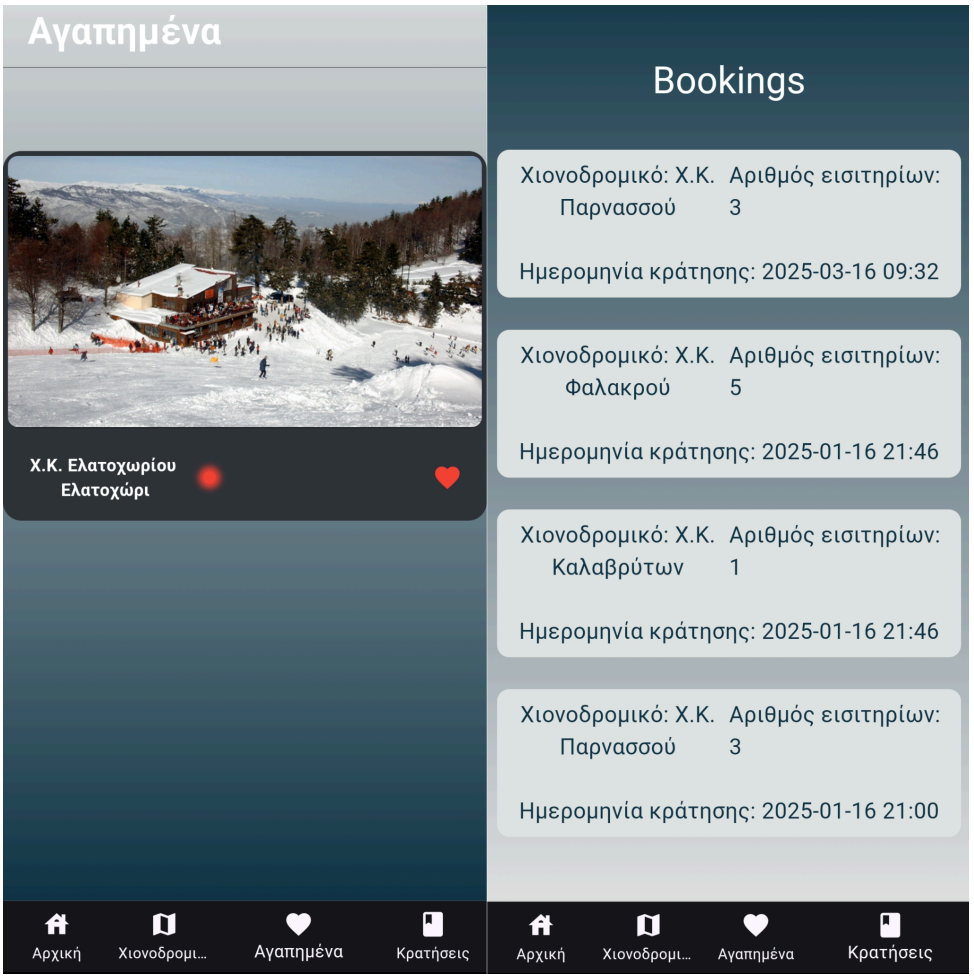
### 3.2.3 Σελίδες Αγαπημένα και Κρατήσεις

Οι σελίδες "Αγαπημένα" και "Κρατήσεις" ακολουθούν κοινή δομή και σχεδιαστική φιλοσοφία, βασισμένες στην απλή και λειτουργική παρουσίαση λίστας ομοιογενών στοιχείων. Συγκεκριμένα, κάθε σελίδα αποτελείται από μία κατακόρυφη λίστα, όπου κάθε στοιχείο της αναπαρίσταται από ένα συγκεκριμένο τύπο widget.

Στην περίπτωση της σελίδας "Αγαπημένα", η λίστα περιλαμβάνει widgets που αναπαριστούν χιονοδρομικά κέντρα, τα οποία ο χρήστης έχει αποθηκεύσει ως αγαπημένα για ευκολότερη και ταχύτερη πρόσβαση. Εφόσον ο χρήστης δεν έχει αποθηκεύσει κάποιο αγαπημένο ή δεν είναι συνδεδεμένος στον λογαριασμό του, προβάλλεται ένα κατάλληλο μήνυμα, ενημερώνοντάς τον για την έλλειψη αγαπημένων ή την ανάγκη σύνδεσης.

Αντίστοιχα, στη σελίδα "Κρατήσεις", η λίστα αποτελείται από containers που περιέχουν πληροφορίες για κάθε κράτηση του χρήστη. Κάθε εγγραφή περιλαμβάνει το όνομα του χιονοδρομικού κέντρου, τον αριθμό των εισιτηρίων που έχουν αγοραστεί καθώς και την ημερομηνία πραγματοποίησης της κράτησης. Αυτή η διάρθρωση προσφέρει μια συνεπή και φιλική προς τον χρήστη εμπειρία πλοήγησης,

ενώ ταυτόχρονα διατηρεί την εφαρμογή ευέλικτη και επεκτάσιμη σε μελλοντικές προσθήκες λειτουργιών.




Σχήμα 3.8: Αγαπημένα και Κρατήσεις

Για την αποθήκευση και ανάκτηση των πληροφοριών που σχετίζονται με τα "Αγαπημένα" και τις "Κρατήσεις" χρησιμοποιείται η υπηρεσία Firebase Firestore. Η ανάκτηση των δεδομένων πραγματοποιείται μέσω της χρήσης του μοναδικού αναγνωριστικού του χρήστη (userId), το οποίο αντλείται μετά τη διαδικασία ταυτοποίησης (authentication). Το userId χρησιμοποιείται για την εκκίνηση ενός Query προς το Firestore, το οποίο φιλτράρει και επιστρέφει μόνο τα δεδομένα που σχετίζονται με τον συγκεκριμένο χρήστη.

```
FirebaseAuth auth = FirebaseAuth.instance;
CollectionReference favCollection =
    FirebaseFirestore.instance.collection('favourites');
Query<Object?> favQuery =
    favCollection.where('userId', isEqualTo: auth.currentUser!.uid);
```

Σχήμα 3.9: Query αγαπημένων

Το ερώτημα (query) που δημιουργείται βάσει του userId ενσωματώνεται μέσα σε έναν StreamBuilder. Ο StreamBuilder παρακολουθεί τη ροή δεδομένων (stream) που επιστρέφεται από το Firestore και, ανάλογα με την τρέχουσα κατάσταση των δεδομένων, ανακατασκευάζει δυναμικά τη διεπαφή χρήστη (UI).

Με αυτόν τον τρόπο, η απεικόνιση των στοιχείων στην οθόνη προσαρμόζεται σε πραγματικό χρόνο, αντικατοπτρίζοντας άμεσα οποιαδήποτε αλλαγή (προσθήκη, διαγραφή ή ενημέρωση) πραγματοποιείται στη βάση δεδομένων. Αυτή η προσέγγιση ενισχύει σημαντικά την εμπειρία του χρήστη, εξασφαλίζοντας ότι οι πληροφορίες που προβάλλονται είναι πάντοτε επικαιροποιημένες, χωρίς την ανάγκη χειροκίνητης ανανέωσης της σελίδας.

Η χρήση του StreamBuilder σε συνδυασμό με τα real-time χαρακτηριστικά του Firebase Firestore αποτελεί βέλτιστη πρακτική για εφαρμογές που απαιτούν άμεση ανταπόκριση στις μεταβολές των δεδομένων, προσφέροντας υψηλή διαδραστικότητα και αίσθηση αμεσότητας.

```
StreamBuilder<QuerySnapshot>(
  stream: favQuery.snapshots(),
  builder: (context, snapshot) {
    if (!snapshot.hasData) {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }
    var resorts = snapshot.data!.docs
        .expand((doc) => (doc['resorts'] as Map).keys);
    for (var resort in state.resorts) {
      if (resorts.contains(resort.id.toString())) {
        favResorts.add(resort);
      }
    }

    if (resorts.isEmpty) {
      return const ListTile(
        title: Text('Δεν έχετε αγαπημένα.'),
      );
    }

    return CustomScrollView(
      slivers: <Widget>[
        SliverList(
          delegate: SliverChildBuilderDelegate(
            (context, index) {
              return Padding(
                padding: const EdgeInsets.all(16.0),
                child: ResortsCardWidget(
                    resort: favResorts[index],
                    id: favResorts[index].id),
              );
            },
            childCount: resorts.length,
          ),
        ),
      ],
    );
  },
```

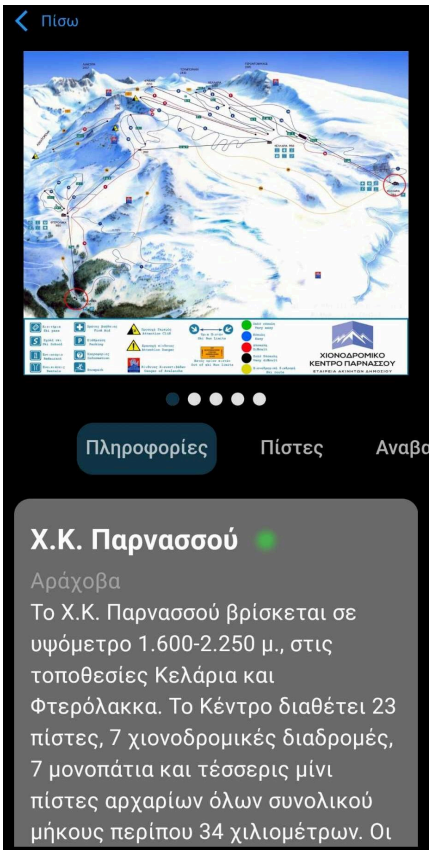
Σχήμα 3.10: StreamBuilder

### 3.2.4 Σελίδα Χιονοδρομικού

Η σελίδα χιονοδρομικού αποτελεί τη πλέον σύνθετη και λειτουργικά πλούσια ενότητα της εφαρμογής. Η αυξημένη πολυπλοκότητά της δικαιολογείται από το γεγονός ότι πρόκειται για το σημείο με το υψηλότερο επίπεδο ενδιαφέροντος για τον χρήστη, καθώς συγκεντρώνει το σύνολο των πληροφοριών που σχετίζονται με το χιονοδρομικό κέντρο που έχει επιλέξει.

Ο σχεδιασμός της σελίδας ακολουθεί μια κατά τμήματα οργάνωση των πληροφοριών, διαχωρίζοντας σαφώς τις διάφορες θεματικές ενότητες. Αυτή η αρχιτεκτονική επιτρέπει την ευκολότερη πλοήγηση και κατανόηση του περιεχομένου, συμβάλλοντας καθοριστικά στη βελτιστοποίηση της εμπειρίας του χρήστη (UX). Ενδεικτικά, περιλαμβάνονται πληροφορίες όπως περιγραφή του χιονοδρομικού, καιρικές συνθήκες, φωτογραφικό υλικό, ποιότητα χιονιού στα επίπεδα του βουνού, δυνατότητα κράτησης εισιτηρίων, διαθέσιμες δραστηριότητες, οδηγίες στο google maps, κατάσταση αναβατήρων και πιστών, καθώς και κάμερες ζωντανής μετάδοσης από το χιονοδρομικό.

Η σελίδα υλοποιείται με τρόπο που να διαχειρίζεται αποδοτικά τόσο τον όγκο όσο και τη διαφορετικότητα των δεδομένων, διατηρώντας παράλληλα υψηλά πρότυπα απόδοσης και διαδραστικότητας.




Σχήμα 3.11 Σελίδα Χιονοδρομικού

## 3.3 Προγραμματιστική υλοποίηση

Σε αυτή την ενότητα παρουσιάζεται η προγραμματιστική υλοποίηση της εφαρμογής, με έμφαση στις τεχνικές επιλογές, τις δομές δεδομένων, την αρχιτεκτονική και τις πρακτικές ανάπτυξης που υιοθετήθηκαν. Περιγράφονται αναλυτικά τα βασικά επιμέρους τμήματα της εφαρμογής, όπως η διαχείριση της κατάστασης (state management), η επικοινωνία με εξωτερικές υπηρεσίες (όπως το Firebase), η δυναμική φόρτωση περιεχομένου, και η αλληλεπίδραση του χρήστη με τη διεπαφή. Η ενότητα αποσκοπεί στο να αποτυπώσει τον τρόπο με τον οποίο μεταφράστηκε ο σχεδιασμός και η ανάλυση απαιτήσεων σε λειτουργικό και αποδοτικό λογισμικό, χρησιμοποιώντας το πλαίσιο ανάπτυξης Flutter και γλώσσα Dart.

### 3.3.1 State Management

Για τη διαχείριση της κατάστασης της εφαρμογής χρησιμοποιήθηκαν δύο προσεγγίσεις, ανάλογα με τη φύση και τη δυναμική των δεδομένων: αφενός, αξιοποιήθηκαν τα βασικά Stateful και Stateless widgets που προσφέρει εγγενώς το Flutter, και αφετέρου, για πιο σύνθετα σενάρια με αυξημένες απαιτήσεις σε αλληλεπιδράσεις και ανανέωση δεδομένων, χρησιμοποιήθηκε το πρότυπο BLoC (Business Logic Component), το οποίο επιτρέπει τον διαχωρισμό της επιχειρησιακής λογικής από τη διεπαφή χρήστη.

Ως παράδειγμα StatelessWidget παρουσιάζεται το ActivityCapsule, ένα επαναχρησιμοποιήσιμο γραφικό στοιχείο (widget), το οποίο έχει ως στόχο να απεικονίζει με συνεπή και απλή μορφή τις δραστηριότητες ενός χιονοδρομικού κέντρου. Η επιλογή υλοποίησης ως StatelessWidget προκύπτει από το γεγονός ότι το συγκεκριμένο στοιχείο δεν διατηρεί εσωτερική κατάσταση και η ανανέωσή του απαιτεί πλήρη επανασχεδίαση (rebuild) από το ανώτερο επίπεδο. Τέτοιου τύπου widgets χρησιμοποιούνται κατά κύριο λόγο για σκοπούς απεικόνισης σταθερού περιεχομένου και δεν ενσωματώνουν σύνθετη λογική ή χειρισμό γεγονότων.

Όπως φαίνεται στο Σχήμα 3.12, το ActivityCapsule περιλαμβάνει μια εικόνα και έναν τίτλο, διαμορφώνοντας οπτικά την εκάστοτε δραστηριότητα σε μορφή "κάψουλας". Η απαιτούμενη πληροφορία εισάγεται μέσω της παραμέτρου activity, η οποία είναι μεταβλητή τύπου String και αντιπροσωπεύει την ονομασία της δραστηριότητας. Στο εσωτερικό του widget ορίζονται δύο βοηθητικές μέθοδοι τύπου String, οι getActivityIcon και getActivityTitle, οι οποίες λαμβάνουν ως είσοδο την τιμή της μεταβλητής activity και επιστρέφουν αντίστοιχα το path της εικόνας και τον μορφοποιημένο τίτλο της δραστηριότητας, όπως φαίνεται και στο Σχήμα 3.13.


Σχήμα 3.12 Activity Capsule

```
String getActivityIcon(String activity) {
  switch (activity) {
    case 'σαλε':
      return 'assets/cabin-wooden-cabin-cottage-svgrepo-com.svg';
    case 'Σκι':
      return 'assets/ski-svgrepo-com.svg';
    case 'σνοουμπορντ':
      return 'assets/snowboard-silhouette-svgrepo-com.svg';
    case 'έλκηθρο':
      return 'assets/sled-svgrepo-com.svg';
    case 'πεζοπορια':
      return 'assets/hiking-svgrepo-com.svg';
    case 'snowmobile':
      return 'assets/snowmobile-svgrepo-com.svg';
    default:
      return 'assets/cabin-wooden-cabin-cottage-svgrepo-com.svg';
  }
}

String getActivityTitle(String activity) {
  switch (activity) {
    case 'σαλε':
      return 'Σαλέ';
    case 'Σκι':
      return 'Σκι';
    case 'σνοουμπορντ':
      return 'Σνόουμπορντ';
    case 'έλκηθρο':
      return 'Έλκηθρο';
    case 'πεζοπορια':
      return 'Πεζοπορία';
    case 'snowmobile':
      return 'Snowmobile';
    default:
      return '-';
  }
}
```

Σχήμα 3.13 Activity Capsule Μέθοδοι

Αντίθετα με τα StatelessWidget, τα StatefulWidget παρέχουν τη δυνατότητα δυναμικής ανανέωσης της διεπαφής χρήστη (UI) όταν μεταβάλλονται συγκεκριμένα δεδομένα ή μετά από ρητή εντολή του συστήματος ή του χρήστη. Ουσιαστικά, τα StatefulWidget συνδέονται με μια κατάσταση (state), η οποία διατηρείται κατά τη διάρκεια ζωής του widget και μπορεί να τροποποιηθεί μέσω της μεθόδου setState(), προκαλώντας επανασχεδίαση (rebuild) του widget και ανανέωση του περιεχομένου της οθόνης.

Τα widgets αυτού του τύπου χρησιμοποιούνται κυρίως σε περιπτώσεις που απαιτείται διαχείριση εσωτερικής λογικής ή αντίδραση σε αλληλεπιδράσεις του χρήστη, όπως για παράδειγμα εισαγωγή δεδομένων, κύλιση λιστών, φόρτωση περιεχομένου ή μετάβαση μεταξύ καταστάσεων. Λόγω της αυξημένης ευελιξίας τους, ενδείκνυνται για πιο σύνθετα λειτουργικά σενάρια, προσφέροντας μεγαλύτερο έλεγχο στην ενημέρωση του UI σε πραγματικό χρόνο.

Ένα χαρακτηριστικό παράδειγμα χρήσης StatefulWidget στην παρούσα εφαρμογή αποτελεί το ResortPageCameraTab, το οποίο ενσωματώνει ένα παράθυρο αναπαραγωγής πολυμέσων (media player) που μεταδίδει ζωντανή εικόνα από κάμερα τοποθετημένη στο εκάστοτε χιονοδρομικό κέντρο. Η δυναμική φύση του περιεχομένου καθιστά αναγκαία τη χρήση StatefulWidget, προκειμένου να διασφαλίζεται η συνεχής ενημέρωση της διεπαφής και η σωστή διαχείριση του κύκλου ζωής του media player.

Κατά την εκκίνηση του widget καλείται η μέθοδος initState(), η οποία δίνει τη δυνατότητα εκτέλεσης εντολών πριν το πρώτο rendering του widget. Σε αυτό το στάδιο, αρχικοποιείται ο controller που διαχειρίζεται την αναπαραγωγή της ροής βίντεο και προστίθεται σε αυτόν ένας listener. Ο listener αυτός παρακολουθεί μεταβολές στην κατάσταση του player και, όποτε ανιχνεύει αλλαγές, καλεί τη μέθοδο setState(), ενεργοποιώντας τον επανασχεδιασμό του widget (rebuild) και επιτυγχάνοντας έτσι άμεση ενημέρωση του περιβάλλοντος χρήστη.

Για την αποφυγή διαρροών μνήμης (memory leaks), είναι κρίσιμο να γίνεται σωστή χρήση της μεθόδου dispose(), η οποία καλείται όταν το widget παύει να είναι ενεργό. Στην dispose(), ο controller τερματίζεται και οι πόροι που χρησιμοποιεί αποδεσμεύονται. Αν παραλειφθεί αυτό το βήμα, κάθε επαναφόρτωση του widget θα δημιουργεί νέο controller χωρίς να καταστρέφει τον προηγούμενο, με αποτέλεσμα την κατασπατάληση πόρων και την πιθανή αποσταθεροποίηση της εφαρμογής.

```dart
class ResortPageCamerasTab extends StatefulWidget {
  const ResortPageCamerasTab({super.key});

  @override
  State<ResortPageCamerasTab> createState() => _ResortPageCamerasTabState();
}

class _ResortPageCamerasTabState extends State<ResortPageCamerasTab> {
  late VideoPlayerController _controller;
  final Uri videoUri =
      Uri.parse('https://pluto13.cybex.gr/appl/meteocam/vod/mc23cx144936.mp4');

  @override
  void initState() {
    super.initState();

    _controller = VideoPlayerController.networkUrl(
      videoUri,
      videoPlayerOptions: VideoPlayerOptions(mixWithOthers: true),
    );

    _controller.addListener(() {
      setState(() {});
    });
    _controller.setLooping(true);
    _controller.initialize();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
```

Σχήμα 3.14 ResortPageCameraTab

Τέλος, η προσέγγιση BLoC (Business Logic Component) διαφοροποιείται σημαντικά από τις βασικές μεθόδους διαχείρισης κατάστασης, καθώς στοχεύει στον σαφή διαχωρισμό της λογικής της εφαρμογής από τη διεπαφή χρήστη. Η δομή αυτή προάγει την αρχιτεκτονική “separation of concerns”, διευκολύνοντας την επεκτασιμότητα, την επαναχρησιμοποίηση κώδικα και τη συντήρηση.

Στο πλαίσιο του BLoC, το UI παραμένει καθαρά δηλωτικό και υλοποιείται συχνά ως StatelessWidget, ενώ η λογική διαχείρισης της κατάστασης χωρίζεται σε δύο επιμέρους αρχεία:

- Το αρχείο state, το οποίο περιέχει τις μεταβλητές που επηρεάζουν την κατάσταση της διεπαφής (π.χ. λίστες, flags, δείκτες κατάστασης).
- Το αρχείο cubit, το οποίο περιλαμβάνει τις μεθόδους που τροποποιούν τις μεταβλητές του state και προκαλούν την ενημέρωση της διεπαφής μέσω εκπομπής νέων καταστάσεων.

Ένα αντιπροσωπευτικό παράδειγμα χρήσης της προσέγγισης BLoC είναι η σελίδα ResortsPage, όπου προβάλλεται δυναμικά μια λίστα με χιονοδρομικά κέντρα. Όπως απεικονίζεται στο Σχήμα 3.15, αριστερά βρίσκεται το state, το οποίο περιλαμβάνει μια μεταβλητή λίστας τύπου List<CompactResort>, ενώ δεξιά παρουσιάζεται το cubit, όπου η μέθοδος initialize() είναι υπεύθυνη για την ανάκτηση και αποθήκευση των δεδομένων στη λίστα αυτή.

Η διεπαφή, όπως φαίνεται στο Σχήμα 3.16, "ακούει" τις αλλαγές στο state, και με βάση την τρέχουσα τιμή της μεταβλητής resorts, καθορίζει τι εμφανίζεται στην οθόνη. Όταν η λίστα είναι άδεια, εμφανίζεται ένα loader για την ενημέρωση του χρήστη ότι η εφαρμογή βρίσκεται σε διαδικασία φόρτωσης. Με την ολοκλήρωση της μεθόδου initialize() και την πλήρωση της λίστας, η διεπαφή

επανασχεδιάζεται (rebuild) και εμφανίζονται δυναμικά τα widgets με τις πληροφορίες των χιονοδρομικών.

Η συγκεκριμένη προσέγγιση εξασφαλίζει υψηλή ανταπόκριση της διεπαφής, δομημένο κώδικα, και ευκολότερη επεκτασιμότητα σε περιβάλλοντα που απαιτούν σύνθετη λογική ή χειρισμό δεδομένων σε πραγματικό χρόνο.

```dart
part of 'compact_resort_cubit.dart';

class CompactResortState {
  final List<CompactResort> resorts;
  final bool isLoading;
  final String errorMessage;

  CompactResortState({
    required this.resorts,
    required this.isLoading,
    required this.errorMessage,
  });

  factory CompactResortState.initial() {
    return CompactResortState(
      resorts: [],
      isLoading: true,
      errorMessage: '',
    );
  }
```

```dart
part 'compact_resort_state.dart';

class CompactResortCubit extends Cubit<CompactResortState> {
  CompactResortCubit.internal() : super(CompactResortState.initial());

  static final CompactResortCubit _instance = CompactResortCubit.internal();

  static CompactResortCubit get instance => _instance;

  Future<void> initialize() async {
    emit(state.copyWith(isLoading: true));
    // Load resorts
    final List<CompactResort> resorts = await getResorts();
    emit(state.copyWith(resorts: resorts, isLoading: false));
  }

  Future<List<CompactResort>> getResorts() async {
    return await CompactResortRepository.instance.getResorts();
  }

  CompactResort getResortById(int id) {
    return state.resorts.firstWhere((element) => element.id == id);
```

Σχήμα 3.15 State και Cubit

```dart
return BlocConsumer<CompactResortCubit, CompactResortState>(
    bloc: context.read<CompactResortCubit>(),
    listener: (context, state) {},
    builder: (context, state) {
      if (state.resorts.isEmpty) {
        return Container(
          padding: const EdgeInsets.symmetric(vertical: 40),
          decoration: const BoxDecoration(
            gradient: LinearGradient(
              begin: Alignment.topCenter,
              end: Alignment.bottomCenter,
              colors: [AppColors.snowhubWhite, AppColors.snowhubBlue],
            ),
          ),
          width: double.infinity,
          height: double.infinity,
          child: const Center(
            child: SpinKitSpinningLines(
              color: AppColors.snowhubBlue,
            ),
          ),
        );
      }
      List<CompactResort> resorts = state.resorts;
      return Container(
        padding: const EdgeInsets.symmetric(vertical: 40),
        decoration: const BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [AppColors.snowhubWhite, AppColors.snowhubBlue],
```

Σχήμα 3.16 BlocConsumer

### 3.3.2 Αντικείμενα για την οργάνωση των πληροφοριών

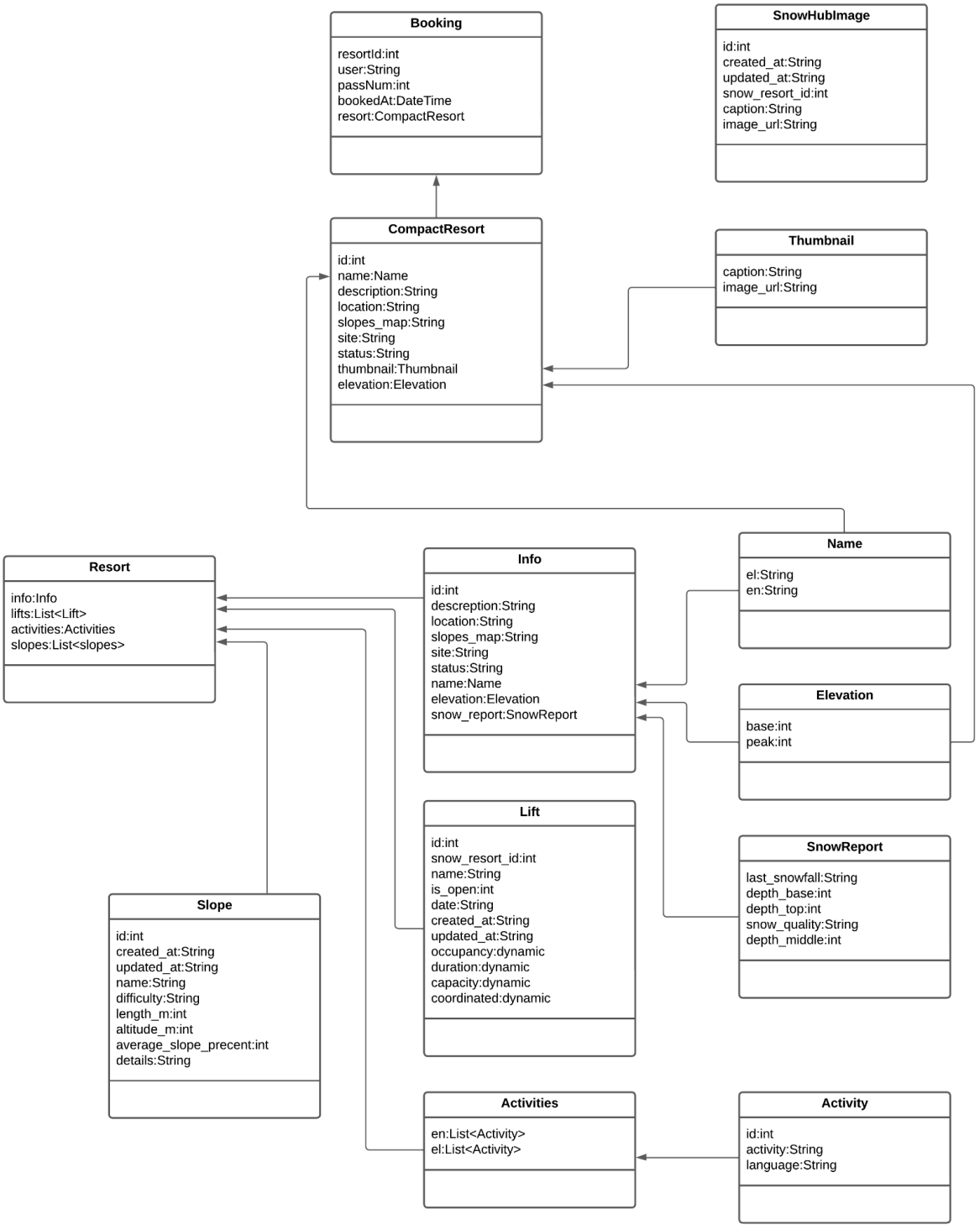
Η ανάπτυξη της εφαρμογής βασίστηκε σε αντικειμενοστραφή σχεδίαση, με στόχο τη σαφή οριοθέτηση και οργάνωση των δεδομένων που διαχειρίζεται το σύστημα. Τα αντικείμενα (data models) αποτελούν τον πυρήνα της εσωτερικής αναπαράστασης των πληροφοριών και χρησιμοποιούνται για την αποθήκευση, μεταφορά και απόδοσή τους τόσο στο backend όσο και στο frontend περιβάλλον της εφαρμογής.

Στο παρόν κεφάλαιο παρουσιάζονται αναλυτικά τα αντικείμενα που σχεδιάστηκαν και υλοποιήθηκαν στο πλαίσιο της εφαρμογής, όπως αυτά αποτυπώνονται και στο διάγραμμα κλάσεων (UML). Κάθε αντικείμενο μοντελοποιεί μία συγκεκριμένη οντότητα, όπως για παράδειγμα ένα χιονοδρομικό κέντρο, μια κρατηση, μια δραστηριότητα ή μια εικόνα από κάμερα. Η χρήση τέτοιων δομών επιτρέπει την ευέλικτη διαχείριση δεδομένων, την εύκολη επεκτασιμότητα της εφαρμογής και τη διατήρηση καθαρού και οργανωμένου κώδικα.

Συνοπτικά, στο διάγραμμα περιλαμβάνονται οι παρακάτω βασικές οντότητες:

- Resort: Το πλήρες αντικείμενο που περιέχει όλες τις λεπτομέρειες ενός χιονοδρομικού.
- CompactResort: Συνοπτική μορφή του resort, βελτιστοποιημένη για απεικόνιση σε λίστες.
- Booking: Αναπαριστά τις κρατήσεις που πραγματοποιεί ο χρήστης.
- Lift, Slope, Activities, Activity: Υποστηρικτικές οντότητες που περιγράφουν τις εγκαταστάσεις και δραστηριότητες κάθε χιονοδρομικού.
- SnowReport: Αντικείμενο που σχετίζεται με τις καιρικές συνθήκες και την κάλυψη χιονιού.
- Info, Name, Elevation: Επικουρικές κλάσεις που προσφέρουν επιπλέον πληροφορίες για την απεικόνιση και περιγραφή των χιονοδρομικών.
- Thumbnail: Υποστηρικτικό αντικείμενο για την εικόνα των χιονοδρομικών όταν εμφανίζονται ως CompactResort.

Κάθε μία από τις παραπάνω κλάσεις περιλαμβάνει μεταβλητές με τύπους δεδομένων κατάλληλους για το περιεχόμενό τους (π.χ. String, int, DateTime, List, dynamic), οι οποίες είτε προέρχονται από την βάση δεδομένων με API κλήσεις είτε από την Firestore είτε δημιουργούνται εσωτερικά στην εφαρμογή.




Σχήμα 3.17 Class Diagram

### 3.3.3 API κλήσεις και Firebase

Για τη λειτουργικότητα της εφαρμογής απαιτήθηκε η αλληλεπίδραση με εξωτερικές πηγές δεδομένων μέσω API κλήσεων, καθώς και η ενσωμάτωση υπηρεσιών του Firebase για την υποστήριξη της αυθεντικοποίησης χρηστών και της αποθήκευσης προσωποποιημένων δεδομένων.

Αρχικά, όσον αφορά τα API endpoints, η εφαρμογή επικοινωνεί με ένα RESTful backend που παρέχει πρόσβαση σε δεδομένα που σχετίζονται με τα χιονοδρομικά κέντρα. Συγκεκριμένα, αξιοποιούνται τρία βασικά endpoints:

1. Κατάλογος Χιονοδρομικών (Resorts): Επιστρέφει τη λίστα όλων των διαθέσιμων χιονοδρομικών, με βασικές πληροφορίες για καθένα.
2. Εικόνες Χιονοδρομικών: Παρέχει τις εικόνες των χιονοδρομικών, οι οποίες προβάλονται για αισθητικούς λόγους στην αρχική οθόνη.
3. Λεπτομέρειες Χιονοδρομικού: Επιστρέφει αναλυτικά στοιχεία για ένα επιλεγμένο χιονοδρομικό, όπως πίστες, ανελκυστήρες, και δραστηριότητες.

Τα δεδομένα αυτά ενσωματώνονται δυναμικά στην εφαρμογή και αποτελούν τον βασικό κορμό πληροφορίας που εμφανίζεται στον τελικό χρήστη.

Παράλληλα, για την κάλυψη λειτουργιών που απαιτούν προσωποποιημένη εμπειρία χρήστη, χρησιμοποιήθηκαν υπηρεσίες της πλατφόρμας Firebase. Συγκεκριμένα:

- Το Firebase Authentication αξιοποιήθηκε για την ταυτοποίηση των χρηστών μέσω email και κωδικού πρόσβασης.
- Το Cloud Firestore χρησιμοποιήθηκε για την αποθήκευση δεδομένων που σχετίζονται με τις κρατήσεις και τα αγαπημένα χιονοδρομικά του κάθε χρήστη.

Αξίζει να σημειωθεί πως, οι λειτουργίες κρατήσεων και αγαπημένων αποτελούσαν μέρος των αρχικών προδιαγραφών του έργου, αλλά δεν υποστηρίχθηκαν από το βασικό backend που υλοποιήθηκε από συνάδελφο Ξενοφών. Ο λόγος ήταν η προσαρμογή της υπηρεσίας backend σε πραγματικές απαιτήσεις αγοράς. Για τη διατήρηση της πληρότητας και της λειτουργικότητας της εφαρμογής, η παρούσα υλοποίηση εμπλούτισε το σύστημα με Firebase ως εναλλακτική λύση για τα συγκεκριμένα χαρακτηριστικά.

Η συνδυαστική χρήση εξωτερικών API και της πλατφόρμας Firebase επιτρέπει στην εφαρμογή να προσφέρει μια ολοκληρωμένη εμπειρία, εξισορροπώντας την αξιοπιστία σταθερών δεδομένων με την ευελιξία δυναμικών, προσωποποιημένων λειτουργιών.

Περιγραφή των API Endpoints

Η επικοινωνία της εφαρμογής με τον απομακρυσμένο διακομιστή (backend) πραγματοποιείται μέσω συγκεκριμένων RESTful API endpoints, τα οποία είναι ορισμένα στην κλάση ApiConst. Ορίζεται αρχικά μια βασική διεύθυνση (base URL) και κατόπιν καθορίζονται τα επιμέρους endpoints που χρησιμοποιούνται για την ανάκτηση των δεδομένων.

```dart
class ApiConst {
  static String baseUrl = 'https://www.snowhub.gr/api/';
  static String resorts = '${baseUrl}SnowResorts';
  static String snowhubImages = '${baseUrl}images';
  static String resort = '${baseUrl}SnowResort/';
}
```

Σχήμα 3.18 Κλάση ApiConst

Τα endpoints που χρησιμοποιούνται είναι τα εξής:

- SnowResorts Endpoint (https://www.snowhub.gr/api/SnowResorts)

  Μέσω αυτού του endpoint γίνεται η ανάκτηση της λίστας όλων των διαθέσιμων χιονοδρομικών κέντρων. Η απάντηση του server επιστρέφει τα χιονοδρομικά σε συνοπτική μορφή (compact), περιλαμβάνοντας βασικές πληροφορίες όπως το όνομα, η τοποθεσία και η κατάσταση λειτουργίας κάθε χιονοδρομικού.

| | |
|---|---|
| **id** | 1 |
| **description** | Το Χ.Κ. Παρνασσού βρίσκεται σε υψόμετρο 1.600-2.250 μ., στις τοποθεσίες Κελάρια και Φτερόλακκα. Το Κέντρο διαθέτει 23 πίστες, 7 χιονοδρομικές διαδρομές, 7 μονοπάτια και τέσσερις μίνι πίστες αρχαρίων όλων συνολικού μήκους περίπου 34 χιλιομέτρων. Οι πίστες είναι μήκους κατάβασης από 50 μ. έως 4 χλμ. για αρχαρίους, μέσους και καλούς χιονοδρόμους. Για τους λάτρεις της περιπέτειας, υπάρχουν 12 εκτός πίστας "μαύρες" διαδρομές. Λειτουργεί από Δεκέμβριο έως αρχές Μαΐου. Βρίσκεται πολύ κοντά στην Αθήνα, την Λαμία και την Πάτρα και οι εγκαταστάσεις, του επιτρέπουν την εξυπηρέτηση μεγάλου αριθμού επισκεπτών και χιονοδρόμων. Στο Κέντρο λειτουργούν δύο καφετέριες-chalet και εστιατόριο, σχολές εκμάθησης σκι και snowboard, καταστήματα ενοικίασης χιονοδρομικού εξοπλισμού, καθώς και υπηρεσία φύλαξης με παιδική χαρά για τους μικρούς επισκέπτες. Στη θέση Κελάρια 1750 κ στη θέση Κελάρια 1950 (chalet) , λειτουργεί κατάστημα συντήρησης για σκι και snowboard, με τα πλέον εξελιγμένα τεχνολογικά μηχανήματα. |
| **location** | Αράχοβα |
| **slopes_map** | 38.55874296502439, 22.589506461326437 |
| **site** | https://parnassos-ski.gr/ |
| **status** | Closed |
| **facebook_name** | parnassosski.gr |
| **stream_url** | mc219cx1328580.m3u8 |
| **thumbnail** {2} | |
| **name** {2} | |
| **elevation** {2} | |

Σχήμα 3.19 Παράδειγμα Απάντησης SnowResorts Endpoint

Η αρχική κλήση πραγματοποιείται εντός της μεθόδου main() της εφαρμογής, η οποία εκτελείται αυτόματα κατά την εκκίνησή της. Η επιλογή να εκτελεστεί το αίτημα σε αυτό το σημείο οφείλεται στο γεγονός ότι τα δεδομένα που ανακτώνται σε αυτό το στάδιο — όπως η γενική λίστα των χιονοδρομικών κέντρων και οι συνοδευτικές εικόνες — είναι σχετικά σταθερά και δεν παρουσιάζουν συχνές μεταβολές.

Κατά συνέπεια, η κλήση υλοποιείται μία φορά, αποφεύγοντας την περιττή κατανάλωση πόρων και την επιβάρυνση του δικτύου. Αντίθετα, πληροφορίες που ενδέχεται να μεταβάλλονται συχνότερα (όπως η διαθεσιμότητα ανελκυστήρων, οι καιρικές συνθήκες, ή το χιόνι σε πραγματικό χρόνο) ανακτώνται δυναμικά και κατά περίπτωση, κάθε φορά που ο

χρήστης επιλέγει να μεταβεί σε μια σελίδα συγκεκριμένου χιονοδρομικού. Η προσέγγιση αυτή ενισχύει τη βελτιστοποίηση της απόδοσης της εφαρμογής και εξασφαλίζει ότι οι χρήστες λαμβάνουν επίκαιρες και ακριβείς πληροφορίες μόνο όταν αυτές είναι απαραίτητες.

```
void main() async {
  Http().init();
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(
    MultiBlocProvider(
      providers: [
        BlocProvider(
            create: (context) => CompactResortCubit.instance..initialize()),
        BlocProvider(
            create: (context) => SnowhubImageCubit.instance..initialize()),
      ],
      child: const MyApp(),
    ),
  );
}
```

Σχήμα 3.20 Initialization

```
class ResortCubit extends Cubit<ResortState> {
  ResortCubit.internal() : super(ResortState.initial());

  static final ResortCubit _instance = ResortCubit.internal();

  static ResortCubit get instance => _instance;

  Future<void> initialize(int id) async {
    emit(state.copyWith(isLoading: true));
    // Load resort
    final Resort resort = await getResort(id);
    emit(state.copyWith(resort: resort, isLoading: false));
  }

  Future<Resort> getResort(int id) async {
    return await ResortRepository.instance.getResort(id);
  }
}
```

Σχήμα 3.21 getResort

- Images Endpoint (https://www.snowhub.gr/api/images)

  Επιστρέφει τις εικόνες όλων των χιονοδρομικών κέντρων που φιλοξενούνται στο Snowhub. Κάθε εγγραφή περιλαμβάνει το id του χιονοδρομικού στο οποίο αντιστοιχεί η εικόνα, την περιγραφή της, καθώς και το URL της εικόνας. Οι εικόνες αυτές χρησιμοποιούνται για την οπτική αναπαράσταση των χιονοδρομικών, τόσο στη λίστα όσο και στις λεπτομέρειες.

| | id | created_at | updated_at | snow_resort_id | caption | image_url |
|---|---|---|---|---|---|---|
| 0 | 1 | null | null | 1 | map | https://parnassos-ski.gr/wp-content/uploads/2022/01/Map_Parnassos-2022-scaled.jpg |
| 1 | 2 | null | null | 2 | thumbnail | https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT5FN6VMD98vZ3p7okawayeRigOs4I_NXJ79w&s |
| 2 | 4 | null | null | 4 | thumbnail | https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRkq6HoAw3sX90m-6nSYa6vs8bLSsvxircPRA&s |
| 3 | 5 | null | null | 5 | thumbnail | https://seli-ski.gr/wp-content/uploads/2018/10/2016-01-21-15.55.03-300x225.jpg |
| 4 | 6 | null | null | 6 | thumbnail | https://st2.depositphotos.com/2793427/6202/i/950/depositphotos_62022871-stock-photo-aerial-view-of-falakro-ski.jpg |
| 5 | 7 | null | null | 7 | thumbnail | https://pieriaguide.gr/wp-content/uploads/2017/10/ski-sale.jpg |
| 6 | 8 | null | null | 8 | thumbnail | https://findayhotel.gr/wp-content/uploads/2017/10/Ski-center-Helmos.jpg |
| 7 | 9 | null | null | 9 | thumbnail | https://35pigadia.com/wp-content/uploads/2023/03/Pigadia-2023-5.jpg |
| 8 | 10 | null | null | 9 | view | https://35pigadia.com/wp-content/uploads/2023/03/Pigadia-2023-1.jpg |
| 9 | 11 | null | null | 9 | view | https://35pigadia.com/wp-content/uploads/2023/03/Pigadia-2023-14.jpg |
| 10 | 12 | null | null | 9 | view | https://35pigadia.com/wp-content/uploads/2023/03/Pigadia-2023-10.jpg |
| 11 | 13 | null | null | 9 | view | https://35pigadia.com/wp-content/uploads/2023/03/Pigadia-2023-6.jpg |
| 12 | 14 | null | null | 1 | thumbnail | https://parnassos-ski.gr/wp-content/uploads/2022/01/parnass3.jpg |

Σχήμα 3.22 Παράδειγμα Απάντησης images Endpoint

- SnowResort Details Endpoint (https://www.snowhub.gr/api/SnowResort/{id})

  Το endpoint αυτό χρησιμοποιείται για την ανάκτηση όλων των αναλυτικών πληροφοριών ενός συγκεκριμένου χιονοδρομικού, με βάση το μοναδικό του id. Κατά την κλήση προστίθεται το id του επιθυμητού χιονοδρομικού στη διεύθυνση. Η απάντηση περιλαμβάνει πλήρη δεδομένα, όπως λίφτ, πίστες, υψόμετρο, δραστηριότητες, χιονολογικές συνθήκες κ.ά.


Σχήμα 3.23 Παράδειγμα Απάντησης SnowResort Endpoint

Η χρήση των παραπάνω endpoints επιτρέπει την αναλυτική, πολυεπίπεδη απεικόνιση της πληροφορίας, υποστηρίζοντας τόσο τις λίστες όσο και τις λεπτομέρειες κάθε χιονοδρομικού, ενισχύοντας την εμπειρία του τελικού χρήστη.

Για την κάλυψη των προαναφερθεισών λειτουργικών απαιτήσεων, όπως η αυθεντικοποίηση χρηστών και η αποθήκευση εξατομικευμένων δεδομένων (αγαπημένα και κρατήσεις), χρησιμοποιήθηκε η υπηρεσία Firebase της Google.

Προκειμένου να επιτευχθεί η σύνδεση της εφαρμογής με τον εκάστοτε Firebase server, απαιτείται η ενσωμάτωση του αυτόματα παραγόμενου αρχείου firebase_options.dart στον πηγαίο κώδικα της εφαρμογής. Το αρχείο αυτό περιέχει τις παραμέτρους διαμόρφωσης (όπως API keys, project ID, application ID κ.ά.) που είναι απαραίτητες για την ταυτοποίηση της εφαρμογής και τη σύνδεση με τον αντίστοιχο Firebase backend.

Αφού ενσωματωθεί το εν λόγω αρχείο, πραγματοποιείται η αρχικοποίηση της Firebase μέσω της μεθόδου Firebase.initializeApp(), η οποία εκτελείται κατά την εκκίνηση της εφαρμογής, μέσα στη main() μέθοδο (βλ. Σχήμα 3.20). Η αρχικοποίηση αυτή αποτελεί απαραίτητη προϋπόθεση για τη χρήση όλων των επιμέρους υπηρεσιών του Firebase στο υπόλοιπο της εφαρμογής, όπως Firestore, Authentication, και άλλες cloud λειτουργίες.

Για τη διαδικασία αυθεντικοποίησης των χρηστών χρησιμοποιήθηκε η υπηρεσία Firebase Authentication, η οποία παρέχει ένα ασφαλές και επεκτάσιμο πλαίσιο διαχείρισης ταυτοτήτων χρηστών. Στο πλαίσιο της εφαρμογής, υλοποιήθηκε η δυνατότητα σύνδεσης με email και κωδικό πρόσβασης, δίνοντας στον χρήστη τη δυνατότητα δημιουργίας προσωπικού λογαριασμού.

Κατά τη διαδικασία εγγραφής και σύνδεσης, πραγματοποιούνται έλεγχοι εγκυρότητας των δεδομένων εισόδου, όπως το αν το email είναι ήδη καταχωρημένο στο σύστημα και αν ο κωδικός πληροί τις ελάχιστες προϋποθέσεις ασφαλείας (π.χ. μήκος τουλάχιστον 8 χαρακτήρων). Οι έλεγχοι αυτοί διασφαλίζουν την ομαλή και ασφαλή λειτουργία του μηχανισμού αυθεντικοποίησης.

Επιπλέον, έχει υλοποιηθεί η λειτουργικότητα επαναφοράς κωδικού πρόσβασης, μέσω της επιλογής Forgot Password, όπου το σύστημα αποστέλλει αυτόματα email επαναφοράς στον χρήστη μέσω της Firebase, επιτρέποντάς του να θέσει νέο κωδικό.

Σε κάθε περίπτωση, η εφαρμογή διασφαλίζει ότι ο χρήστης ενημερώνεται άμεσα για το αποτέλεσμα των ενεργειών του, μέσω της προβολής μηνυμάτων τύπου Snackbar. Ενδεικτικά, κατά την επιτυχή δημιουργία λογαριασμού εμφανίζεται σχετικό επιβεβαιωτικό μήνυμα, όπως και κατά την αποστολή email επαναφοράς κωδικού ή σε περιπτώσεις αποτυχίας (π.χ. μη έγκυρα στοιχεία ή αποτυχία σύνδεσης), με στόχο την ενίσχυση της διαφάνειας και της ευχρηστίας της εφαρμογής.

```
class _GuestAvatarDialogWidgetState extends State<GuestAvatarDialogWidget> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  late bool hasAccount = true;
  @override
  void initState() {
    // hasAccount = true;
    super.initState();
  }
```

```
onPressed: () async {
  if (!hasAccount) {
    try {
      UserCredential userCredential =
          await _auth.createUserWithEmailAndPassword(
        email: _emailController.text,
        password: _passwordController.text,
      );

      log('User created: ${userCredential.user?.email}');
      Navigator.of(context).pop();
      favCollection.add(
          {'userId': userCredential.user?.uid, 'resorts': {}});
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Ο λογαριασμός δημιουργήθηκε.'),
        ),
      );
    } on FirebaseAuthException catch (e) {
      log('Failed to create user: $e');
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text('Σφάλμα'),
            content: const Text(
                'Υπάρχει ήδη λογαριασμός με αυτό το email.'),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                },
                child: const Text('Κλείσιμο',
                    style:
                        TextStyle(color: AppColors.snowhubBlue)),
              ),
            ],
          );
        },
      );
    }
  } else {
    try {
      UserCredential userCredential =
          await _auth.signInWithEmailAndPassword(
        email: _emailController.text,
        password: _passwordController.text,
      );

      log('User logged in: ${userCredential.user?.email}');
      Navigator.of(context).pop();
    } on FirebaseAuthException catch (e) {
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text('Σφάλμα'),
            content: const Text('Λάθος email ή κωδικός.'),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                },
```

```
    onTap: () async {
      try {
        await _auth.sendPasswordResetEmail(
            email: _emailController.text);
        showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: const Text('Επιτυχία'),
              content:
                  const Text('Στάλθηκε email επαναφοράς κωδικού.'),
              actions: [
                TextButton(
                  onPressed: () {
                    Navigator.of(context).pop();
                  },
                  child: const Text('Κλείσιμο',
                      style:
                          TextStyle(color: AppColors.snowhubBlue)),
                ),
              ],
            );
          },
        );
      } catch (e) {
        log('Failed to send password reset email: $e');
      }
    },
    child: const Text(
      'Ξέχασα τον κωδικό μου.',
    ),
  ),
),
```

Σχήμα 2.24 Διαδικασία Σύνδεσης και Δημιουργίας Λογαριασμού

Διαχείριση Κρατήσεων και Αγαπημένων με χρήση Firebase Firestore

Για τη διαχείριση κρατήσεων και αγαπημένων χιονοδρομικών κέντρων, αξιοποιήθηκε η υπηρεσία Cloud Firestore της Firebase, η οποία προσφέρει δυνατότητες real-time βάσης δεδομένων με έμφαση στην ευελιξία και την κλιμακωσιμότητα.

Αγαπημένα Χιονοδρομικά

Κατά την είσοδο του χρήστη στην εφαρμογή, πραγματοποιείται μία αρχική κλήση προς τη Firestore με στόχο την πρόσβαση στο Collection Favorites, χρησιμοποιώντας το μοναδικό User ID του χρήστη. Αν το συγκεκριμένο Document δεν υφίσταται, η ίδια κλήση λειτουργεί και ως trigger για τη δημιουργία νέου Document. Σε διαφορετική περίπτωση, ανακτώνται τα δεδομένα που έχουν ήδη αποθηκευτεί.

Η δομή του συγκεκριμένου collection αποτελείται από ένα αντικείμενο τύπου Map με την ονομασία resorts, στο οποίο:

- Το key είναι το resortId (το μοναδικό αναγνωριστικό κάθε χιονοδρομικού).
- Το value είναι μια boolean τιμή: true για προσθήκη στα αγαπημένα και false για αφαίρεση.

Η προσέγγιση αυτή επιλέχθηκε σκόπιμα, ώστε το collection να είναι αρχικά κενό και να ενημερώνεται δυναμικά μόνο όταν ο χρήστης αλληλεπιδρά με την επιλογή προσθήκης στα αγαπημένα. Με αυτόν τον τρόπο, σε περίπτωση που προστεθούν ή αφαιρεθούν χιονοδρομικά στο backend της εφαρμογής, το Firestore δεν χρειάζεται να ενημερωθεί εκ νέου με λίστες resorts, εξασφαλίζοντας δυναμική προσαρμογή στα δεδομένα του backend.

Κρατήσεις

Στην περίπτωση των κρατήσεων, για κάθε νέα κράτηση που πραγματοποιεί ο χρήστης δημιουργείται νέο έγγραφο (document) στο collection user-bookings. Το έγγραφο αυτό περιλαμβάνει τις εξής πληροφορίες:

- Το χρονικό στίγμα (timestamp) της κράτησης.
- Τον αριθμό εισιτηρίων.
- Το ID του χιονοδρομικού κέντρου.
- Το User ID του χρήστη.

Η ανάκτηση των δεδομένων κρατήσεων γίνεται κατά την πλοήγηση του χρήστη στη σχετική σελίδα της εφαρμογής. Εκεί, η Firestore εκτελεί ερώτημα στο collection user-bookings και επιστρέφει όλα τα έγγραφα όπου το πεδίο userId ταιριάζει με τον τρέχοντα χρήστη.

Η διαδικασία υποβολής κράτησης ενεργοποιείται από τον χρήστη στην κύρια σελίδα του χιονοδρομικού. Στο κάτω μέρος της καρτέλας, ο χρήστης επιλέγει τον αριθμό των εισιτηρίων και πατώντας το κουμπί "Κράτηση", δημιουργείται αυτόματα νέο έγγραφο στο Firestore.

```
Widget build(BuildContext context) {
  CollectionReference userBookingsCollection =
      FirebaseFirestore.instance.collection('user-bookings');
  Query<Object?> userBookingsQuery =
      userBookingsCollection.where('user', isEqualTo: auth.currentUser!.uid);
```

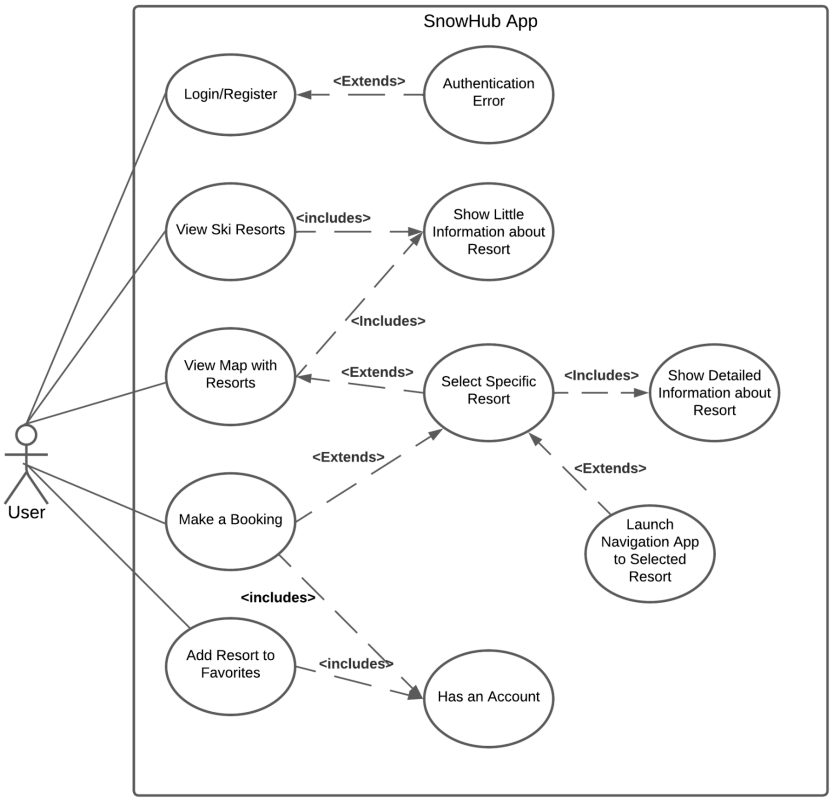
Σχήμα 3.25 user-bookings Firebase Κλήση

# Κεφάλαιο 4ο: Παρουσίαση της Εφαρμογής και Ροή Χρήστη

Η παρούσα ενότητα έχει ως στόχο την αναλυτική παρουσίαση της εφαρμογής SnowHub, τόσο από την πλευρά της λειτουργικότητας όσο και από την οπτική του τελικού χρήστη. Μέσα από τη μελέτη της συνολικής εμπειρίας χρήσης (user experience), περιγράφεται η δομή της εφαρμογής, οι επιλογές πλοήγησης που παρέχονται, καθώς και οι βασικές λειτουργίες που είναι διαθέσιμες στον χρήστη.

Η εφαρμογή αναπτύχθηκε με γνώμονα την ευχρηστία, την αμεσότητα πληροφορίας και τη δυνατότητα εξατομικευμένων ενεργειών, όπως κρατήσεις και αποθήκευση αγαπημένων. Ο χρήστης έχει τη δυνατότητα να περιηγηθεί στα διαθέσιμα χιονοδρομικά κέντρα, να δει χάρτη με τις τοποθεσίες τους, να επιλέξει ένα συγκεκριμένο χιονοδρομικό και να ενημερωθεί για λεπτομέρειες, να πραγματοποιήσει κράτηση ή να το προσθέσει στη λίστα αγαπημένων του.

Για την αποτύπωση της αλληλεπίδρασης του χρήστη με το σύστημα, παρουσιάζεται το παρακάτω Use Case Diagram. Το διάγραμμα αυτό αποσαφηνίζει τις βασικές ενέργειες (λειτουργίες) που μπορεί να εκτελέσει ο χρήστης, καθώς και τη ροή αυτών των ενεργειών εντός του οικοσυστήματος της εφαρμογής.




Σχήμα 4.1 Use Case Diagram

Όπως φαίνεται, ο κεντρικός δρών (actor) είναι ο χρήστης, ο οποίος έχει πρόσβαση σε μια σειρά από δυνατότητες όπως:

- Εγγραφή/Σύνδεση στο σύστημα

- Προβολή λίστας χιονοδρομικών και χάρτη
- Επιλογή συγκεκριμένου χιονοδρομικού και προβολή πληροφοριών
- Δυνατότητα κράτησης και αποθήκευσης στα αγαπημένα
- Χρήση εξωτερικής εφαρμογής πλοήγησης

Η παραπάνω απεικόνιση θέτει τις βάσεις για την αναλυτική παρουσίαση κάθε λειτουργίας που ακολουθεί στις επόμενες ενότητες του κεφαλαίου.

### 4.1 Εγγραφή/Σύνδεση στο Σύστημα

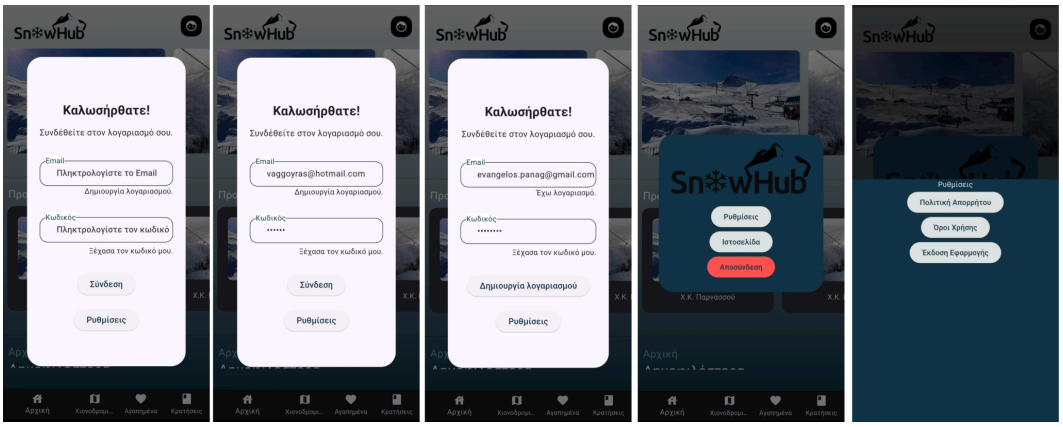
Η πρόσβαση του χρήστη στις βασικές λειτουργίες της εφαρμογής προϋποθέτει την ταυτοποίησή του μέσω διαδικασίας εγγραφής ή σύνδεσης. Η διαχείριση χρηστών υλοποιείται με τη χρήση της υπηρεσίας Firebase Authentication, η οποία προσφέρει αξιόπιστους και ασφαλείς μηχανισμούς αυθεντικοποίησης.

Κατά την εγγραφή, ο χρήστης καλείται να εισάγει ένα έγκυρο email και έναν κωδικό πρόσβασης. Το σύστημα ελέγχει ότι το email δεν έχει ήδη χρησιμοποιηθεί και ότι ο κωδικός πληροί τις ελάχιστες απαιτήσεις ασφάλειας, όπως το μήκος των 8 χαρακτήρων. Στην περίπτωση αποτυχίας οποιουδήποτε από τους παραπάνω ελέγχους, εμφανίζεται μήνυμα ειδοποίησης μέσω snackbar, ώστε ο χρήστης να ενημερώνεται άμεσα και ξεκάθαρα.

Αντιστοίχως, κατά τη διαδικασία σύνδεσης, επαληθεύεται αν ο συνδυασμός email και κωδικού αντιστοιχεί σε ήδη εγγεγραμμένο χρήστη. Σε περίπτωση επιτυχούς σύνδεσης, ο χρήστης μεταφέρεται στην κύρια σελίδα της εφαρμογής. Σε περίπτωση σφάλματος, όπως λανθασμένα στοιχεία ή μη έγκυρη διεύθυνση email, εμφανίζεται ανάλογο μήνυμα σφάλματος.

Η εφαρμογή υποστηρίζει επιπλέον δυνατότητα επαναφοράς κωδικού πρόσβασης, δίνοντας στον χρήστη τη δυνατότητα να ζητήσει την αποστολή σχετικού email από το Firebase με οδηγίες αλλαγής. Και σε αυτή την περίπτωση, η επιτυχής αποστολή του email επιβεβαιώνεται στον χρήστη μέσω snackbar μηνύματος.

Η υλοποίηση της αυθεντικοποίησης έχει δομηθεί με τέτοιο τρόπο ώστε να ενισχύεται η ασφάλεια της εφαρμογής, ενώ παράλληλα διασφαλίζεται η ομαλή και κατανοητή εμπειρία χρήσης, ειδικά για νέους χρήστες.




Σχήμα 4.2 Διαδικασία Εγγραφής/Σύνδεσης

### 4.2 Προβολή λίστας χιονοδρομικών και χάρτη

Μετά την επιτυχή σύνδεση, ο χρήστης μεταφέρεται στην αρχική οθόνη της εφαρμογής, η οποία περιλαμβάνει τη λίστα των διαθέσιμων χιονοδρομικών κέντρων. Η λίστα αυτή αποτελεί την κύρια διεπαφή αναζήτησης και πλοήγησης στις βασικές λειτουργίες της εφαρμογής.

Για την εμφάνιση των χιονοδρομικών, αξιοποιούνται δεδομένα που ανακτώνται από εξωτερική API κατά την εκκίνηση της εφαρμογής (στη μέθοδο main()), καθώς θεωρούνται σταθερά δεδομένα με περιορισμένες μεταβολές. Η φόρτωσή τους κατά την εκκίνηση βελτιώνει την απόδοση της εφαρμογής και εξασφαλίζει ταχύτητα στην αλληλεπίδραση με τον χρήστη.

Η κάθε εγγραφή στη λίστα περιλαμβάνει:

- Την ονομασία του χιονοδρομικού
- Την περιοχή στην οποία βρίσκεται
- Τη σχετική εικόνα (thumbnail)
- Την κατάσταση λειτουργίας (ενεργό/ανενεργό)
- Ένα εικονίδιο καρδιάς το οποίο υποδηλώνει εάν το χιονοδρομικό έχει προστεθεί στα αγαπημένα του χρήστη.

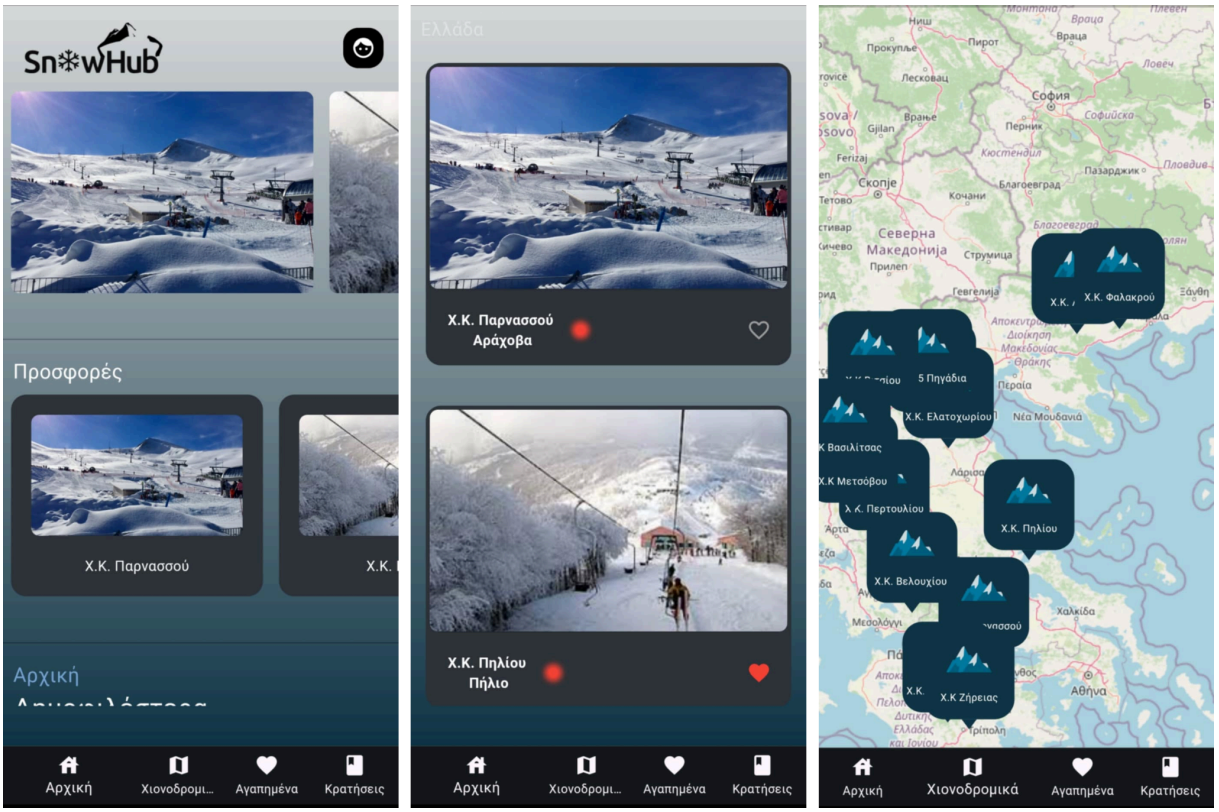
Η καρδιά εμφανίζεται:

- Κενή (outline) όταν το χιονοδρομικό δεν είναι στα αγαπημένα
- Γεμάτη κόκκινη όταν είναι ήδη αποθηκευμένο στα αγαπημένα του χρήστη

Η προσθήκη ή αφαίρεση από τα αγαπημένα γίνεται με απλό άγγιγμα του εικονιδίου και ενημερώνεται άμεσα στο Firestore, επιτρέποντας στον χρήστη να διαχειρίζεται εύκολα τις προτιμήσεις του.

Παράλληλα, ο χρήστης μπορεί να επιλέξει προβολή μέσω χάρτη, στον οποίο παρουσιάζονται όλα τα διαθέσιμα χιονοδρομικά ως markers. Ο διαδραστικός αυτός χάρτης ενισχύει την πλοήγηση παρέχοντας γεωγραφικό πλαίσιο στην αναζήτηση. Με την επιλογή ενός marker, ο χρήστης μεταφέρεται άμεσα στη σελίδα του αντίστοιχου χιονοδρομικού για περισσότερες πληροφορίες.

Η συνδυαστική παρουσίαση μέσω λίστας και χάρτη καλύπτει τις ανάγκες διαφορετικών τύπων χρηστών, παρέχοντας τόσο γρήγορη πλοήγηση όσο και οπτική αντίληψη της τοποθεσίας των χιονοδρομικών.




Σχήμα 4.3 Αρχική και Χάρτης

### 4.3 Επιλογή συγκεκριμένου χιονοδρομικού και προβολή πληροφοριών

Κατά την επιλογή ενός χιονοδρομικού κέντρου από τη λίστα ή από το διαδραστικό χάρτη, ο χρήστης μεταφέρεται στη σελίδα λεπτομερειών του επιλεγμένου χιονοδρομικού. Στο σημείο αυτό πραγματοποιείται νέα κλήση API, η οποία ζητά τις επικαιροποιημένες πληροφορίες του συγκεκριμένου χιονοδρομικού βάσει του μοναδικού του αναγνωριστικού (ID). Αυτό επιτρέπει στην εφαρμογή να εμφανίζει πρόσφατα δεδομένα, απαραίτητα για μια εξατομικευμένη και δυναμική εμπειρία χρήσης.

Η σελίδα λεπτομερειών περιλαμβάνει μεταξύ άλλων:

- Αναλυτική περιγραφή του χιονοδρομικού (τοποθεσία, ιστοσελίδα, κατάσταση λειτουργίας)
- Στοιχεία υψομέτρου (βάση και κορυφή)
- Καιρικές συνθήκες και αναφορές χιονιού (τελευταία χιονόπτωση, ποιότητα χιονιού, βάθος)
- Λίστα αναβατήρων με πληροφορίες λειτουργίας, πληρότητας, διάρκειας, χωρητικότητας και συντεταγμένων
- Διαθέσιμες δραστηριότητες ανά γλώσσα (π.χ., ski, snowboard, snowmobile)
- Λίστα πιστών με βασικά χαρακτηριστικά όπως δυσκολία, μήκος, υψόμετρο και ποσοστό κλίσης
- Πίνακα επεξήγησης χρωματισμών των πιστών

Επιπλέον, η σελίδα συνοδεύεται από πρόσφατες εικόνες του χιονοδρομικού, οι οποίες ανακτώνται μέσω του endpoint /images, προσφέροντας στον χρήστη μια πιο παραστατική εικόνα των συνθηκών.

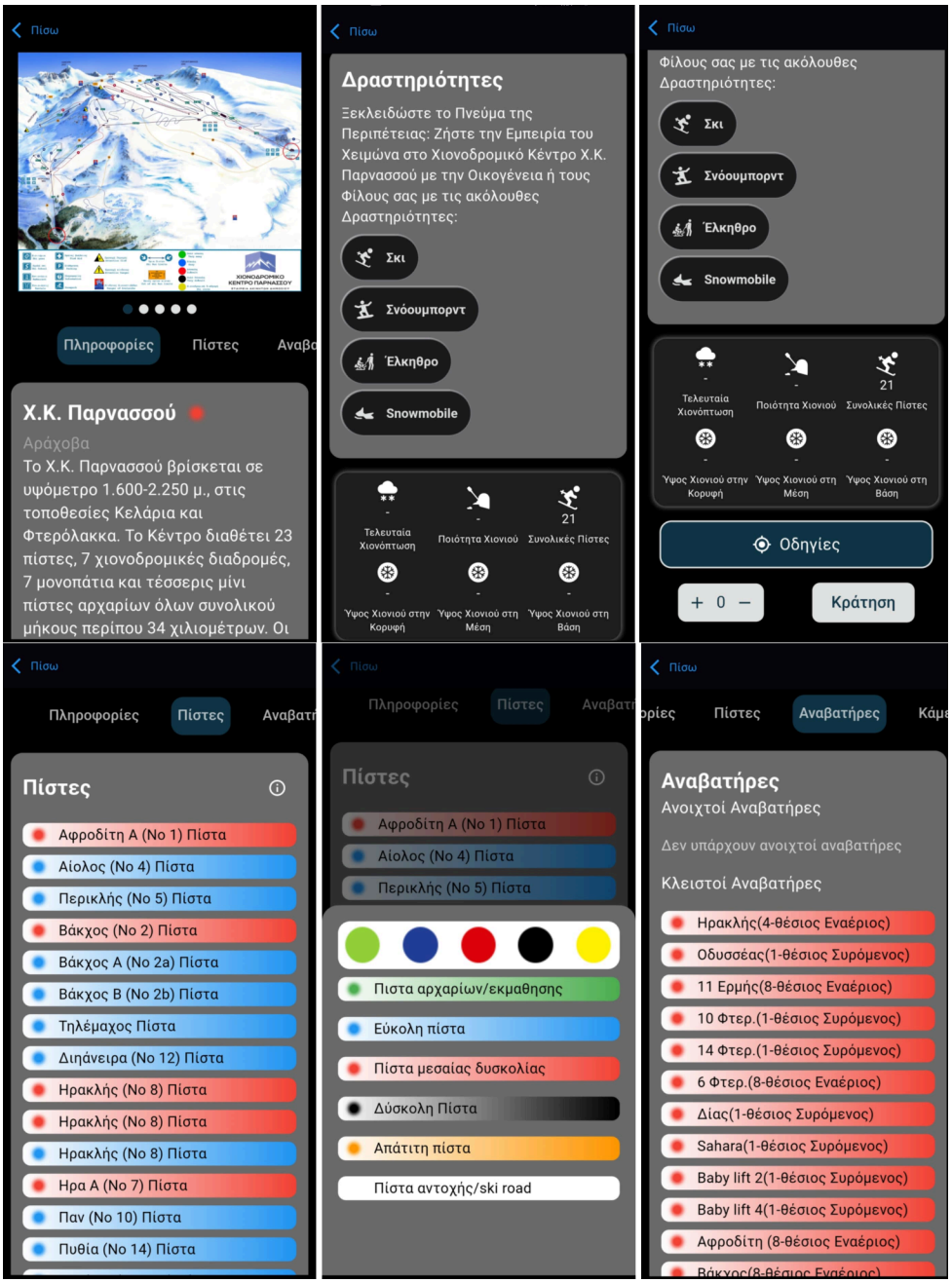
Η δομή της πληροφορίας είναι προσεκτικά σχεδιασμένη ώστε να διαχωρίζει την στατική πληροφορία (όπως η περιγραφή και το υψόμετρο) από την δυναμική (όπως η κατάσταση λειτουργίας των lift και οι

καιρικές συνθήκες), επιτυγχάνοντας έναν ισορροπημένο συνδυασμό πληροφόρησης και απόκρισης σε πραγματικό χρόνο.

Η οθόνη αυτή προσφέρει, τέλος, τη δυνατότητα άμεσης ενέργειας από τον χρήστη, καθώς περιλαμβάνονται επιλογές για:

- Κράτηση εισιτηρίου (pass)
- Πλοήγηση στο χιονοδρομικό μέσω εφαρμογής χαρτών

Αυτή η λειτουργικότητα εξυπηρετεί τόσο τη διευκόλυνση της απόφασης του χρήστη όσο και τη βελτιστοποίηση της εμπειρίας πλοήγησης εντός της εφαρμογής.




Σχήμα 4.4 Resort

### 4.4 Δυνατότητα κράτησης και αποθήκευσης στα αγαπημένα

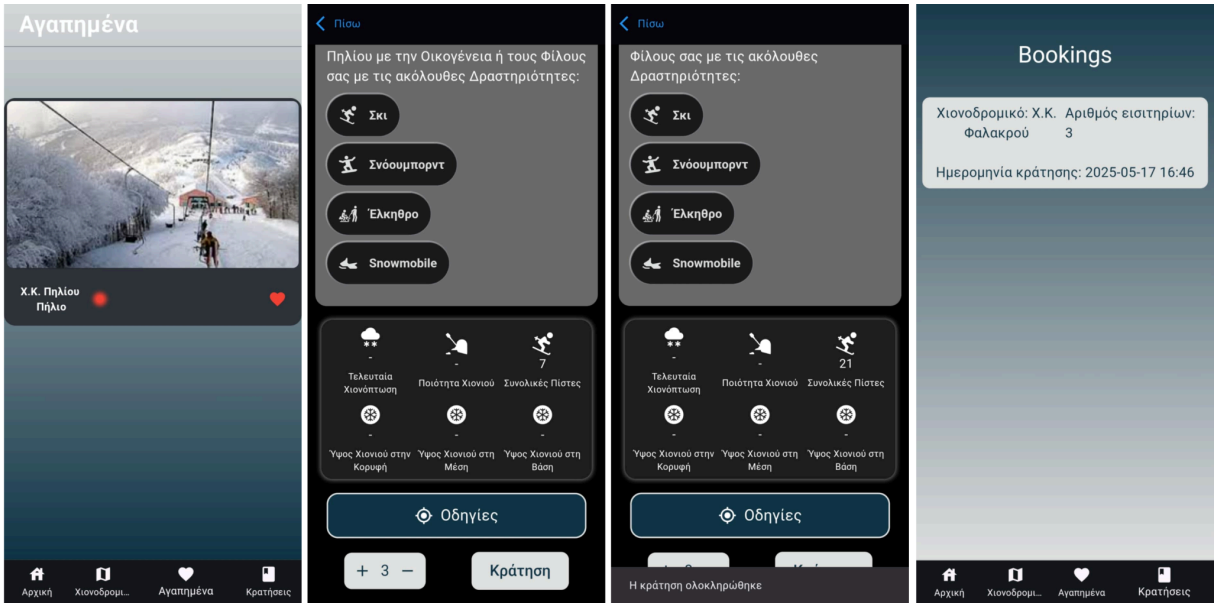
Ο χρήστης έχει τη δυνατότητα να προσθέσει οποιοδήποτε χιονοδρομικό στις προσωπικές του προτιμήσεις πατώντας το εικονίδιο της καρδιάς που εμφανίζεται σε κάθε κάρτα χιονοδρομικού. Αν το χιονοδρομικό δεν βρίσκεται ήδη στη λίστα αγαπημένων, το εικονίδιο είναι κενό, ενώ όταν προστεθεί, γεμίζει με κόκκινο χρώμα, παρέχοντας άμεση οπτική ένδειξη της ενέργειας.

Η προσθήκη ή η αφαίρεση από τα αγαπημένα συγχρονίζεται με τη βάση δεδομένων Firestore, όπου κάθε χρήστης έχει αποθηκευμένο το δικό του σύνολο αγαπημένων, βάσει του μοναδικού του userId. Έτσι, οι αγαπημένες επιλογές είναι διαθέσιμες σε κάθε επόμενη συνεδρία σύνδεσης και από οποιαδήποτε συσκευή.

Η δυνατότητα κράτησης ενεργοποιείται από τη σελίδα ενός συγκεκριμένου χιονοδρομικού. Όταν ο χρήστης είναι συνδεδεμένος, εμφανίζονται τα αντίστοιχα κουμπιά που του επιτρέπουν να επιλέξει αριθμό εισιτηρίων. Αν δεν έχει πραγματοποιηθεί σύνδεση, τα κουμπιά κράτησης δεν εμφανίζονται καθόλου, διασφαλίζοντας ότι η λειτουργία είναι προσβάσιμη μόνο σε ταυτοποιημένους χρήστες.

Μόλις ολοκληρωθεί η κράτηση, ο χρήστης ενημερώνεται μέσω SnackBar μηνύματος, π.χ. «Η κράτησή ολοκληρώθηκε». Η πληροφορία της κράτησης (όνομα χιονοδρομικού, ημερομηνία και αριθμός εισιτηρίων) αποθηκεύεται στο Firestore, συνδεδεμένη με τον λογαριασμό του χρήστη.

Στη συνέχεια, ο χρήστης μπορεί να μεταβεί στην ειδική σελίδα "Κρατήσεις", όπου και θα εμφανίζεται η κράτησή του με όλα τα απαραίτητα στοιχεία.




Σχήμα 4.5 Αγαπημένα και Κρατήσεις

# Κεφάλαιο 5ο: Συμπεράσματα και Προτάσεις Βελτίωσης

Η ανάπτυξη της εφαρμογής SnowHub υπήρξε μία εξαιρετικά δημιουργική και απολαυστική διαδικασία. Η συνεργασία με τον συνάδελφο Ξενοφών Πάντσος ήταν άριστη, με σαφή κατανομή

αρμοδιοτήτων, αποτελεσματική επικοινωνία και κοινό στόχο την υλοποίηση μιας εφαρμογής υψηλής χρηστικότητας και λειτουργικότητας.

Το αντικείμενο της εργασίας αποτέλεσε λύση σε ένα πραγματικό πρόβλημα που αντιμετωπίζουμε τόσο εγώ προσωπικά όσο και πολλοί φίλοι και συνάδελφοι χιονοδρόμοι κάθε χειμερινή περίοδο: την έλλειψη μιας σύγχρονης, αξιόπιστης και συγκεντρωτικής πλατφόρμας πληροφόρησης για τα ελληνικά χιονοδρομικά κέντρα. Το έργο είχε άμεση σύνδεση με τις ανάγκες της κοινότητας και, παρά τη δυσκολία ορισμένων φάσεων, αποτέλεσε μια συνολικά θετική και εποικοδομητική εμπειρία.

Κατά τη διάρκεια της ανάπτυξης προέκυψαν τεχνικές προκλήσεις και σημεία που απαιτούσαν επανασχεδιασμό ή βελτίωση. Με την ολοκλήρωση της υλοποίησης, παρατηρούνται πλέον λεπτομέρειες και αστοχίες που μπορούν να βελτιωθούν, γεγονός φυσιολογικό σε κάθε στάδιο ανάπτυξης. Αν και δεν πρόκειται για μια πλήρως έτοιμη λύση προς άμεση εμπορική αξιοποίηση, η παρούσα εφαρμογή υπερβαίνει τα χαρακτηριστικά ενός απλού proof of concept και μπορεί να θεωρηθεί ως ένα ολοκληρωμένο demo, το οποίο με στοχευμένες βελτιώσεις θα μπορούσε να κυκλοφορήσει στην αγορά.

Προτάσεις Βελτίωσης:

1. Καθαρισμός και αναδιάρθρωση κώδικα (Code Refactoring): Ο κώδικας μπορεί να οργανωθεί περαιτέρω ώστε να είναι περισσότερο ευανάγνωστος, επαναχρησιμοποιήσιμος και επεκτάσιμος (scalable). Η τήρηση των αρχών του clean architecture και η χρήση design patterns θα ενίσχυαν τη συντηρησιμότητα του έργου.
2. Εντοπισμός και Διόρθωση Σφαλμάτων (Bugs): Κατά τη χρήση της εφαρμογής εντοπίστηκαν ορισμένα σφάλματα, τα οποία θα πρέπει να επιλυθούν ώστε να βελτιωθεί η εμπειρία χρήστη και η σταθερότητα της εφαρμογής.
3. Ανάπτυξη Διαχειριστικού Περιβάλλοντος: Ύστερα από επαφές με εκπροσώπους χιονοδρομικών κέντρων, όπως το χιονοδρομικό κέντρο Σελίου, αναδείχθηκε η ανάγκη ύπαρξης ενός ανεξάρτητου διαχειριστικού συστήματος, μέσω του οποίου κάθε χιονοδρομικό θα μπορεί να ενημερώνει άμεσα και με αξιοπιστία τις πληροφορίες του, χωρίς να βασίζεται σε τηλεφωνική επικοινωνία ή τρίτους.
4. Μετατροπή των Κρατήσεων σε Πραγματικά Εισιτήρια: Προτείνεται οι κρατήσεις που πραγματοποιούνται μέσω της εφαρμογής να εξελιχθούν σε πλήρως λειτουργικά ψηφιακά εισιτήρια με χρήση QR code και ενδεχόμενη ενσωμάτωση ενός συστήματος πληρωμών, όπως το Stripe, για την ασφαλή διεκπεραίωση των συναλλαγών.

Συνολική Αξιολόγηση

Το εγχείρημα του SnowHub απέδειξε τη δυνατότητα υλοποίησης μιας εφαρμογής που όχι μόνο επιλύει πρακτικά προβλήματα των χιονοδρόμων αλλά έχει και προοπτικές εμπορικής αξιοποίησης. Η μέχρι τώρα πορεία, παρά τις τεχνικές προκλήσεις και τα εμπόδια, υπήρξε ενθαρρυντική. Με συνέχιση της ανάπτυξης και στρατηγικές βελτιώσεις, η εφαρμογή μπορεί να εξελιχθεί σε ένα πλήρες και ανταγωνιστικό εργαλείο για την ελληνική – και όχι μόνο – χειμερινή τουριστική αγορά.

# ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Flutter. (n.d.). Flutter - Build apps for any screen. Retrieved May 15, 2025, from https://flutter.dev/

[2] Lucid Software. (n.d.). *Lucidchart - Online Diagramming Application*. Retrieved May 15, 2025, from https://lucid.app/

[3] Firebase. (n.d.). *Firebase - App development platform*. Google. Retrieved May 15, 2025, from https://firebase.google.com/

[4] BLoC Library. (n.d.). *Flutter BLoC - predictable state management library*. Retrieved May 15, 2025, from https://bloclibrary.dev/

[5] Fleaflet. (n.d.). *Fleaflet - Flutter leaflet maps*. Retrieved May 15, 2025, from https://docs.fleaflet.dev/

[6] GitHub. (n.d.). *GitHub Documentation*. [Online]. Available: https://docs.github.com/en

[7] Postman. (n.d.). *Postman Documentation*. [Online]. Available: https://learning.postman.com/docs