ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός & Ανάπτυξη iOS Εφαρμογής για Χιονοδρομικά στην Ελλάδα»

Του φοιτητή
Σεφερίδη Βενιαμίν
Αρ. Μητρώου: 174955

Επιβλέπων
Χαράλαμπος Μπράτσας
Επίκουρος Καθηγητής

Ημερομηνία Σεπτέμβριος 2024

Τίτλος Π.Ε. **Σχεδιασμός & Ανάπτυξη iOS Εργαφμοργής για Χιονοδρομικά στην Ελλάδα**
Κωδικός Π.Ε. **24112**
Φοιτητής: **Βενιαμίν Σεφερίδης**
Εισηγητής: **Χαράλαμπος Μπράτσας**
Ημερομηνία ανάληψης Π.Ε. **25-01-2024**
Ημερομηνία περάτωσης Π.Ε. **25-09-2024**

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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Πρόλογος

Η επιλογή της παρούσας πτυχιακής εργασίας ήταν αποτέλεσμα τόσο της επαγγελματικής και ακαδημαϊκής μου σταδιοδρομίας όσο και του προσωπικού μου ενδιαφέροντος για τον χειμερινό αθλητισμό. Έχοντας υπάρξει επαγγελματίας μηχανικός λογισμικού εφαρμογών iOS τα τελευταία δύο έτη, αλλά και ερασιτέχνης σκιέρ από τότε που θυμάμαι τον εαυτό μου, το πεδίο της ανάπτυξης μιας εφαρμογής χιονοδρομικών κέντρων στην Ελλάδα μου κέντρισε αμέσως το ενδιαφέρον. Η ιδέα για την εφαρμογή γεννήθηκε από την ενασχόλησή μου ως σκιέρ, καθώς παρατήρησα ότι όποτε ήθελα να επισκεφθώ ένα χιονοδρομικό κέντρο, δεν υπήρχε μια εφαρμογή όπου θα μπορούσα να κλείσω εισιτήριο ή να βρω πληροφορίες για το χιονοδρομικό κέντρο που με ενδιέφερε. Προσωπικός μου στόχος είναι με την περάτωση της πτυχιακής εργασίας να έχω εξελιχθεί στον καθένα από τους παραπάνω τομείς έστω και λίγο, καθώς επίσης και να λάβω εφόδια για να συνεχίσω την επιτυχημένη ενασχόληση με αυτούς. Τόσο ο προγραμματισμός όσο και το σκι είναι ασχολίες τις οποίες αγαπώ και συνδυαστικά αποτελούν το μεγαλύτερο κομμάτι της καθημερινότητάς μου.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής για το λειτουργικό σύστημα iOS, με αντικείμενο την παροχή ενημερωμένων πληροφοριών σχετικά με την κατάσταση των χιονοδρομικών κέντρων στην Ελλάδα. Η εφαρμογή που δημιουργήθηκε κατά τη διάρκεια της εργασίας αυτής στοχεύει, με μελλοντικές υλοποιήσεις και συνεχή βελτίωση, να καλύψει ένα σημαντικό μέρος των αναγκών του χειμερινού τουρισμού, ο οποίος παρουσιάζει αυξητική τάση τα τελευταία χρόνια.

Πέρα από την ίδια την υλοποίηση της εφαρμογής, η πτυχιακή εργασία επιδιώκει να παρουσιάσει λεπτομερώς όλη τη διαδικασία ανάπτυξης μιας τέτοιας εφαρμογής. Αυτό περιλαμβάνει τα στάδια της αρχικής μελέτης και σχεδίασης, την ανάπτυξη του λογισμικού, καθώς και την επιλογή της κατάλληλης αρχιτεκτονικής και των εργαλείων. Επιπλέον, παρέχεται ένας πλούτος κώδικα σε γλώσσα Swift, που αφορά την υλοποίηση της mobile εφαρμογής.

Με την ενδελεχή μελέτη των προαναφερθέντων θεμάτων, ο αναγνώστης της εργασίας αυτής θα αποκτήσει πολύτιμες γνώσεις σχετικά με την επαγγελματική ανάπτυξη εφαρμογών και θα είναι σε θέση να κάνει τα πρώτα του βήματα σε αυτόν τον τομέα, εφόσον το επιθυμεί. Από την προσωπική μου εκτίμηση, οι στόχοι της εργασίας αυτής επιτεύχθηκαν σε ικανοποιητικό βαθμό. Το τελικό αποτέλεσμα, το οποίο αναλύεται αναλυτικά στις επόμενες σελίδες, είναι μια εφαρμογή που προσφέρει βασικές λειτουργίες και, παρόλο που απαιτεί περαιτέρω βελτιώσεις και εξέλιξη, έχει θεμελιωθεί σωστά και αποτελεί μια πολύ καλή βάση για μελλοντική ανάπτυξη.

«Design & Development of an iOS Application

for Ski Resorts in Greece »

«Benjamin Seferidis»

Abstract

This thesis focuses on the development of an application for the iOS operating system, specifically designed to provide updated information regarding the status of ski resorts in Greece. The application developed during this thesis aims, with future implementations and continuous improvement, to address a significant portion of the needs of the winter tourism sector, which has shown an upward trend in recent years.

Beyond the implementation of the application itself, the thesis seeks to meticulously present the entire process of developing such an application. This encompasses the stages of initial study and design, software development, as well as the selection of appropriate architecture and tools. Additionally, a wealth of code segments in Swift language is provided, which pertains to the implementation of the mobile application.

By thoroughly studying the aforementioned topics, the reader of this thesis will acquire valuable knowledge concerning the professional development of applications and will be able to take their first steps in this field, should they wish to do so. In my personal assessment, the objectives of this thesis have been achieved to a satisfactory degree. The final outcome, which is analyzed in detail in the following pages, is an application that offers essential functionalities. While it requires further improvements and development, it has been properly established and constitutes an excellent foundation for future advancement.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τους συμφοιτητές, συναδέλφους και καλούς μου φίλους, Ανδρέα Μεϊμάρογλου και Ξενοφών Νικόλαο Πάντσο, οι οποίοι υλοποίησαν πτυχιακή εργασία με αντίστοιχο θέμα σε front-end & back-end δίνοντας μου την δυνατοτητα να υλοποιήσω μια ολοκληρωμένη εφαρμογή. Η εκπόνηση τέτοιου είδους εργασίας αποτελεί μια δύσκολο χρονοβόρα και επίπονη διαδικασια, μέσα από την συνεργασία μας όμως ήταν ιδιαιτέρως ευχάριστη και δημιουργική.

Συνεχίζοντας θα ήθελα ιδιαιτέρως να ευχαριστήσω το μέντορα μου, Χρήστο Χρήστου, ο οποίος αποτελεί πρότυπο συναδέλφου καθώς μου πρόσφερε απλόχερα γνωση και ήταν πάντα πρόθυμος να με καθοδηγησει και να με βοηθησει με οποιαδηποτε δυσκολια η απορία είχα. Τα εφόδια που μου προσέφερε σε συνδυασμό με την προσωπικη μου μελέτη και ενασχόληση συνέβαλαν στην μετάβαση μου απο φοιτητη πληροφορικής σε μηχανικό λογισμικού. Η παρούσα εργασία δε θα ήταν δυνατό να πραγματοποιηθεί δίχως τις βάσεις και τις γνώσεις που αποκόμισα από αυτόν.

Κλείνοντας, θα ήθελα να ευχαριστήσω όλους τους ανθρώπους που αποτελούν σημαντικό κομμάτι της ζωής μου, με πρώτους από όλους την οικογένεια μου, που συνθέτουν το ποιος είμαι σήμερα. Ευτυχώς είναι πολλοί για να τους αναφέρω ονομαστικά, ξέρουν όμως ποιοι είναι και προσπαθώ να τους ευχαριστώ καθημερινά με την εξέλιξή μου και την πορεία μου.

Περιεχόμενα

Συντομογραφίες

Π.Ε.	Πτυχιακή Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η εξέλιξη των κινητών εφαρμογών έχει φέρει επανάσταση σε πολλούς τομείς της καθημερινής ζωής, επηρεάζοντας τον τρόπο με τον οποίο αλληλεπιδρούμε με την τεχνολογία και τις υπηρεσίες γύρω μας. Οι εφαρμογές κινητών συσκευών αποτελούν πλέον αναπόσπαστο κομμάτι της καθημερινότητάς μας, παρέχοντας δυνατότητες για επικοινωνία, ψυχαγωγία, αγορές, αλλά και για την οργάνωση και διαχείριση ποικίλων δραστηριοτήτων και πληροφοριών. Στο πλαίσιο αυτό, η ανάπτυξη εφαρμογών που εξυπηρετούν συγκεκριμένες ανάγκες χρηστών και καταστάσεων έχει γνωρίσει ιδιαίτερη άνθηση, με τις εφαρμογές για ταξίδια και δραστηριότητες στη φύση να βρίσκονται στο επίκεντρο του ενδιαφέροντος τα τελευταία χρόνια.

Η παρούσα πτυχιακή εργασία εστιάζει στο σχεδιασμό και την ανάπτυξη μιας iOS εφαρμογής που απευθύνεται στους λάτρεις των χειμερινού τουρισμού στην Ελλάδα. Η εφαρμογή αυτή στοχεύει να προσφέρει ολοκληρωμένη ενημέρωση και λειτουργικότητα, παρέχοντας στους χρήστες πληροφορίες για τα χιονοδρομικά κέντρα, όπως τις τρέχουσες καιρικές συνθήκες, την κατάσταση των πιστών, τις τιμές των εισιτηρίων, καθώς και δυνατότητες πλοήγησης και κρατήσεων. Στο κεφάλαιο αυτό θα γίνει μια συνοπτική παρουσίαση της ιστορίας και της εξέλιξης των iOS εφαρμογών, εστιάζοντας στη σημασία τους για τον τουριστικό και αθλητικό τομέα, καθώς και στη συμβολή τους στην ανάπτυξη των χιονοδρομικών δραστηριοτήτων.

Η εργασία αυτή έρχεται να προσφέρει μια ολοκληρωμένη λύση στο πρόβλημα της έλλειψης ψηφιακών εργαλείων για τα χιονοδρομικά κέντρα, προωθώντας την καινοτομία και την αποτελεσματικότητα στον τομέα του τουρισμού.

1.2 Σύλληψη της ιδέας

Ο χειμερινός τουρισμός στην Ελλάδα γνωρίζει σημαντική άνοδο τα τελευταία χρόνια, με τα χιονοδρομικά κέντρα να επενδύουν σε ανανέωση εξοπλισμού και βελτίωση των υποδομών τους. Αυτή η εξέλιξη αντικατοπτρίζει την αυξημένη ζήτηση και το ενδιαφέρον τόσο των εγχώριων όσο και των διεθνών επισκεπτών για τις χειμερινές δραστηριότητες στα ελληνικά βουνά. Παρά τις θετικές αυτές τάσεις, παρατηρείται ένα σημαντικό κενό στην αγορά: η απουσία μιας ολοκληρωμένης εφαρμογής για κινητά που να συγκεντρώνει όλες τις απαραίτητες πληροφορίες σχετικά με τα χιονοδρομικά κέντρα της χώρας.

Στο εξωτερικό, αντίστοιχες εφαρμογές είναι ευρέως διαδεδομένες και έχουν συμβάλει ουσιαστικά στην προώθηση και την ανάπτυξη του χειμερινού τουρισμού. Παρέχουν στους χρήστες άμεση πρόσβαση σε πληροφορίες για τις καιρικές συνθήκες, την κατάσταση των πιστών, τις τιμές των εισιτηρίων, καθώς και δυνατότητες κράτησης και πλοήγησης. Στην Ελλάδα, οι πληροφορίες αυτές είναι συχνά κατακερματισμένες και διάσπαρτες σε διάφορες πηγές, καθιστώντας δύσκολη την άμεση και αξιόπιστη ενημέρωση των ενδιαφερόμενων.

Η έλλειψη μιας τέτοιας εφαρμογής αποτελεί σημαντικό εμπόδιο στην περαιτέρω ανάπτυξη του χειμερινού τουρισμού. Οι σύγχρονοι ταξιδιώτες, ιδιαίτερα οι νεότερες γενιές που είναι εξοικειωμένες με την τεχνολογία, αναζητούν εύκολη και γρήγορη πρόσβαση σε πληροφορίες μέσω των κινητών τους

συσκευών. Η απουσία μιας ενιαίας πλατφόρμας μειώνει την ανταγωνιστικότητα των ελληνικών χιονοδρομικών κέντρων σε σχέση με αυτά του εξωτερικού.

Η παρούσα πτυχιακή εργασία επικεντρώνεται στο σχεδιασμό και την ανάπτυξη μιας iOS εφαρμογής που στοχεύει να καλύψει αυτό το κενό στην αγορά.

Η εφαρμογή θα προσφέρει:

- **Συγκεντρωμένη πληροφόρηση** για όλα τα χιονοδρομικά κέντρα της Ελλάδας, συμπεριλαμβανομένων των καιρικών συνθηκών, της κατάστασης των πιστών και των διαθέσιμων υπηρεσιών.
- **Άμεση ενημέρωση** για τις τιμές των εισιτηρίων, τις προσφορές και τις εκδηλώσεις που πραγματοποιούνται στα χιονοδρομικά κέντρα.
- **Δυνατότητες κράτησης**, διευκολύνοντας τον προγραμματισμό των επισκέψεων.
- **Ενσωματωμένη πλοήγηση** και χάρτες για εύκολη πρόσβαση στα κέντρα.
- **Διαδραστικά χαρακτηριστικά**, όπως δημιουργία χρηστών, που ενισχύουν την εμπειρία και την εμπιστοσύνη των επισκεπτών.

Η ανάπτυξη αυτής της εφαρμογής δεν είναι απλώς μια τεχνολογική καινοτομία, αλλά μια στρατηγική κίνηση που μπορεί να συμβάλλει σημαντικά στην ενίσχυση του χειμερινού τουρισμού. Με τη συγκέντρωση όλων των απαραίτητων πληροφοριών σε μια ενιαία πλατφόρμα, διευκολύνεται η διαδικασία λήψης αποφάσεων των επισκεπτών και ενισχύεται η προβολή των χιονοδρομικών κέντρων.

1.3 Ιστορικό iOS Εφαρμογών

Από την παρουσίαση του πρώτου iPhone το 2007 και την επακόλουθη κυκλοφορία του App Store το 2008, οι iOS εφαρμογές έχουν γίνει ένα σημαντικό εργαλείο για εκατομμύρια χρήστες παγκοσμίως. Η επαναστατική αυτή πλατφόρμα άνοιξε νέους ορίζοντες για τους προγραμματιστές, δίνοντάς τους τη δυνατότητα να αναπτύξουν και να διανείμουν εφαρμογές για μια παγκόσμια αγορά [4]. Η ιστορία των iOS εφαρμογών μπορεί να χωριστεί σε συγκεκριμένες περιόδους, καθεμία από τις οποίες χαρακτηρίζεται από συγκεκριμένες καινοτομίες και τάσεις.

- **Η Γένεση της Πλατφόρμας (2007-2008):** Το πρώτο iPhone, με την οθόνη αφής και το προηγμένο λειτουργικό σύστημα, αποτελεί την αρχή μιας νέας εποχής για τα smartphones. Με την κυκλοφορία του App Store το 2008, η Apple έδωσε στους προγραμματιστές τη δυνατότητα να δημιουργήσουν και να μοιραστούν τις εφαρμογές τους με ένα παγκόσμιο κοινό. Αυτή η κίνηση αναδιαμορφωσε τη βιομηχανία της κινητής τηλεφωνίας, δημιουργώντας το οικοσύστημα των εφαρμογών όπως το γνωρίζουμε σήμερα.
- **Η Ανάπτυξη και Διάδοση των Εφαρμογών (2008-2012):** Κατά την πρώτη περίοδο ανάπτυξης, οι εφαρμογές επικεντρώθηκαν σε βασικές λειτουργίες, όπως η επικοινωνία και η ψυχαγωγία. Ωστόσο, με την εισαγωγή του iPad το 2010, οι δυνατότητες επεκτάθηκαν σημαντικά, επιτρέποντας την ανάπτυξη πιο εξειδικευμένων εφαρμογών, όπως αυτές για ταξίδια και δραστηριότητες στη φύση, συμπεριλαμβανομένων των εφαρμογών για χιονοδρομικά κέντρα.
- **Εξέλιξη και Καινοτομία (2012-2015):** Η εισαγωγή της γλώσσας προγραμματισμού Swift το 2014 κατέστησε την ανάπτυξη εφαρμογών πιο αποδοτική και προσβάσιμη. Οι

προγραμματιστές άρχισαν να δημιουργούν εφαρμογές που απευθύνονται σε συγκεκριμένες ανάγκες και ενδιαφέροντα, όπως οι εφαρμογές για ταξίδια και τουρισμό, που προσφέρουν στους χρήστες πληροφορίες σε πραγματικό χρόνο και δυνατότητες προγραμματισμού των διακοπών τους.

- **Εστίαση στον Χρήστη και την Εμπειρία (2015-Παρόν):** Η Apple, δίνοντας έμφαση στην ασφάλεια και την προστασία των προσωπικών δεδομένων, προώθησε την ανάπτυξη εφαρμογών που σέβονται την ιδιωτικότητα των χρηστών. Παράλληλα, οι εφαρμογές έγιναν πιο προηγμένες, προσφέροντας λειτουργικότητες όπως η ενσωμάτωση τεχνητής νοημοσύνης για την παροχή προσωποποιημένων προτάσεων, κάτι που είναι ιδιαίτερα χρήσιμο σε εφαρμογές που αφορούν ταξίδια και δραστηριότητες στη φύση.
- **Παγκόσμια Επιρροή και Εξειδίκευση (2017-Παρόν):** Οι iOS εφαρμογές έχουν καταστεί παγκόσμιο φαινόμενο, με το App Store να φιλοξενεί εκατομμύρια εφαρμογές για κάθε πιθανή ανάγκη. Στον τομέα των χιονοδρομικών δραστηριοτήτων, οι εφαρμογές που παρέχουν χρήσιμες πληροφορίες για τους επισκέπτες, όπως αυτή που αναπτύσσεται στην παρούσα πτυχιακή εργασία, αποκτούν ολοένα και μεγαλύτερη σημασία, καθώς συνδυάζουν την τεχνολογία με τη φύση, διευκολύνοντας την εμπειρία των χρηστών.

Συνολικά, οι εφαρμογές iOS έχουν επιφέρει σημαντικές αλλαγές σε πολλούς τομείς, συμπεριλαμβανομένου του τουρισμού και των αθλητικών δραστηριοτήτων. Η ανάπτυξη της συγκεκριμένης εφαρμογής για χιονοδρομικά κέντρα στην Ελλάδα αποσκοπεί στη βελτίωση της εμπειρίας των επισκεπτών, προσφέροντάς τους ένα χρήσιμο εργαλείο για τον προγραμματισμό και την απόλαυση των χειμερινών τους διακοπών.

1.4 Εφαρμογές για Χιονοδρομικά Κέντρα

Οι εφαρμογές για κινητές συσκευές έχουν αρχίσει να παίζουν καθοριστικό ρόλο στη βελτίωση της εμπειρίας των επισκεπτών στα χιονοδρομικά κέντρα. Μέσω της άμεσης πρόσβασης σε χρήσιμες πληροφορίες, όπως οι καιρικές συνθήκες, η κατάσταση των πιστών και οι διαθέσιμες υπηρεσίες, οι εφαρμογές αυτές διευκολύνουν σημαντικά τον προγραμματισμό και την οργάνωση της επίσκεψης.

Η ανάπτυξη της εφαρμογής iOS με την ονομασία “SnowHub” επιδιώκει να καλύψει την ανάγκη για ένα ολοκληρωμένο ψηφιακό εργαλείο που να προσφέρει στους επισκέπτες των χιονοδρομικών κέντρων στην Ελλάδα όλες τις απαραίτητες πληροφορίες σε πραγματικό χρόνο. Με τη χρήση της εφαρμογής, οι χρήστες μπορούν να προγραμματίσουν καλύτερα τις δραστηριότητές τους, να ενημερώνονται άμεσα για τυχόν αλλαγές στις συνθήκες και να αξιοποιούν προσφορές και υπηρεσίες των χιονοδρομικών κέντρων.

Παρόλο που οι εφαρμογές αυτού του τύπου προσφέρουν σημαντικά οφέλη, η ανάπτυξή τους συνοδεύεται και από ορισμένες προκλήσεις. Ένα από τα βασικά ζητήματα είναι η ανάγκη για διαρκή και ακριβή ενημέρωση των δεδομένων, ώστε οι χρήστες να λαμβάνουν αξιόπιστες πληροφορίες. Η εφαρμογή SnowHub έχει σχεδιαστεί με στόχο να παρέχει ακριβείς και ενημερωμένες πληροφορίες, διασφαλίζοντας ότι οι χρήστες θα έχουν πρόσβαση σε όλα όσα χρειάζονται για μια ασφαλή και απολαυστική εμπειρία

Επιπλέον, η ποιότητα της εμπειρίας χρήστη είναι καίριας σημασίας. Η SnowHub αναπτύχθηκε με τη γλώσσα προγραμματισμού Swift, προσφέροντας υψηλή απόδοση και ευχρηστία. Η εφαρμογή έχει σχεδιαστεί με γνώμονα τις ανάγκες των επισκεπτών στα χιονοδρομικά κέντρα, με έμφαση στη διαισθητική πλοήγηση και την εύκολη πρόσβαση στις πιο σημαντικές πληροφορίες.

Συνολικά, η εφαρμογή SnowHub έχει τη δυναμική να βελτιώσει σημαντικά την εμπειρία των επισκεπτών στα χιονοδρομικά κέντρα, προσφέροντας τους ένα πολύτιμο εργαλείο για τον προγραμματισμό και την οργάνωση των δραστηριοτήτων τους, ενισχύοντας έτσι την απόλαυση και την ασφάλειά τους κατά την επίσκεψή τους στα χιονοδρομικά κέντρα της Ελλάδας..

1.5 Χειμερινός Τουρισμός την περίοδο του COVID-19

Η πανδημία του COVID-19 είχε βαθιά επίδραση σε πολλούς τομείς της καθημερινότητας, και ο χειμερινός τουρισμός δεν αποτέλεσε εξαίρεση. Τα χιονοδρομικά κέντρα, τα οποία αποτελούν βασικούς προορισμούς για τους λάτρεις των χειμερινών σπορ, αντιμετώπισαν σοβαρές προκλήσεις κατά τη διάρκεια αυτής της περιόδου. Η αυστηρή τήρηση των μέτρων κοινωνικής αποστασιοποίησης, τα περιοριστικά μέτρα μετακίνησης, και οι φόβοι για την εξάπλωση του ιού οδήγησαν σε δραματική μείωση της επισκεψιμότητας στα χιονοδρομικά κέντρα σε όλη την Ελλάδα.

Η αναστολή των δραστηριοτήτων και η μειωμένη ζήτηση για χειμερινά σπορ είχαν ως αποτέλεσμα την ακύρωση πολλών προγραμματισμένων επισκέψεων και εκδηλώσεων, κάτι που επηρέασε όχι μόνο τα ίδια τα χιονοδρομικά κέντρα, αλλά και την ευρύτερη οικονομία της περιοχής που στηρίζεται στον τουρισμό. Οι λάτρεις των σπορ αυτών βρέθηκαν αντιμέτωποι με την αδυναμία να απολαύσουν τις αγαπημένες τους δραστηριότητες, γεγονός που δημιούργησε την ανάγκη για νέα μέσα επικοινωνίας και ενημέρωσης σχετικά με τις συνθήκες και τη λειτουργία των κέντρων.

Η ανάγκη για μια σύγχρονη και αξιόπιστη πλατφόρμα που θα παρέχει στους χρήστες όλες τις απαραίτητες πληροφορίες σε πραγματικό χρόνο έγινε πιο επιτακτική από ποτέ. Η εφαρμογή iOS "SnowHub" δημιουργήθηκε ακριβώς για να καλύψει αυτό το κενό. Στόχος της εφαρμογής είναι να βοηθήσει τους επισκέπτες των χιονοδρομικών κέντρων, να προγραμματίσουν τις επισκέψεις τους με ασφάλεια και αποτελεσματικότητα, προσφέροντάς τους άμεση πρόσβαση σε πληροφορίες σχετικά με τις καιρικές συνθήκες, την κατάσταση των πιστών, και τις διαθέσιμες υπηρεσίες.

Η ανάπτυξη της εφαρμογής SnowHub προέκυψε ως απάντηση στις ανάγκες που δημιουργήθηκαν κατά τη διάρκεια της πανδημίας και φιλοδοξεί να βελτιώσει την εμπειρία των επισκεπτών στα χιονοδρομικά κέντρα, τόσο κατά την περίοδο του COVID-19 όσο και μετά από αυτήν.

1.6 Κίνητρο Πτυχιακής Εργασίας

Η δημιουργία της εφαρμογής iOS SnowHub για τα χιονοδρομικά κέντρα στην Ελλάδα προέκυψε από την ανάγκη για έναν σύγχρονο και εύχρηστο ψηφιακό οδηγό που θα εξυπηρετεί τους λάτρεις των χειμερινών σπορ. Η έλλειψη ολοκληρωμένων εργαλείων που παρέχουν σε πραγματικό χρόνο πληροφορίες για τις καιρικές συνθήκες, την κατάσταση των πιστών και τις διαθέσιμες υπηρεσίες στα χιονοδρομικά κέντρα, αποτέλεσε το κύριο κίνητρο για την ανάπτυξη αυτής της εφαρμογής. Η πανδημία του COVID-19 και η μείωση του χειμερινού τουρισμού ανέδειξαν ακόμα περισσότερο την ανάγκη για τέτοιες λύσεις, προσφέροντας έμπνευση για την ανάπτυξη της “SnowHub”.

Η προσωπική μου εμπειρία ως προγραμματιστής iOS, σε συνδυασμό με την αγάπη μου για τα χειμερινά σπορ, με ώθησαν να αναπτύξω μια εφαρμογή που όχι μόνο καλύπτει πρακτικές ανάγκες, αλλά επίσης εμπλουτίζει την εμπειρία των επισκεπτών των χιονοδρομικών κέντρων. Μέσα από την ανάπτυξη της “SnowHub”, είχα την ευκαιρία να εφαρμόσω και να επαληθεύσω τις γνώσεις που έχω αποκτήσει κατά τη διάρκεια της καριέρας μου, ενώ παράλληλα αναπτύσσω νέες δεξιότητες που θα είναι πολύτιμες για το μέλλον.

Η εφαρμογή SnowHub δεν είναι απλώς ένα προσωπικό έργο, αλλά ένα εργαλείο που μπορεί να προσφέρει αξία σε πολλούς χρήστες. Η επιτυχής ολοκλήρωση της εφαρμογής αποτελεί για μένα ένα σημαντικό επίτευγμα, τόσο σε επαγγελματικό όσο και σε προσωπικό επίπεδο, και ενδυναμώνει την επιθυμία μου να συνεχίσω να δημιουργώ καινοτόμες λύσεις στο χώρο της τεχνολογίας. Ελπίζω ότι αυτή η εφαρμογή θα μπορούσε να χρησιμεύσει ως παράδειγμα και οδηγός για άλλους προγραμματιστές και φοιτητές που ενδιαφέρονται να ασχοληθούν με την ανάπτυξη εφαρμογών iOS, δείχνοντας πώς μπορεί κανείς να συνδυάσει την τεχνική εξειδίκευση με την πρακτική εφαρμογή σε πραγματικά έργα.

Τέλος, η ολοκλήρωση αυτής της πτυχιακής εργασίας σηματοδοτεί την επίτευξη ενός σημαντικού προσωπικού στόχου, την ολοκλήρωση των προπτυχιακών σπουδών μου. Μέσα από αυτή τη δημιουργική διαδικασία, απέκτησα πολύτιμα εφόδια και γνώσεις που θα με συνοδεύουν στην επαγγελματική μου σταδιοδρομία, αλλά και στην καθημερινή μου ενασχόληση με την επιστήμη της πληροφορικής, έναν τομέα που αποτελεί αναπόσπαστο κομμάτι της ζωής μου.

Κεφάλαιο 2ο: Τεχνολογίες και Εργαλεία

2.1 Εισαγωγή

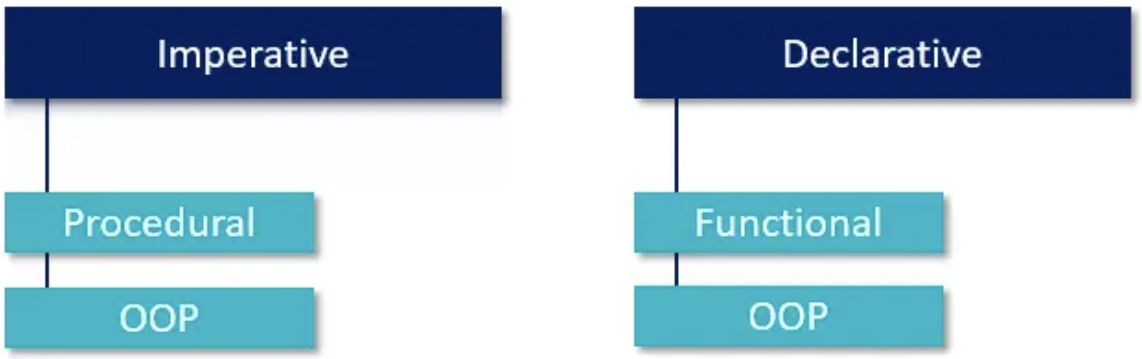
Η ανάπτυξη της εφαρμογής iOS "SnowHub" για τα χιονοδρομικά κέντρα στην Ελλάδα βασίστηκε σε μια σειρά από τεχνολογίες και εργαλεία που επέτρεψαν τη δημιουργία ενός σύγχρονου και λειτουργικού προϊόντος. Στο κεφάλαιο αυτό, θα αναλύσουμε τις βασικές τεχνολογίες που χρησιμοποιήθηκαν, συμπεριλαμβανομένης της γλώσσας προγραμματισμού Swift, του περιβάλλοντος ανάπτυξης Xcode, καθώς και των βιβλιοθηκών και εργαλείων που διευκόλυναν την ανάπτυξη της εφαρμογής.

2.1.1 Imperative και Declarative Programming

Ένα από τα κεντρικά θέματα στην ανάπτυξη της "SnowHub" ήταν η επιλογή μεταξύ imperative και declarative programming. Το imperative programming, όπως αυτό υλοποιείται με το UIKit, απαιτεί από τον προγραμματιστή να καθορίζει ρητά κάθε βήμα της διεπαφής χρήστη, ενώ το declarative programming, όπως υλοποιείται στο SwiftUI, επιτρέπει την περιγραφή της διεπαφής σε υψηλότερο επίπεδο, καθορίζοντας τι θέλουμε να συμβεί και όχι πώς θα συμβεί.[5]

Η επιλογή του SwiftUI για την ανάπτυξη της "SnowHub" επέτρεψε τη δημιουργία μιας πιο καθαρής και ευανάγνωστης βάσης κώδικα, η οποία διευκολύνει την ανάπτυξη και τη συντήρηση της εφαρμογής. Η declarative προσέγγιση ήταν ιδιαίτερα χρήσιμη για την υλοποίηση reactive χαρακτηριστικών, όπου η διεπαφή χρήστη προσαρμόζεται αυτόματα στις αλλαγές δεδομένων.

Στο Σχήμα 2.1 αναπαριστάται η βασικη διαφορά Imperative και Declarative Programming.



Σχήμα 2.1 Imperative vs. Declarative Programming

2.2 Swift και SwiftUI

Η εφαρμογή "SnowHub" αναπτύχθηκε χρησιμοποιώντας τη γλώσσα προγραμματισμού Swift, η οποία αποτελεί τη βασική γλώσσα για την ανάπτυξη native εφαρμογών για το iOS. Η Swift είναι γνωστή για την ταχύτητά της, την ασφάλεια και τη δυνατότητα διαχείρισης της μνήμης, καθιστώντας την ιδανική για την ανάπτυξη υψηλών επιδόσεων εφαρμογών.[2] [4]

Ένα από τα κύρια χαρακτηριστικά της ανάπτυξης της "SnowHub" ήταν η χρήση του SwiftUI, ενός declarative framework για τη δημιουργία user interfaces. Σε αντίθεση με το παραδοσιακό UIKit, το οποίο ακολουθεί το imperative programming μοντέλο, το SwiftUI επιτρέπει στους προγραμματιστές να περιγράφουν την εμφάνιση και τη συμπεριφορά της διεπαφής χρήστη σε δηλωτική μορφή. Αυτή η προσέγγιση καθιστά τη διαχείριση των UI elements πιο απλή και σαφή, διευκολύνοντας την ανάπτυξη reactive εφαρμογών που προσαρμόζονται δυναμικά στις αλλαγές της κατάστασης.

Η επιλογή του SwiftUI αντί του UIKit δεν ήταν μόνο μια τεχνική απόφαση, αλλά και μια στρατηγική για την επίτευξη ενός πιο μοντέρνου και ευέλικτου UI, το οποίο μπορεί να αντιδράσει άμεσα στις αλλαγές των δεδομένων και της αλληλεπίδρασης του χρήστη με την εφαρμογή.

Στο Σχήμα 2.2 παρουσιάζονται τα λογότυπα της Swift και της SwiftUI αντιστοιχα.

Σχήμα 2.2 Λογότυπα της Swift και της SwiftUI.

2.3 Εργαλεία

2.3.1 Xcode

Για την ανάπτυξη της "SnowHub", χρησιμοποιήθηκε το Xcode, το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) της Apple για macOS. Το Xcode παρέχει όλα τα απαραίτητα εργαλεία για τη δημιουργία εφαρμογών iOS, από τη συγγραφή και τη διαχείριση του κώδικα, μέχρι τη σύνταξη και τη δοκιμή της εφαρμογής σε προσομοιωτές και πραγματικές συσκευές.[3]

Το Xcode προσφέρει ενσωματωμένη υποστήριξη για Swift και SwiftUI, διευκολύνοντας την ανάπτυξη της εφαρμογής με εργαλεία όπως το Interface Builder για τη δημιουργία UI, το Instruments για τον εντοπισμό και τη βελτιστοποίηση της απόδοσης, και το XCTest για την αυτοματοποίηση των δοκιμών.

2.3.2 Postman

Κατά την ανάπτυξη της "SnowHub", ήταν απαραίτητο να δοκιμαστούν τα APIs που χρησιμοποιήθηκαν για την παροχή δεδομένων στην εφαρμογή. Για τον σκοπό αυτό, χρησιμοποιήθηκε το Postman, ένα εργαλείο ανάπτυξης που επιτρέπει την αποστολή και την ανάλυση αιτημάτων HTTP. Το Postman διευκόλυνε την επικύρωση της επικοινωνίας μεταξύ του frontend της εφαρμογής και των εξωτερικών υπηρεσιών, επιτρέποντας τον έλεγχο των δεδομένων που επιστρέφονται από τα APIs. Με το Postman, μπορέσαμε να διασφαλίσουμε ότι τα δεδομένα που λαμβάνει η "SnowHub" είναι ακριβή και αξιόπιστα, γεγονός που συνέβαλε στη σταθερότητα και την ακρίβεια της εφαρμογής.

Στο Σχήμα 2.3 παρουσιάζεται το λογότυπο του Postman.

Σχήμα 2.3 Λογότυπα Postman.

2.3.3 Git και GitHub

Για την παρακολούθηση της ανάπτυξης και τη διαχείριση των αλλαγών στον κώδικα της "SnowHub", χρησιμοποιήθηκε το Git, ένα διανεμημένο σύστημα ελέγχου εκδόσεων. Το Git επέτρεψε την αποτελεσματική διαχείριση των εκδόσεων της εφαρμογής και τη συνεργασία με άλλους προγραμματιστές. Η αποθήκευση του κώδικα έγινε μέσω της πλατφόρμας GitHub, η οποία παρείχε επιπλέον εργαλεία συνεργασίας, όπως τα pull requests και τα issues, για την αναθεώρηση και τον έλεγχο των αλλαγών.[7][8]

Στο Σχήμα 2.4 παρουσιάζονται τα λογότυπα του git και του GitHub αντιστοιχα.

Σχήμα 2.4 Λογότυπα του git και του GitHub.

2.4 Βιβλιοθήκες και Frameworks

Κατά την ανάπτυξη της "SnowHub", χρησιμοποιήθηκαν αρκετές βιβλιοθήκες και frameworks για την ενίσχυση της λειτουργικότητας της εφαρμογής. Παρακάτω αναφέρονται οι βασικές βιβλιοθήκες που χρησιμοποιήθηκαν

2.4.1 Alamofire

Μια δημοφιλής βιβλιοθήκη για την εκτέλεση δικτυακών αιτημάτων HTTP. Το Alamofire διευκόλυνε την επικοινωνία με εξωτερικά APIs, επιτρέποντας την ασφαλή και αποδοτική ανταλλαγή δεδομένων, κάτι που ήταν κρίσιμο για τη λειτουργία της "SnowHub".[13]

Στο Σχήμα 2.5 παρουσιάζεται το λογότυπο του Alamofire.

Σχήμα 2.5 Λογότυπα Alamofire.

2.4.2 Kingfisher

Αυτή η βιβλιοθήκη χρησιμοποιήθηκε για το fetching και caching των εικόνων. Το Kingfisher βελτίωσε την απόδοση της εφαρμογής, διασφαλίζοντας την ταχεία φόρτωση εικόνων και την αποφυγή περιττών δικτυακών αιτημάτων, προσφέροντας μια ομαλή εμπειρία χρήστη.[14]

Στο Σχήμα 2.6 παρουσιάζεται το λογότυπο του Kingfisher.

Σχήμα 2.6 Λογότυπα Kingfisher.

2.4.3 IOS Security Suite

Μια βιβλιοθήκη σχεδιασμένη για την ενίσχυση της ασφάλειας της εφαρμογής, βασισμένη στις οδηγίες του OWASP. Το IOSSecuritySuite χρησιμοποιήθηκε για την ανίχνευση απειλών, όπως jailbreaking, debugging, και την προστασία των δεδομένων της εφαρμογής από επιθέσεις.[15]

Στο Σχήμα 2.7 παρουσιάζεται το λογότυπο του iOS Security Suite.

Σχήμα 2.7 Λογότυπα iOS Security Suite.

2.4.4 swift-crypto

Αυτή η βιβλιοθήκη παρέχει κρυπτογραφικές λειτουργίες υψηλής απόδοσης και χρησιμοποιήθηκε για την εξασφάλιση της ασφάλειας των δεδομένων στην "SnowHub". Το swift-crypto επιτρέπει την ασφαλή διαχείριση και αποθήκευση των δεδομένων, προστατεύοντας τα προσωπικά στοιχεία των χρηστών.[17]

2.4.5 Firebase

Το Firebase χρησιμοποιήθηκε για την ενσωμάτωση αναλυτικών στοιχείων (analytics) και την παρακολούθηση σφαλμάτων (crashlytics) στην "SnowHub". Με το Firebase Analytics, μπορέσαμε να κατανοήσουμε τη συμπεριφορά των χρηστών και να βελτιστοποιήσουμε την εφαρμογή, ενώ το Crashlytics μας επέτρεψε να εντοπίσουμε και να διορθώσουμε γρήγορα τυχόν προβλήματα.[16]

Στο Σχήμα 2.7 παρουσιάζεται το λογότυπο της Firebase.

Σχήμα 2.8 Λογότυπα της Firebase.

2.5 Ευχρηστία και Apple Developer Guidelines

Μετά τον καθορισμό των απαιτήσεων της εφαρμογής, είναι κρίσιμο να ληφθούν ενημερωμένες σχεδιαστικές αποφάσεις που θα οδηγήσουν στο βέλτιστο δυνατό αποτέλεσμα. Στόχος είναι η δημιουργία μιας εφαρμογής που θα διακρίνεται για την ευχρηστία και τη λειτουργικότητά της.

2.5.1 Βασικά Χαρακτηριστικά Σχεδίασης

Κατά τον σχεδιασμό μιας εφαρμογής για iOS, είναι ζωτικής σημασίας να κατανοηθούν τα βασικά χαρακτηριστικά που κάνουν την εμπειρία χρήσης μιας iOS συσκευής μοναδική. Αυτά τα χαρακτηριστικά μπορεί να διαφέρουν από συσκευή σε συσκευή, επομένως η κατανόησή τους επιτρέπει τη λήψη ενημερωμένων αποφάσεων κατά τον σχεδιασμό, διασφαλίζοντας ότι η εφαρμογή θα εκτιμηθεί από τον χρήστη και θα προσφέρει μια απολαυστική εμπειρία. Τα βασικά χαρακτηριστικά που πρέπει να ληφθούν υπόψη περιλαμβάνουν:

- **Οθόνη (Display):** Τα iPhone διαθέτουν συνήθως οθόνες μεσαίου μεγέθους και υψηλής ανάλυσης. Για παράδειγμα, η μικρότερη οθόνη iPhone έχει διαγώνιο 3.5 ίντσες (8,9 εκ.) και ανάλυση 640 x 960, ενώ η μεγαλύτερη έχει διαγώνιο 6.7 ίντσες (17 εκ.) και ανάλυση 1284 x 2778.
- **Εργονομία (Ergonomics)**: Οι χρήστες συνήθως κρατούν το iPhone με το ένα χέρι ή σπανιότερα με τα δύο, και εναλλάσσουν τον προσανατολισμό από κάθετο (portrait) σε οριζόντιο (landscape) και αντίστροφα, ανάλογα με την περίπτωση. Κατά τη χρήση, η απόσταση του χρήστη από τη συσκευή δεν υπερβαίνει συνήθως τα 0.6 μέτρα.
- **Είσοδοι (Inputs)**: Η συσκευή υποστηρίζει πολλαπλές χειρονομίες αφής, πληκτρολόγια και φωνητικές εντολές, επιτρέποντας στους χρήστες να πραγματοποιούν ενέργειες και να επιτυγχάνουν στόχους ενώ βρίσκονται εν κινήσει. Επιπλέον, δεδομένα από αισθητήρες όπως το γυροσκόπιο και το επιταχυνσιόμετρο, καθώς και πληροφορίες τοποθεσίας, διευκολύνουν λειτουργίες που σχετίζονται με την κίνηση και την τοποθεσία του χρήστη.
- **Αλληλεπιδράσεις εφαρμογών (App Interactions)**: Οι χρήστες συχνά εναλλάσσονται μεταξύ πολλών ανοιχτών εφαρμογών, είτε η χρήση τους διαρκεί λίγα λεπτά είτε περισσότερο. Είναι σημαντικό να ληφθεί υπόψη η διαχείριση αυτών των εναλλαγών μεταξύ προσκηνίου (foreground mode) και παρασκηνίου (background mode) κατά τον σχεδιασμό της εφαρμογής.
- **Λειτουργίες συστήματος (System Features)**: Το iOS προσφέρει πληθώρα λειτουργιών που βοηθούν τους χρήστες να αλληλεπιδρούν τόσο με την εφαρμογή όσο και με το σύστημα με τρόπους που τους είναι οικείοι. Αυτές οι λειτουργίες περιλαμβάνουν τα Widgets, τις Γρήγορες Ενέργειες από την Αρχική Οθόνη **(Home Screen Quick Actions)**, το Spotlight, τις Συντομεύσεις **(Shortcuts)** και τις Προβολές Δραστηριοτήτων **(Activity Views).**

2.5.2 Βέλτιστες Πρακτικές Σχεδιασμού

Για να προσφέρουν εξαιρετικές εμπειρίες χρήστη, οι κορυφαίες εφαρμογές αξιοποιούν στο έπακρο τις δυνατότητες τόσο του λειτουργικού συστήματος όσο και της συσκευής. Αυτό απαιτεί προσεκτική προτεραιοποίηση και σχεδιασμό, σύμφωνα με τις ακόλουθες κατευθυντήριες γραμμές:

- Εστίαση στις κύριες λειτουργίες: Η διεπαφή χρήστη θα πρέπει να είναι απλή και ευδιάκριτη, επιτρέποντας στον χρήστη να επικεντρώνεται στους βασικούς στόχους του. Αυτό επιτυγχάνεται με τη μείωση των διαθέσιμων κουμπιών και τρόπων αλληλεπίδρασης. Οτιδήποτε είναι δευτερεύον πρέπει να παραμένει προσβάσιμο με όσο το δυνατόν λιγότερες και απλούστερες ενέργειες.

- Ομαλή προσαρμογή στις αλλαγές περιβάλλοντος: Η διεπαφή πρέπει να προσαρμόζεται απρόσκοπτα σε μεταβολές, όπως η αλλαγή προσανατολισμού της συσκευής ή η εναλλαγή μεταξύ σκοτεινής και φωτεινής λειτουργίας. Παράλληλα, είναι σημαντικό να παρέχεται στον χρήστη η δυνατότητα να εξατομικεύει την εμφάνιση της διεπαφής, ώστε να ταιριάζει στις προτιμήσεις του και στις συνήθειές του.
- Σχεδιασμός με γνώμονα τη χρήση της συσκευής: Ο τρόπος με τον οποίο οι χρήστες κρατούν τη συσκευή πρέπει να λαμβάνεται υπόψη κατά τον σχεδιασμό της διεπαφής. Για παράδειγμα, τα κουμπιά και τα εργαλεία αλληλεπίδρασης θα πρέπει να τοποθετούνται σε περιοχές που είναι εύκολα προσβάσιμες, όπως τα μεσαία και χαμηλά τμήματα της οθόνης.
- Εμπλουτισμός της εμπειρίας με ενσωματωμένες λειτουργίες: Η αξιοποίηση λειτουργιών που παρέχονται από το σύστημα, όπως συστήματα πληρωμών, βιομετρική ταυτοποίηση ή πληροφορίες τοποθεσίας, μπορεί να βελτιώσει σημαντικά την εμπειρία του χρήστη. Φυσικά, η χρήση αυτών των λειτουργιών προϋποθέτει τη συναίνεση του χρήστη και την προσεκτική διαχείριση των προσωπικών του δεδομένων.

Αυτές οι πρακτικές αποτελούν το θεμέλιο για την επιτυχή σχεδίαση μιας διεπαφής χρήστη. Στα επόμενα στάδια, θα προχωρήσουμε σε βαθύτερη ανάλυση συγκεκριμένων θεμάτων σχεδίασης, εξετάζοντας τις θεμελιώδεις τεχνολογίες, καθώς και τις βέλτιστες πρακτικές και πρότυπα.

2.5.3 Θεμελιώδεις Τεχνολογίες στη SnowHub

Σε αυτό το κεφάλαιο, εξετάζουμε τις βασικές τεχνολογίες και στοιχεία που συμβάλλουν στη δημιουργία εφαρμογών που είναι πλούσιες σε δυνατότητες και ευχάριστες στη χρήση. Για την ανάλυση αυτή, θα επικεντρωθούμε σε τεχνολογίες που χρησιμοποιούνται στο SnowHub, αντλώντας από τις κατευθυντήριες γραμμές της Apple.

2.5.3.1 Λογότυπο Εφαρμογής, Εικονίδια και SF Symbols

Η πρώτη επαφή του χρήστη με μια εφαρμογή είναι το εικονίδιό της. Αυτό το μικρό γραφικό στοιχείο λειτουργεί σαν μια βιτρίνα, παρουσιάζοντας την ταυτότητα και τον σκοπό της εφαρμογής με μια ματιά. Ένα εικονίδιο που αποτυπώνει με ακρίβεια και απλότητα την ουσία της εφαρμογής είναι απαραίτητο για να δημιουργήσει μια θετική πρώτη εντύπωση. Γι' αυτό, τόσο το λογότυπο όσο και όλα τα εικονίδια που χρησιμοποιούνται μέσα στην εφαρμογή πρέπει να σχεδιαστούν με ιδιαίτερη προσοχή.

2.5.3.2 Λογότυπο Εφαρμογής

Ένα καλοσχεδιασμένο λογότυπο είναι απαραίτητο για κάθε εφαρμογή. Ωστόσο, είναι σημαντικό να διασφαλιστεί ότι το λογότυπο αυτό παραμένει συμβατό με όλες τις πλατφόρμες iOS, διατηρώντας παράλληλα τη μοναδικότητά του και επιτυγχάνοντας οπτική συνοχή. Για να επιτευχθούν αυτοί οι στόχοι, η Apple προτείνει τις ακόλουθες πρακτικές:

- **Έμφαση στην απλότητα**: Τα πιο επιτυχημένα λογότυπα είναι συνήθως τα απλά, καθώς είναι πιο κατανοητά από τον χρήστη. Η αποφυγή πολύπλοκων λεπτομερειών βοηθά στη διατήρηση της καθαρότητας του λογότυπου, ιδιαίτερα στις μικρότερες οθόνες όπου μπορεί να είναι δύσκολο για τον χρήστη να διακρίνει μικρά στοιχεία. Επίσης, το παρασκήνιο του λογότυπου πρέπει να είναι λιτό ώστε να μην αποσπά την προσοχή του χρήστη. Παραδείγματα τέτοιων λογοτύπων που έχουν γίνει trademarks της βιομηχανίας φαίνονται στο Σχήμα 2.3.
- **Αποφυγή χρήσης κειμένου**: Το κείμενο στο λογότυπο πρέπει να αποφεύγεται, εκτός εάν είναι απολύτως απαραίτητο για την εφαρμογή. Συνήθως, το κείμενο στα λογότυπα είναι πολύ μικρό

για να διαβαστεί εύκολα και, δεδομένου ότι το όνομα της εφαρμογής εμφανίζεται κάτω από το εικονίδιο, η προσθήκη κειμένου στο λογότυπο καθίσταται περιττή.

- **Χρήση πρωτότυπου γραφικού και όχι φωτογραφίας**: Οι φωτογραφίες περιέχουν συνήθως πολλές λεπτομέρειες, που είναι δύσκολο να αναγνωριστούν σε μικρότερες συσκευές. Επίσης, η χρήση ενός στιγμιότυπου οθόνης από το εσωτερικό της εφαρμογής ως λογότυπο θεωρείται κακή πρακτική και πρέπει να αποφεύγεται. Τέλος, είναι σημαντικό να αποφεύγεται η χρήση γραφικών που μιμούνται λογότυπα άλλων εφαρμογών της Apple, καθώς αυτό είναι παράνομο.

Σχήμα 2.3 Λογότυπα δημοφιλών εφαρμογών

Οι σχεδιαστικές προδιαγραφές και οι διαστάσεις του λογότυπου της εφαρμογής παρουσιάζονται παρακάτω στα Σχήματα 2.4 και 2.5 αντίστοιχα.

Platform	Layers	Transparency	Asset shape
iOS, iPadOS	Single	No	Square
macOS	Single	Yes, as appropriate	Square with rounded corners
tvOS	Multiple	No	Rectangle
watchOS	Single	No	Square

Σχήμα 2.4 Τεχνικά χαρακτηριστικά λογότυπου εφαρμογής

@2x (pixels)	@3x (pixels) iPhone only	Usage
120x120	180x180	Home Screen on iPhone
167x167	–	Home Screen on iPad Pro
152x152	–	Home Screen on iPad, iPad mini
80x80	120x120	Spotlight on iPhone, iPad Pro, iPad, iPad mini
58x58	87x87	Settings on iPhone, iPad Pro, iPad, iPad mini
76x76	114x114	Notifications on iPhone, iPad Pro, iPad, iPad mini

Σχήμα 2.5 Μεγέθη λογότυπου εφαρμογής για συσκευές iPhone

2.5.3.3 Εικονίδια Εφαρμογής SnowHub

Τα εικονίδια είναι κρίσιμα για τη λειτουργικότητα μιας εφαρμογής, καθώς καθοδηγούν τον χρήστη στις δυνατότητες και τις ενέργειες που μπορεί να εκτελέσει.

Ενώ τα λογότυπα της εφαρμογής πρέπει να είναι διακριτικά και να αντικατοπτρίζουν τον χαρακτήρα της, τα εικονίδια χρειάζεται να είναι σχεδιασμένα με απλότητα και ευκολία αναγνώρισης. Αυτό διευκολύνει τους χρήστες να τα αναγνωρίζουν και να τα χρησιμοποιούν με ευκολία, ακόμη και αν τα έχουν δει και σε άλλες εφαρμογές. Για παράδειγμα, ένα εικονίδιο που αναπαριστά μια φωτογραφική μηχανή πρέπει να είναι άμεσα κατανοητό ότι σχετίζεται με τη χρήση της κάμερας της συσκευής, καθιστώντας τη διαδραστικότητα της εφαρμογής διαισθητική.

Η Apple υπογραμμίζει τη σημασία της διατήρησης της συνέπειας στα εικονίδια σε ολόκληρη την εφαρμογή. Αυτό σημαίνει ότι τα εικονίδια πρέπει να διατηρούν παρόμοιο μέγεθος, στυλ σχεδίασης, πάχος και προοπτική. Επίσης, προτείνεται η εναρμόνιση των εικονιδίων με το κείμενο που τα συνοδεύει, για να επιτευχθεί μια καθαρή και ισορροπημένη εμφάνιση. Όπου απαιτείται, τα εικονίδια θα πρέπει να προσφέρονται σε διαφορετικές παραλλαγές, όπως σε ενεργή και μη ενεργή κατάσταση.

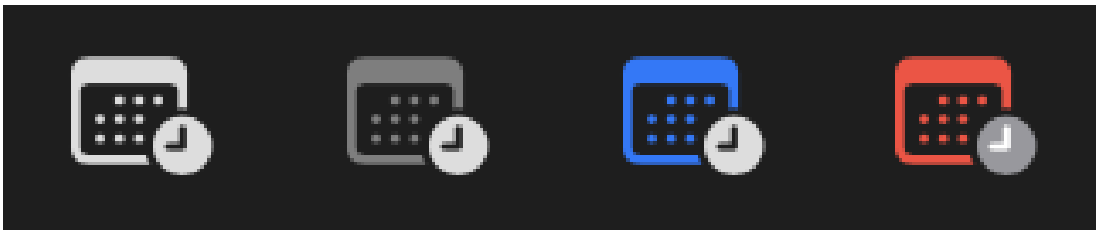
Από τεχνική άποψη, η Apple συνιστά τα εικονίδια να δημιουργούνται σε διανυσματικές μορφές (όπως PDF ή SVG), έτσι ώστε να μπορούν να προσαρμόζονται αυτόματα σε διαφορετικά μεγέθη χωρίς απώλεια ποιότητας. Αν αυτό δεν είναι δυνατό, τότε τα εικονίδια πρέπει να σχεδιάζονται σε πολλαπλές αναλύσεις, τόσο για υψηλής όσο και για χαμηλής ανάλυσης οθόνες, ώστε να διατηρείται η ευκρίνεια και η ποιότητα της εικόνας σε όλες τις συσκευές.

2.5.3.4 SF Symbols

Τα SF Symbols είναι ένα σύνολο από χιλιάδες σύμβολα (εικονίδια) που έχουν σχεδιαστεί σύμφωνα με όλες τις προδιαγραφές που αναφέρθηκαν προηγουμένως. Ένα από τα πιο σημαντικά χαρακτηριστικά τους είναι η υψηλή δυνατότητα παραμετροποίησης, γεγονός που τα καθιστά εξαιρετικά ευέλικτα για την κάλυψη διαφόρων αναγκών. Οι κύριοι τρόποι παραμετροποίησης περιλαμβάνουν:

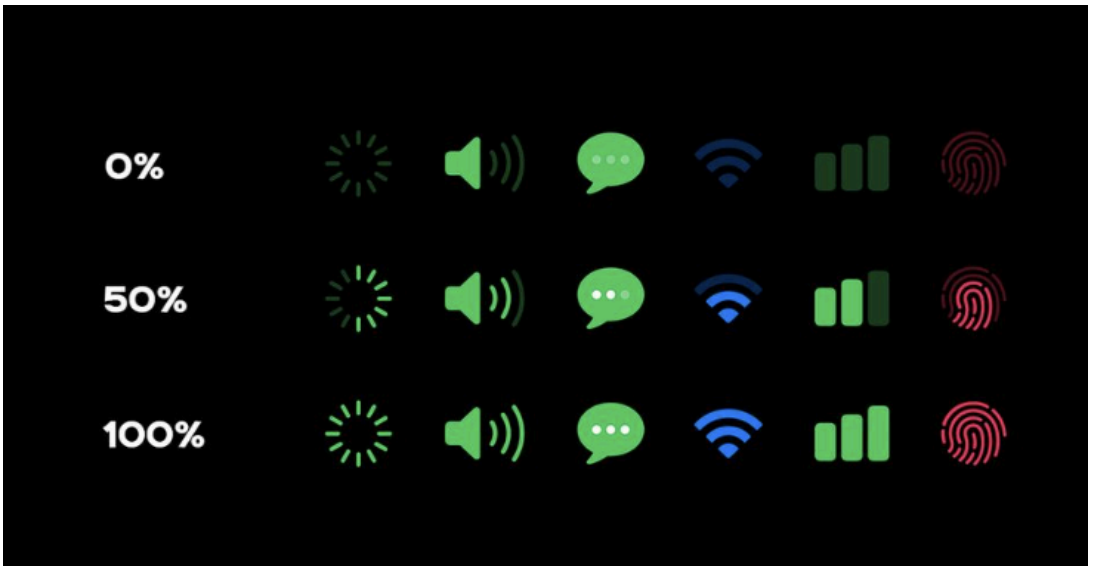
- **Λειτουργίες Απόδοσης (Rendering Modes)**: Υπάρχουν τέσσερις διαθέσιμες λειτουργίες απόδοσης που μπορούν να εφαρμοστούν στα σύμβολα. Τα σύμβολα διαχωρίζονται σε διάφορα επίπεδα (layers), στα οποία μπορεί να εφαρμοστεί η επιθυμητή λειτουργία απόδοσης,

αλλάζοντας έτσι την εμφάνιση του εικονιδίου. Οι διαθέσιμες λειτουργίες είναι η μονοχρωματική, η ιεραρχική, η παλέτα και η πολύχρωμη απόδοση. Για παράδειγμα, το σύμβολο calendar.badge.clock χρησιμοποιεί τη δυνατότητα μεταβλητού χρώματος για να απεικονίσει τις διαφορετικες εκδοχές του calendar όπως φαίνεται στο Σχήμα 2.6.



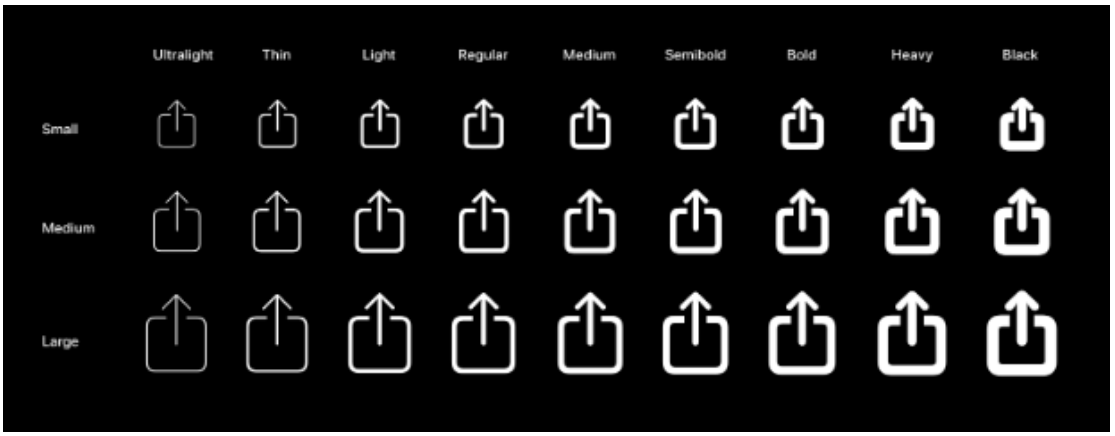
Σχήμα 2.6 calendar.badge.clock με τη χρήση rendering mode

- **Μεταβλητό Χρώμα (Variable Color)**: Αυτή η λειτουργία επιτρέπει σε ένα σύμβολο να αλλάζει ένα ή περισσότερα από τα χαρακτηριστικά του, ανεξάρτητα από τη λειτουργία απόδοσης που έχει επιλεγεί. Η αλλαγή μπορεί να ρυθμιστεί ως ποσοστό (0% - 100%) και να εφαρμοστεί σε ένα ή περισσότερα επίπεδα του εικονιδίου. Παρακάτω παρουσιάζονται μερικά παραδείγματα στο Σχήμα 2.7.



Σχήμα 2.7 πολλαπλά sf symbols με τη χρήση variable color

- **Weight and Scales**: Τα SF Symbols προσφέρουν διάφορα επίπεδα βάρους, παρόμοια με αυτά της γραμματοσειράς San Francisco, καθώς και τρία μεγέθη (small, medium, large) που επιτρέπουν την προσαρμογή της εμφάνισής τους ανάλογα με τις ανάγκες της εφαρμογής. Στο Σχήμα 2.8 παρουσιάζονται τα διαφορετικά μεγέθη και βάρη των SF Symbols.



Σχήμα 2.8 πολλαπλά βάρη και μεγεθύνσεις (weight & scales)

- **Κινήσεις(Animations)**: Οι κινήσεις (animations) που υποστηρίζονται από τα SF Symbols προσφέρουν ζωντάνια και εκφραστικότητα στην εφαρμογή, βελτιώνοντας σημαντικά την εμπειρία χρήστη. Τα animations μπορούν να εφαρμοστούν σε οποιοδήποτε σύμβολο, ανεξάρτητα από άλλες παραμετροποιήσεις που έχουν ήδη εφαρμοστεί. Το λειτουργικό σύστημα παρέχει πλήρη έλεγχο στον προγραμματιστή, ο οποίος μπορεί να καθορίσει τη διάρκεια, τον αριθμό επαναλήψεων, το σημείο εκκίνησης και λήξης, καθώς και τις συνθήκες υπό τις οποίες το animation θα εκτελεστεί. Για αναλυτικές πληροφορίες και παραδείγματα διαφόρων ειδών animation, συνιστάται η αναφορά στο επίσημο documentation της Apple .

Με τη χρήση αυτών των δυνατοτήτων, τα SF Symbols[15] προσφέρουν ευελιξία στη δημιουργία εικονιδίων και animations, προσδίδοντας στο SnowHub μια σύγχρονη και δυναμική αισθητική που ενισχύει την εμπειρία του χρήστη.

2.5.3.5 Light & Dark Mode

Σήμερα, πολλοί χρήστες επιλέγουν το dark mode ως προεπιλεγμένο στυλ διεπαφής στις συσκευές τους. Έτσι, όταν κάποιος χρησιμοποιεί το SnowHub σε dark mode, αναμένει η εφαρμογή να προσφέρει μια εμπειρία προσαρμοσμένη σε αυτό το στυλ. Το dark mode, σε αντίθεση με το light mode, χρησιμοποιεί μια σκοτεινή χρωματική παλέτα που διευκολύνει την ανάγνωση και μειώνει την καταπόνηση των ματιών, ιδιαίτερα σε περιβάλλοντα με χαμηλό φωτισμό, κάνοντας τη χρήση της εφαρμογής πιο άνετη.

Η Apple τονίζει τη σημασία της ομαλής λειτουργίας της εφαρμογής τόσο στο light όσο και στο dark mode, ειδικά καθώς οι χρήστες μπορεί να επιλέξουν να αλλάζουν δυναμικά μεταξύ αυτών των δύο στυλ, ανάλογα με τις συνθήκες φωτισμού. Είναι σημαντικό το SnowHub να μπορεί να προσαρμόζεται άμεσα σε αυτές τις αλλαγές, ακόμα και αν η εφαρμογή είναι ανοιχτή. Πρέπει να δοθεί ιδιαίτερη προσοχή στην αντίθεση μεταξύ του φόντου και των βασικών στοιχείων της εφαρμογής (όπως ετικέτες και κουμπιά), ώστε όλες οι πληροφορίες να είναι ευδιάκριτες σε dark mode.

Η Apple προτείνει να μην υπάρχει ξεχωριστή ρύθμιση εντός της εφαρμογής για εναλλαγή μεταξύ light και dark mode. Αντίθετα, το SnowHub θα πρέπει να ακολουθεί την επιλογή που έχει κάνει ο χρήστης σε επίπεδο συστήματος, προσαρμόζοντας αυτόματα το στυλ διεπαφής σε πραγματικό χρόνο. Σε ειδικές περιπτώσεις, και ανάλογα με τη φύση της εφαρμογής, μπορεί να επιλέγεται μόνο το dark mode, εάν αυτό ταιριάζει καλύτερα με την αισθητική και το περιεχόμενο της εφαρμογής. Ωστόσο, αυτή η επιλογή θα πρέπει να γίνεται προσεκτικά, καθώς περιορίζει τις δυνατότητες προσαρμογής για τον χρήστη.

Χρησιμοποιώντας τα χρώματα και τα εικονίδια του συστήματος (System Colors, SF Symbols), η μετάβαση μεταξύ light και dark mode γίνεται αυτόματα από το λειτουργικό σύστημα, απαιτώντας ελάχιστη πρόσθετη προσπάθεια από τον προγραμματιστή. Ωστόσο, αν χρησιμοποιούνται προσαρμοσμένα (custom) χρώματα και εικόνες, θα πρέπει να δοθεί ιδιαίτερη προσοχή ώστε να διασφαλιστεί μια συνεπής και ομαλή εμπειρία χρήσης.

Για συσκευές iOS, η χρήση των System Colors είναι ιδιαίτερα σημαντική για το background, δεδομένης της ιδιαίτερης λειτουργίας του dark mode σε αυτές τις συσκευές. Το σύστημα χρησιμοποιεί δύο σετ χρωματικών παλετών: το βασικό (base) και το υπερυψωμένο (elevated). Αυτά τα σετ εναλλάσσονται αυτόματα από το σύστημα για να δημιουργήσουν μια αίσθηση βάθους. Στο base set, τα χρώματα είναι πιο απαλά, κάνοντας το φόντο να φαίνεται ότι υποχωρεί, ενώ στο elevated set, τα χρώματα είναι πιο φωτεινά, δίνοντας μια αίσθηση προβολής.

Με την επιλογή των χρωμάτων του συστήματος, η εναλλαγή μεταξύ αυτών των σετ γίνεται αυτόματα, ακόμα και όταν η εφαρμογή είναι ανοιχτή. Στην περίπτωση που χρησιμοποιούνται custom χρώματα, η διαχείριση αυτής της εναλλαγής πρέπει να γίνεται από τον προγραμματιστή, προσομοιώνοντας την αντίστοιχη λειτουργικότητα του συστήματος. Για την καλύτερη προσαρμογή της εφαρμογής στα σωστά modes, η Apple παρέχει και ένα χρήσιμο βίντεο για καθοδήγηση[1][6].

2.5.3.6 Βέλτιστες Πρακτικές και Μοτίβα

Κατά την ανάπτυξη οποιασδήποτε εφαρμογής, προκύπτουν συγκεκριμένα σενάρια χρήσης που συναντώνται συχνά σε πολλές εφαρμογές. Ένα τυπικό παράδειγμα είναι η εισαγωγή δεδομένων από τον χρήστη. Για τέτοιου είδους κοινά μοτίβα, η Apple παρέχει καθοδήγηση ώστε να διασφαλίζονται οι βέλτιστες πρακτικές, βελτιώνοντας έτσι την εμπειρία του χρήστη. Η εφαρμογή SnowHub δεν αποτελεί εξαίρεση, γι' αυτό παρακάτω παρουσιάζονται μερικά βασικά μοτίβα που ενσωματώθηκαν στο SnowHub, σύμφωνα με τις οδηγίες της Apple. Τα μοτίβα παρατίθενται με τη σειρά που εμφανίζονται και στην εφαρμογή:

- **Διαχείριση Λογαριασμών**: Περιλαμβάνει τη δημιουργία, επεξεργασία και διαγραφή λογαριασμών χρήστη, καθώς και τις σχετικές ρυθμίσεις ασφαλείας και πρόσβασης.
- **Διαχείριση Ειδοποιήσεων**: Ρυθμίσεις που επιτρέπουν στον χρήστη να επιλέξει ποιες ειδοποιήσεις θα λαμβάνει και πώς αυτές θα εμφανίζονται.
- **Ρυθμίσεις**: Ενότητα που επιτρέπει την προσαρμογή της εφαρμογής στις προσωπικές προτιμήσεις του χρήστη, όπως θέματα, γλώσσα και άλλες επιλογές.

Αυτά τα μοτίβα έχουν σχεδιαστεί για να διασφαλίσουν ότι το SnowHub προσφέρει μια ομαλή και ευχάριστη εμπειρία χρήσης, ακολουθώντας τις καθιερωμένες βέλτιστες πρακτικές της Apple.

2.5.3.7 Διαχείριση Λογαριασμού

Για να μπορέσει ο χρήστης να αξιοποιήσει πλήρως τις δυνατότητες του SnowHub, είναι απαραίτητο να δημιουργήσει έναν λογαριασμό. Ο τρόπος με τον οποίο παρουσιάζουμε αυτήν την επιλογή και οι διαθέσιμες επιλογές για την εύκολη ολοκλήρωση της διαδικασίας θα επηρεάσουν άμεσα την εμπειρία του χρήστη.

Η δημιουργία λογαριασμού θα πρέπει να προσφέρεται στον χρήστη ως επιλογή μόνο όταν είναι απαραίτητη για τη λειτουργικότητα της εφαρμογής. Ιδανικά, θα πρέπει να επιτρέπεται στους χρήστες να εξερευνήσουν βασικές λειτουργίες της εφαρμογής χωρίς να απαιτείται άμεση εγγραφή. Ακόμη και αν, όπως στο SnowHub, η δημιουργία λογαριασμού είναι σχεδόν απαραίτητη, είναι σημαντικό να εξηγήσουμε στους χρήστες τα οφέλη που θα αποκτήσουν, όπως η δυνατότητα εξατομικευμένου

περιεχομένου και αγορών μέσα από την εφαρμογή. Μια καλή πρακτική είναι να καθυστερήσουμε την υποχρεωτική εγγραφή, επιτρέποντας στους χρήστες να δοκιμάσουν κάποιες λειτουργίες πριν αποφασίσουν να δημιουργήσουν λογαριασμό προσφέροντας την guest περιήγηση, ώστε να παρατηρήσουμε αν η εφαρμογή ανταποκρίνεται στις προσδοκίες τους.

Η δημιουργία λογαριασμού περιλαμβάνει τη συλλογή προσωπικών δεδομένων, και για αυτόν τον λόγο η ασφάλεια είναι υψίστης σημασίας. Στο SnowHub, η ασφάλεια διασφαλίζεται μέσω της χρήσης JWT tokens και εξωτερικής αυθεντικοποίησης μέσω backend service. Αυτή η προσέγγιση προσφέρει έναν ασφαλή και αξιόπιστο τρόπο διαχείρισης λογαριασμών, επιτρέποντας στους χρήστες να απολαμβάνουν μια ασφαλή και απρόσκοπτη εμπειρία χρήσης. Σε αντίθεση με τη χρήση άλλων μεθόδων, η εξωτερική αυθεντικοποίηση μέσω API παρέχει αυξημένη ασφάλεια και ταυτόχρονα απλοποιεί τη διαδικασία για τους χρήστες.

Επιπλέον, είναι απαραίτητο να παρέχουμε στους χρήστες τη δυνατότητα να διαγράψουν τον λογαριασμό τους, εάν το επιθυμούν. Η διαδικασία αυτή πρέπει να είναι απλή και διαφανής, ακολουθώντας το νομικό πλαίσιο που διέπει την προστασία προσωπικών δεδομένων. Ο χρήστης πρέπει να ενημερώνεται με σαφήνεια τόσο για την επιτυχή δημιουργία του λογαριασμού του όσο και για την επιτυχή διαγραφή του.

Συνολικά, η διαχείριση λογαριασμών στο SnowHub πρέπει να χαρακτηρίζεται από σαφήνεια και ασφάλεια, διασφαλίζοντας ότι οι χρήστες έχουν μια θετική και αξιόπιστη εμπειρία, είτε κατά τη δημιουργία είτε κατά τη διαγραφή των λογαριασμών τους.

2.5.3.8 Διαχείριση Ειδοποιήσεων

Οι ειδοποιήσεις αποτελούν ένα ισχυρό εργαλείο για το SnowHub, καθώς προσφέρουν στους χρήστες κρίσιμες πληροφορίες ακόμα και όταν η συσκευή τους είναι κλειδωμένη. Ωστόσο, επειδή οι ειδοποιήσεις μπορούν να διακόψουν τη ροή των δραστηριοτήτων του χρήστη, είναι απαραίτητο να διασφαλίζεται ότι αποστέλλονται τη σωστή στιγμή και με τον κατάλληλο τρόπο.

Ένας θεμελιώδης κανόνας που πρέπει να τηρείται είναι ότι η εφαρμογή χρειάζεται να έχει τη ρητή **άδεια του χρήστη για την αποστολή ειδοποιήσεων**. Αυτή η άδεια μπορεί να ρυθμιστεί μέσα από τις ρυθμίσεις της συσκευής. Είναι δικαίωμα του χρήστη να επιλέξει εάν θα λαμβάνει ειδοποιήσεις από την εφαρμογή, ανεξάρτητα από το πόσο σημαντικές θεωρούνται από την πλευρά της εφαρμογής.

Η Apple κατηγοριοποιεί τις ειδοποιήσεις σε δυο κατηγορίες:

- **Communication Notifications**. Είναι όλες οι ειδοποιήσεις που σχετίζονται με επικοινωνία (όπως τηλεφωνήματα ή μηνύματα)
- **Noncommunication Notifications**. Είναι όλα τα υπόλοιπα είδη ειδοποιήσεων.

Στο SnowHub, χρησιμοποιούμε το Firebase Messaging για την αποστολή ειδοποιήσεων, οι οποίες κατηγοριοποιούνται ως **noncommunication notifications**. Αυτές οι ειδοποιήσεις διακρίνονται σε τέσσερα επίπεδα διακοπής, τα οποία καθορίζουν τον τρόπο και τον χρόνο με τον οποίο θα παραδοθούν στον χρήστη:

- **Παθητικό (Passive)**: Αφορά ειδοποιήσεις που δεν είναι άμεσα κρίσιμες για τον χρήστη και μπορούν να προβληθούν αργότερα, όταν έχει τον χρόνο, όπως π.χ. μια πρόταση για εστιατόριο.

- **Ενεργό (Active)**: Αυτό είναι το προεπιλεγμένο επίπεδο διακοπής και αφορά ειδοποιήσεις που είναι αρκετά σημαντικές για τον χρήστη, ώστε να εκτιμήσει την άμεση παράδοσή τους χωρίς να νιώθει ότι διακόπτεται, όπως μια ενημέρωση για αλλαγή σκορ σε έναν αγώνα.
- **Χρονικής Ευαισθησίας (Time-sensitive)**: Σε αυτό το επίπεδο, η ειδοποίηση περιέχει πληροφορίες που πρέπει να γνωστοποιηθούν άμεσα στον χρήστη, ακόμα και αν αυτό διακόψει άλλες δραστηριότητες. Ένα παράδειγμα θα μπορούσε να είναι μια ειδοποίηση για την άφιξη ενός πακέτου ή μια προειδοποίηση ασφαλείας.
- **Κρίσιμες (Critical)**: Αυτές οι ειδοποιήσεις είναι ζωτικής σημασίας και σχετίζονται με την υγεία ή την ασφάλεια του χρήστη. Συνήθως αποστέλλονται από κυβερνητικές υπηρεσίες ή εφαρμογές ασφαλείας, και είναι εξαιρετικά σπάνιες.

Στο SnowHub, οι ειδοποιήσεις που αποστέλλουμε ανήκουν κυρίως στην κατηγορία των ενεργών ειδοποιήσεων, καθώς είναι σημαντικό οι χρήστες να τις λαμβάνουν άμεσα. Ωστόσο, δεν είναι τόσο κρίσιμες ώστε να υπερβαίνουν τις προτιμήσεις του χρήστη για το πώς και πότε επιθυμεί να λαμβάνει ειδοποιήσεις.

Η σωστή διαχείριση των ειδοποιήσεων αποδεικνύει τον σεβασμό προς τον χρόνο και την προσοχή του χρήστη, και αυτό είναι κάτι που το SnowHub λαμβάνει πολύ σοβαρά υπόψη.

Στο Σχήμα 2.10 φαίνεται η συμπεριφορά ειδοποιήσεων.

Interruption level	Overrides scheduled delivery	Breaks through Focus	Overrides Ring/Silent switch on iPhone and iPad
Passive	No	No	No
Active	No	No	No
Time Sensitive	Yes	Yes	No
Critical	Yes	Yes	Yes

Σχήμα 2.10 Συμπεριφορά ειδοποιήσεων

2.5.3.9 Ρυθμίσεις

Ένα από τα θετικά χαρακτηριστικά μιας εφαρμογής είναι η δυνατότητα παραμετροποίησης, που επιτρέπει στους χρήστες να προσαρμόζουν το περιεχόμενο και τις λειτουργίες σύμφωνα με τις προσωπικές τους ανάγκες. Παρόλα αυτά, οι χρήστες αναμένουν επίσης από την εφαρμογή να λειτουργεί ομαλά χωρίς την ανάγκη για πολλές ρυθμίσεις εκ μέρους τους. Συνεπώς, οι ρυθμίσεις θα πρέπει να προσφέρονται μόνο όταν είναι απαραίτητες ή προσθέτουν ουσιαστική αξία στην εμπειρία χρήσης. Είναι σημαντικό να έχουμε υπόψη ότι οι χρήστες σπάνια επισκέπτονται την οθόνη των ρυθμίσεων, επομένως αυτές πρέπει να είναι εύκολα προσβάσιμες και να χρησιμοποιούνται με μέτρο.

Μια βασική αρχή που συνιστά η Apple είναι να τοποθετούνται οι ρυθμίσεις κοντά στα σημεία της εφαρμογής όπου έχουν άμεση εφαρμογή. Για παράδειγμα, αν υπάρχει μια λίστα με περιεχόμενο που μπορεί να φιλτραριστεί σύμφωνα με τις προτιμήσεις του χρήστη, η επιλογή φιλτραρίσματος θα πρέπει να βρίσκεται ακριβώς δίπλα στη λίστα, καθιστώντας τη διαδικασία αυτή προφανή και άμεση για τον χρήστη.

Γενικά, είναι κακή πρακτική να αναγκάζεται ο χρήστης να μεταβεί στην οθόνη των ρυθμίσεων για βασικές λειτουργίες. Η οθόνη ρυθμίσεων θα πρέπει να εξυπηρετεί ρυθμίσεις που αφορούν ολόκληρη την εφαρμογή, όπως η επιλογή θέματος (theme). Αν η εφαρμογή περιλαμβάνει τέτοια οθόνη (και είναι σημαντικό να σημειωθεί ότι δεν είναι απαραίτητο να υπάρχει πάντα), πρέπει να δίνεται προσοχή στον αριθμό των διαθέσιμων επιλογών. Πολύπλοκες ή πολυάριθμες επιλογές μπορεί να προκαλέσουν σύγχυση και να αποθαρρύνουν τους χρήστες, μειώνοντας τον χρόνο που αφιερώνουν στην πραγματική χρήση της εφαρμογής.

Είναι επίσης σημαντικό να διαχωρίζονται οι ρυθμίσεις που προσφέρει το λειτουργικό σύστημα από εκείνες της ίδιας της εφαρμογής. Ρυθμίσεις που διαχειρίζονται λειτουργίες σε επίπεδο συστήματος, όπως η τοποθεσία, δεν πρέπει να επαναλαμβάνονται εντός της εφαρμογής.

Κεφάλαιο 3ο: Σχεδίαση και Υλοποίηση της Εφαρμογής

3.1 Εισαγωγή

……..

3.2 Σχεδιασμός Αρχιτεκτονικής

3.2.1 Clean Architecture & S.O.L.I.D principles

Για την ανάπτυξη της εφαρμογής “SnowHub” επιλέχθηκε η χρήση της Clean Architecture με την προσθήκη δύο επιπλέον επιπέδων, του App Layer, του Core Layer και του Resources Layer. Η συγκεκριμένη αρχιτεκτονική προτείνεται ως μια από τις πιο αποτελεσματικές αρχιτεκτονικές για την οργάνωση και τη συντήρηση σύνθετων εφαρμογών. Το κύριο πλεονέκτημα της Clean Architecture είναι ο σαφής διαχωρισμός των ευθυνών και η απομόνωση των εξαρτήσεων μεταξύ των διαφορετικών επιπέδων (layers) της εφαρμογής, επιτρέποντας ευκολότερη δοκιμή, συντήρηση και επεκτασιμότητα.[10][19]

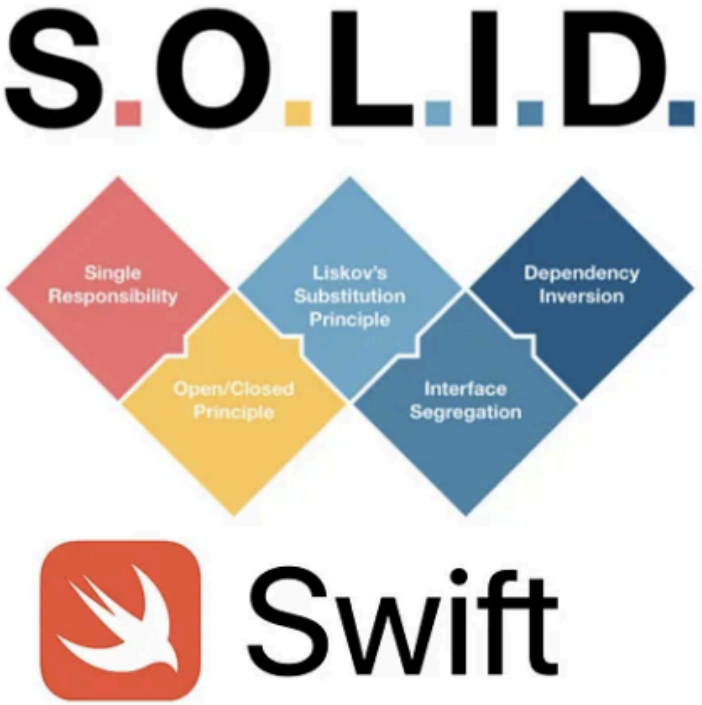
Σημαντικό κρίνεται επίσης να αναφέρουμε ότι παράλληλα με το Clean Architecture η εφαρμογή SnowHub εφαρμόζει τα S.O.L.I.D. principles. Τα S.O.L.I.D. principles είναι πέντε βασικές αρχές της αντικειμενοστραφούς σχεδίασης που στοχεύουν στη βελτίωση της συντηρησιμότητας και της επεκτασιμότητας του κώδικα[18]. Η χρήση τους στη γλώσσα Swift ακολουθεί τις ίδιες αρχές όπως και σε άλλες αντικειμενοστραφείς γλώσσες, αλλά υπάρχουν ορισμένες λεπτομέρειες που σχετίζονται με τα χαρακτηριστικά της Swift όπως αναφέρονται παρακάτω:

3.2.1.1 S.O.L.I.D principles

1. **Single Responsibility Principle (SRP) - Αρχή της Μοναδικής Ευθύνης:**
   Αυτή η αρχή δηλώνει ότι κάθε κλάση πρέπει να έχει **μία και μόνο μία ευθύνη** ή σκοπό. Στην Swift, αυτό σημαίνει ότι κάθε **class**, **struct**, ή **enum** θα πρέπει να εκτελεί μία μοναδική εργασία και να μην περιέχει λογική για πολλές λειτουργίες ταυτόχρονα.

2. **Open/Closed Principle (OCP) - Αρχή του Ανοιχτού/Κλειστού:**
   Σύμφωνα με αυτή την αρχή, οι κλάσεις πρέπει να είναι ανοιχτές για επέκταση, αλλά κλειστές για τροποποίηση. Αυτό σημαίνει ότι θα πρέπει να μπορούμε να επεκτείνουμε τη συμπεριφορά μιας κλάσης χωρίς να χρειάζεται να αλλάξουμε τον υπάρχοντα κώδικα. Στη Swift, αυτό επιτυγχάνεται με **inheritance** (**κληρονομικότητα**) ή **protocols** (**πρωτόκολλα**).
3. **Liskov Substitution Principle (LSP) - Αρχή της Αντικατάστασης της Liskov:**
   Αυτή η αρχή λέει ότι τα αντικείμενα μιας υποκλάσης πρέπει να μπορούν να χρησιμοποιηθούν αντί για τα αντικείμενα της υπερκλάσης τους χωρίς να αλλάζει η ορθότητα του προγράμματος. Στη Swift, αυτό σημαίνει ότι η υποκλάση πρέπει να τηρεί όλες τις συμβάσεις που ορίζει η υπερκλάση ή το πρωτόκολλο.
4. **Interface Segregation Principle (ISP) - Αρχή του Διαχωρισμού Διεπαφών:**
   Αυτή η αρχή υποστηρίζει ότι οι κλάσεις δεν πρέπει να αναγκάζονται να υλοποιούν μεθόδους που δεν χρησιμοποιούν. Στη Swift, αυτό μπορεί να επιτευχθεί με τη χρήση **protocols** που διαχωρίζουν τις ευθύνες.
5. **Dependency Inversion Principle (DIP) - Αρχή της Αντιστροφής Εξαρτήσεων:**
   Η αρχή αυτή δηλώνει ότι οι κλάσεις πρέπει να εξαρτώνται από αφηρημένους τύπους (**protocols**) και όχι από συγκεκριμένες υλοποιήσεις. Στη Swift, αυτό γίνεται με τη χρήση πρωτοκόλλων που επιτρέπουν την ανεξαρτησία των κλάσεων από συγκεκριμένες υλοποιήσεις.

Οι αρχές S.O.L.I.D. βελτιώνουν την ποιότητα του κώδικα, προσφέροντας καλύτερη επεκτασιμότητα, συντηρησιμότητα και επαναχρησιμοποίηση. Στην Swift, τα πρωτόκολλα, οι κλάσεις και η σωστή δομή κώδικα βοηθούν στη σωστή εφαρμογή αυτών των αρχών. Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.1 απεικονίζεται υλοποίηση της δομής το Clean Architecture στην εφαρμογή SnowHub.



Σχήμα: 3.1 S.O.L.I.D principles

3.2.1.2 Clean Architecture

Η Clean Architecture αρχιτεκτονική οργανώνει την εφαρμογή σε τρία βασικά layers: το Data Layer, το Domain Layer και το Presentation Layer[19]. Αυτή η δομή εξασφαλίζει ότι η λογική της εφαρμογής είναι ανεξάρτητη από τις λεπτομέρειες της υλοποίησης, όπως το UI, τα frameworks, τις βάσεις δεδομένων και άλλες εξαρτήσεις.

App Layer

Το App Layer είναι υπεύθυνο για τη διαχείριση της εκκίνησης της εφαρμογής και τη ρύθμιση των βασικών στοιχείων του περιβάλλοντος χρήστη. Στην εφαρμογή "SnowHub", το App Layer περιλαμβάνει:

- App file: Το σημείο εισόδου της εφαρμογής @main WindowGroup, όπου ορίζεται ο αρχικός View και ο custom NavigationView που διαχειρίζεται τις ροές της εφαρμογής. Αυτό το επίπεδο επιτρέπει την κεντρική διαχείριση της δομής της εφαρμογής, καθιστώντας την ευκολότερη στην παρακολούθηση και τροποποίηση.
- Custom NavigationView: Ένας custom NavigationView ο οποίος διαχειρίζεται τη παρουσίαση των διαφορετικών οθονών της εφαρμογής. Αυτός ο NavigationView αρχικοποιείται αμέσως στο @main window group του SwiftUI, εξασφαλίζοντας ότι η εμπειρία του χρήστη είναι ομαλή και συνεκτική από την πρώτη στιγμή που ανοίγει την εφαρμογή.

Core Layer

Το Core Layer περιλαμβάνει τα βασικά στοιχεία που είναι κοινά και επαναχρησιμοποιήσιμα σε ολόκληρη την εφαρμογή. Αυτά τα στοιχεία περιλαμβάνουν:

- Extensions: Επεκτάσεις που προσθέτουν επιπλέον λειτουργικότητα στις υπάρχουσες κλάσεις και τύπους του Swift. Αυτές οι επεκτάσεις διευκολύνουν την επαναχρησιμοποίηση του κώδικα και την προσθήκη χαρακτηριστικών χωρίς να αλλοιώνεται η βασική δομή των κλάσεων.
- Managers: Κλάσεις που διαχειρίζονται συγκεκριμένες λειτουργίες, όπως το API management, το caching, ή άλλες υποστηρικτικές λειτουργίες. Οι managers στο Core Layer είναι υπεύθυνοι για τη διαχείριση λειτουργιών που μπορούν να χρησιμοποιηθούν σε διαφορετικά μέρη της εφαρμογής.
- Classes: Κοινές κλάσεις που χρησιμοποιούνται σε όλη την εφαρμογή. Αυτές οι κλάσεις είναι σχεδιασμένες να είναι γενικής χρήσης και να μην εξαρτώνται από το συγκεκριμένο context στο οποίο χρησιμοποιούνται.
- Reusable Views: Προσαρμοσμένα SwiftUI Views που μπορούν να επαναχρησιμοποιηθούν σε διάφορες οθόνες της εφαρμογής. Αυτά τα views είναι σχεδιασμένα με στόχο την επαναχρησιμοποίηση, μειώνοντας τον κώδικα που χρειάζεται να γραφτεί και εξασφαλίζοντας συνέπεια στην εμφάνιση της εφαρμογής.
- Enums και Helpers: Συλλογές από enums και βοηθητικές κλάσεις ή συναρτήσεις που προσφέρουν σταθερές τιμές, επιλογές, και κοινές λειτουργίες. Αυτά τα στοιχεία

συμβάλλουν στην απλοποίηση της κωδικοποίησης και την αποφυγή επαναλαμβανόμενου κώδικα.

Resources Layer

Το Resources Layer περιλαμβάνει όλα τα αρχεία και τις ρυθμίσεις που είναι απαραίτητα για τη λειτουργία και την εμφάνιση της εφαρμογής. Αυτά τα στοιχεία είναι:

- Fonts: Τα προσαρμοσμένα γραμματοσειρές που χρησιμοποιούνται στην εφαρμογή για τη δημιουργία μοναδικής και συνεπούς οπτικής ταυτότητας.
- Localizable Strings: Τα αρχεία που περιέχουν τις μεταφράσεις και τα κείμενα που χρησιμοποιούνται στην εφαρμογή, επιτρέποντας την υποστήριξη πολλαπλών γλωσσών.
- Google.plist: Το αρχείο ρυθμίσεων για τις υπηρεσίες της Google, όπως το Firebase, που χρησιμοποιείται για την ενσωμάτωση με εργαλεία ανάλυσης και παρακολούθησης σφαλμάτων.
- Info.plist: Το αρχείο ρυθμίσεων της εφαρμογής που περιλαμβάνει βασικές πληροφορίες για την εφαρμογή, όπως δικαιώματα πρόσβασης, ρυθμίσεις αρχικοποίησης και άλλες παραμέτρους.
- Config.xcconfig files: Αρχεία ρύθμισης παραμέτρων που επιτρέπουν τη διαχείριση διαφορετικών ρυθμίσεων για τα περιβάλλοντα ανάπτυξης, όπως ανάπτυξη, δοκιμή και παραγωγή.
- Assets: Τα αρχεία πολυμέσων, όπως εικόνες και εικονίδια, που χρησιμοποιούνται στην εφαρμογή για τη δημιουργία μιας συνεπούς και ελκυστικής οπτικής ταυτότητας.

Data Layer

Το Data Layer είναι υπεύθυνο για την παροχή των δεδομένων στην εφαρμογή και αποτελείται από τα ακόλουθα στοιχεία:

- Entities: Οι οντότητες που καθορίζουν τη βασική δομή των δεδομένων της εφαρμογής. Είναι αντικείμενα που αντιπροσωπεύουν τα κύρια δεδομένα του συστήματος και είναι ανεξάρτητα από άλλες βιβλιοθήκες ή frameworks.
- Repositories Implementation: Τα repositories είναι υπεύθυνα για την υλοποίηση της πρόσβασης στα δεδομένα από διάφορες πηγές, όπως τοπικές βάσεις δεδομένων ή απομακρυσμένα APIs. Αυτά τα repositories υλοποιούν τα πρωτόκολλα που ορίζονται στο Domain Layer, επιτρέποντας την εφαρμογή να παραμείνει ανεξάρτητη από συγκεκριμένες λεπτομέρειες υλοποίησης.
- **Data Sources**: Τα data sources αντιπροσωπεύουν τις πηγές δεδομένων της εφαρμογής, είτε πρόκειται για τοπική αποθήκευση είτε για απομακρυσμένα APIs. Το Data Layer είναι υπεύθυνο για τη διαχείριση αυτών των πηγών και την παροχή των δεδομένων μέσω των repositories.

Domain Layer

Το Domain Layer είναι η καρδιά της εφαρμογής και περιλαμβάνει την επιχειρησιακή λογική. Αποτελείται από:

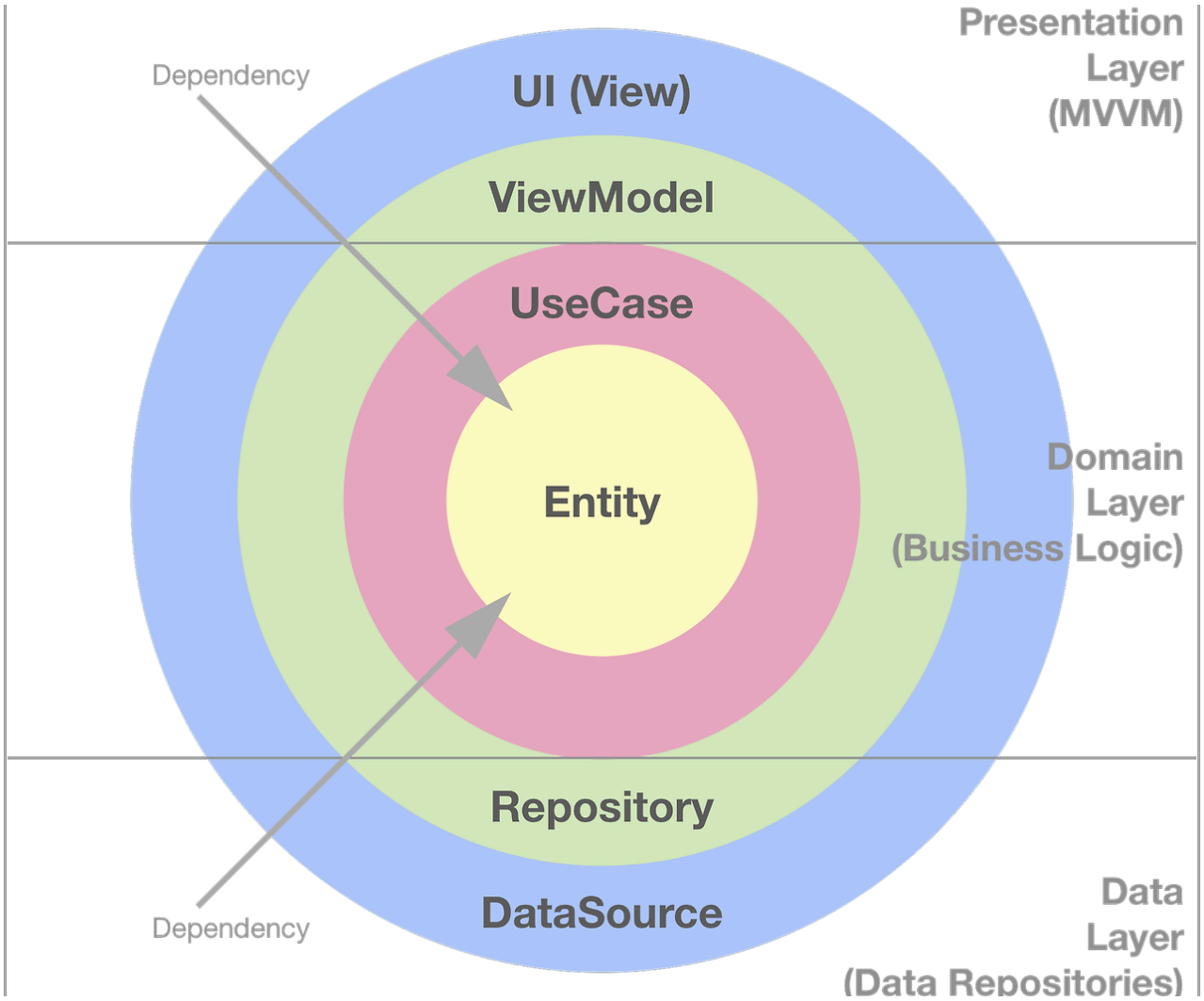
- Repository Protocols: Αυτά είναι τα πρωτόκολλα (interfaces) που καθορίζουν τις λειτουργίες που πρέπει να υποστηρίζουν τα repositories. Τα πρωτόκολλα αυτά επιτρέπουν τη δημιουργία ενός αφαιρετικού στρώματος που διαχωρίζει την επιχειρησιακή λογική από τις λεπτομέρειες υλοποίησης.
- Use Cases: Οι περιπτώσεις χρήσης αντιπροσωπεύουν τις επιχειρησιακές ροές της εφαρμογής. Κάθε use case περιγράφει μια συγκεκριμένη λειτουργικότητα που πρέπει να επιτευχθεί, όπως η ανάκτηση δεδομένων από μια βάση ή η αποστολή δεδομένων σε έναν server. Τα use cases ενσωματώνουν τη λογική της εφαρμογής και χρησιμοποιούν τα repository protocols για να αλληλεπιδράσουν με το Data Layer.
- Domain Entities: Οι οντότητες του Domain Layer αντιπροσωπεύουν τα δεδομένα όπως χρειάζονται στην επιχειρησιακή λογική. Αυτές οι οντότητες είναι καθαρές και ανεξάρτητες από συγκεκριμένες τεχνολογίες και frameworks.

Presentation Layer

Το Presentation Layer είναι υπεύθυνο για την αλληλεπίδραση με τον χρήστη και την παρουσίαση των δεδομένων. Στην εφαρμογή “SnowHub”, το Presentation Layer αποτελείται από τα παρακάτω στοιχεία:

- Views: Τα Views είναι υπεύθυνα για την εμφάνιση των δεδομένων και τη διαχείριση της αλληλεπίδρασης του χρήστη. Στο πλαίσιο της εφαρμογής, χρησιμοποιήθηκε το SwiftUI για την ανάπτυξη των Views, επιτρέποντας τη δημιουργία δηλωτικών και αντιδραστικών user interfaces.
- ViewModels: Τα ViewModels είναι υπεύθυνα για τη διαχείριση των δεδομένων που παρουσιάζονται στα Views και την επεξεργασία των ενεργειών του χρήστη. Τα ViewModels επικοινωνούν με τα use cases στο Domain Layer για να αντλήσουν ή να μεταφέρουν δεδομένα, και στη συνέχεια ενημερώνουν τα Views. Αυτός ο διαχωρισμός εξασφαλίζει ότι τα Views παραμένουν απλά και επικεντρωμένα στην παρουσίαση.

Η ενσωμάτωση του App Layer, του Core Layer και του Resources Layer στην Clean Architecture της εφαρμογής “SnowHub” προσφέρει επιπλέον δομή και οργάνωση, διευκολύνοντας τη διαχείριση της εφαρμογής και την επαναχρησιμοποίηση του κώδικα. Αυτή η προσέγγιση διασφαλίζει ότι η εφαρμογή είναι επεκτάσιμη, εύκολη στη συντήρηση και καλά οργανωμένη, προσφέροντας μια σταθερή βάση για μελλοντική ανάπτυξη και βελτίωση. Η δομή της αρχιτεκτονικής φαίνεται στο Σχήμα 3..

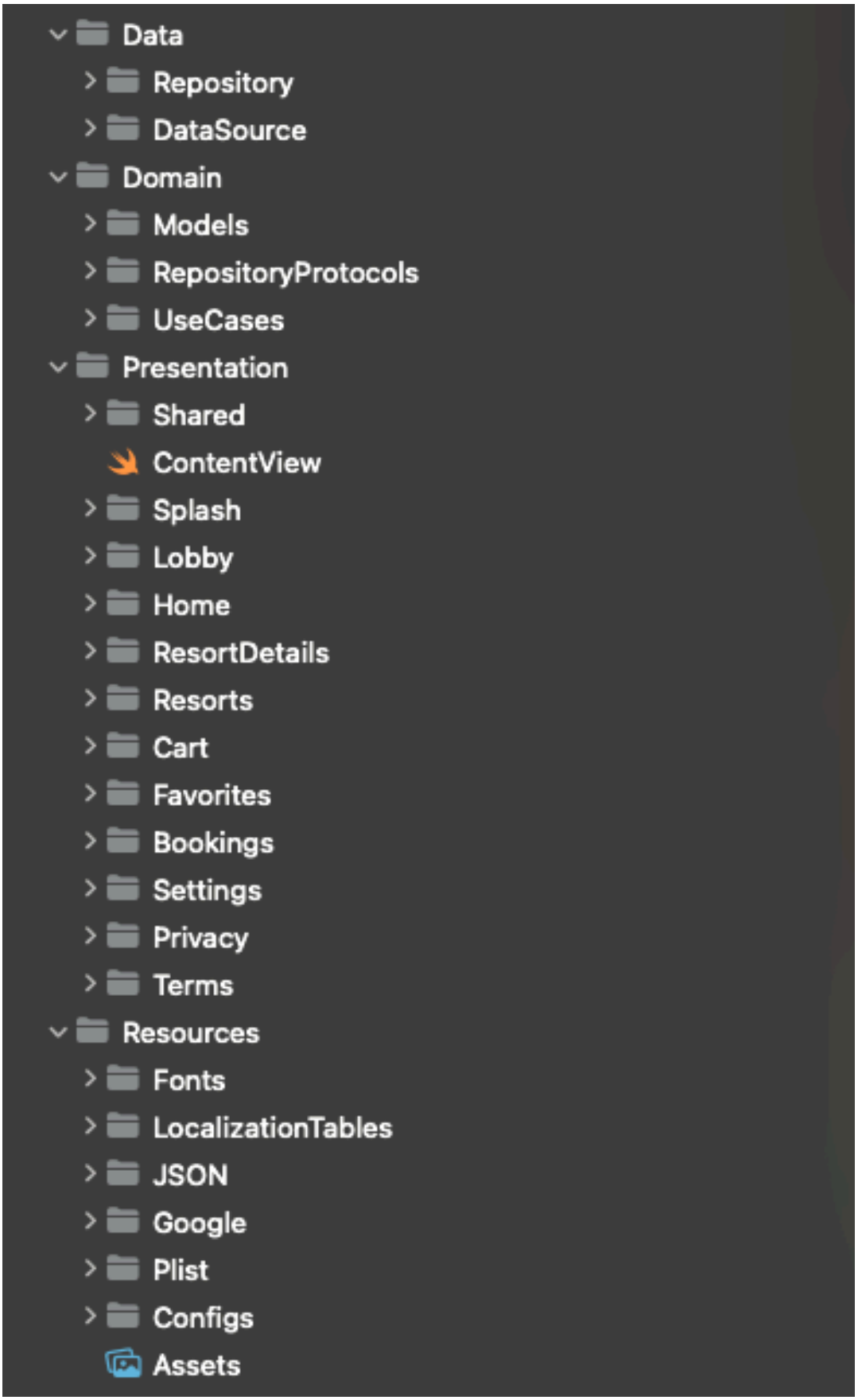


Σχήμα: 3.2 Clean Architecture

3.2.2 Clean Architecture στο SnowHub

Στο υποκεφάλαιο αυτο αναλύεται η συγκεκριμένη custom υλοποίηση του Clean Architecture στο SnowHub App. Η υλοποίηση αυτά είναι βασισμένη στο γενικό μοντέλο του Clean Architecture, λαμβάνοντας υπόψην τις απαιτήσεις της εφαρμογής. Η κάθε επιμέρους οθόνη ακολουθεί αυτό το μοτίβο και αποτελείται από συγκεκριμένα layers που αλληλεπιδρούν μεταξύ τους. Τα layers που χρησιμοποιήθηκαν φαίνονται στο Σχήμα 3.2.

Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.3 απεικονίζεται υλοποίηση της δομής το Clean Architecture στην εφαρμογή SnowHub.



Σχήμα: 3.3 Clean Architecture στο SnowHub

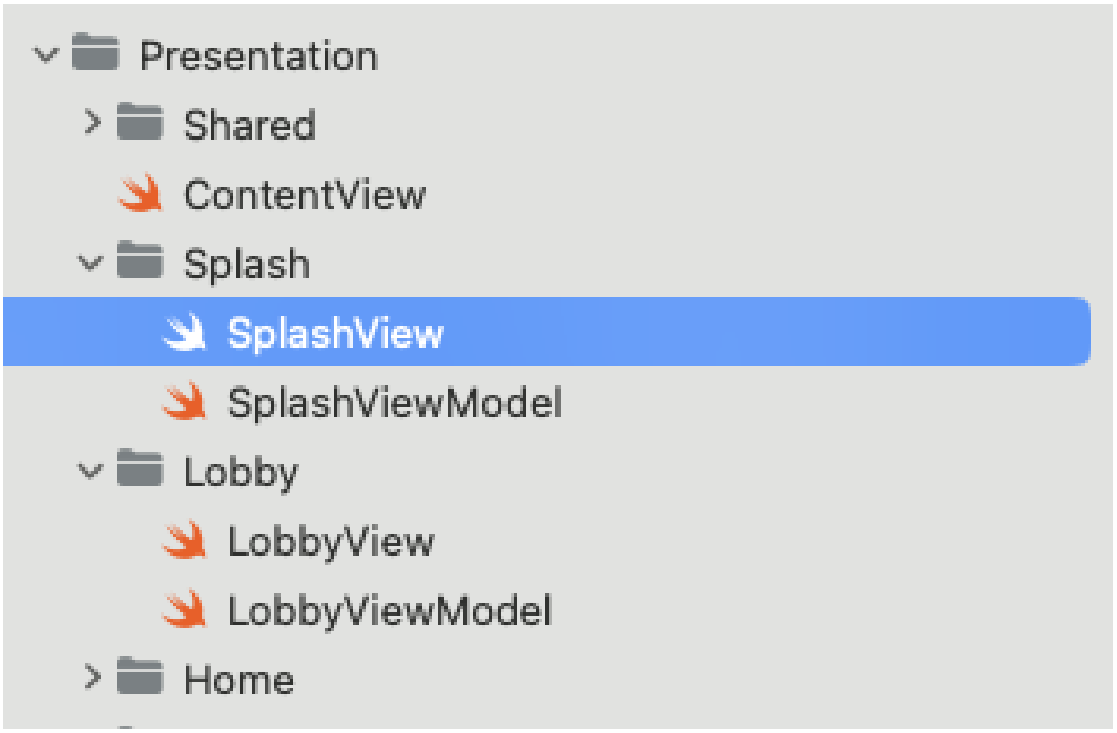
3.2.2.1 Presentation Layer

Στο Presentation Layer της εφαρμογής "SnowHub", το οποίο βασίζεται στο SwiftUI και την Clean Architecture, περιλαμβάνονται όλα τα αρχεία και ο κώδικας που σχετίζονται με τη διάταξη της εφαρμογής, την παρουσίαση των δεδομένων και την αλληλεπίδραση με τον χρήστη. Σε αντίθεση με τις παραδοσιακές iOS εφαρμογές που χρησιμοποιούν ViewControllers και Storyboards, στο SwiftUI η διαχείριση της διεπαφής χρήστη γίνεται μέσω δηλωτικών Views και ViewModels.

Συγκεκριμένα, στο Presentation Layer συναντάμε δύο βασικούς τύπους αρχείων:

- **Views**: Αυτά τα αρχεία περιέχουν τον κώδικα που καθορίζει τη διάταξη και την εμφάνιση της διεπαφής χρήστη. Τα Views στο SwiftUI είναι δηλωτικά, επιτρέποντας την εύκολη και κατανοητή περιγραφή της εμφάνισης της εφαρμογής. Κάθε View είναι υπεύθυνο για την παρουσίαση συγκεκριμένων δεδομένων και την απόκριση στις αλληλεπιδράσεις του χρήστη, όπως πατήματα κουμπιών ή πλοήγηση σε άλλες οθόνες.
- **ViewModels**: Τα ViewModels λειτουργούν ως γέφυρα μεταξύ του Presentation Layer και του Domain Layer. Είναι υπεύθυνα για τη μετατροπή των δεδομένων από το Domain Layer σε μορφή που μπορεί να παρουσιαστεί από τα Views. Επιπλέον, διαχειρίζονται τις ενέργειες του χρήστη και επικοινωνούν με τα Use Cases του Domain Layer για να εκτελέσουν τις απαραίτητες λειτουργίες.

Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.4 απεικονίζεται η δομή το Presentation Layer στην εφαρμογή SnowHub με Views και ViewModels.



Σχήμα 3.5 Παράδειγμα δομής Presentation Layer στην εφαρμογή SnowHub

Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.5 παρατηρούμε το HomeView. Η HomeView της εφαρμογής “SnowHub” είναι η πρώτη οθόνη που συναντούν οι χρήστες και έχει σχεδιαστεί για να προσφέρει μια εντυπωσιακή και λειτουργική εμπειρία χρησιμοποιώντας SwiftUI elements. Στην κορυφή της οθόνης, περιλαμβάνεται ένα image slider που παρουσιάζει εντυπωσιακές φωτογραφίες από διάφορα χιονοδρομικά κέντρα, δημιουργώντας μια ελκυστική πρώτη εντύπωση. Ο image slider δημιουργείται με την χρηση του element ScrollView ο οποίος περιέχει εμφωλευμένα ενα VStack (vertical stack) με εικόνες που δημιουργούνται με την χρηση του element Image.

Ακριβώς κάτω από το image slider, υπάρχει ένα άλλο slider (card slider) που εμφανίζει τα πιο πρόσφατα χιονοδρομικά κέντρα, προσφέροντας στους χρήστες μια γρήγορη επισκόπηση των διαθέσιμων επιλογών για να προγραμματίσουν την επόμενη εξόρμησή τους. Όλα τα παραπάνω στοιχεία συνδυάζονται αρμονικά στο SwiftUI, προσφέροντας μια ομαλή και εντυπωσιακή εμπειρία χρήστη.

```swift
import SwiftUI
import SwiftData

struct HomeView: View {

    // MARK: - PROPERTIES 🌐 PUBLIC
    @EnvironmentObject var coordinator: AppCoordinator
    @EnvironmentObject var authViewModel: AuthViewModel
    @StateObject var viewModel = HomeViewModel()
    @State var showStatusBar = true
    @State var contentHasScrolled = false
    @State var resorts: [Resort] = []

    // MARK: - PROPERTIES 🔰 PRIVATE
    @State private var animateGradient: Bool = false


    // MARK: - VIEW
    var body: some View {
        ZStack {
            LinearGradient(gradient: Gradient(colors: [.white, .white]), startPoint: .topLeading, endPoint: .bottomTrailing)
                .edgesIgnoringSafeArea(.all)
                .hueRotation(.degrees(animateGradient ? 10 : 0))
                .onAppear {
                    withAnimation(Animation.linear(duration: 10).repeatForever(autoreverses: true)) {
                        animateGradient.toggle()
                    }
                }

            ScrollView {
                self.scrollingDetection()
                Group { ••• }
            }
        }
        .padding(.bottom)
        .onAppear {
            self.setUp()
        }
        .refreshable { ••• }
        .onChange( ••• )
        .overlay(NavigationBar(title: ViewStrings.navSettings_title.localized, textField: $textField, contentHasScrolled: $contentHasScrolled), alignment: .top)
        .statusBar(hidden: !showStatusBar)
    }

    // MARK: - VIEW METHODS
    @ViewBuilder
    func makePopularResortsSection() -> some View {
        ScrollView(.vertical, showsIndicators: false) {
            LazyVStack(spacing: 12) {
                ForEach($viewModel.resorts, id: \.self) { resort in
                    ResortCardView(resort: resort)
                        .onTapGesture {
                            coordinator.push(.resort(resort.wrappedValue, resorts))
                        }
                }
            }
            .padding()
        }
    }
```

Σχήμα 3.5 HomeView

Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.6 το HomeViewModel, Το HomeViewModel της εφαρμογής “SnowHub” είναι υπεύθυνο για την κεντρική λογική επεξεργασίας δεδομένων που εμφανίζονται στο HomeView. Αυτό περιλαμβάνει τη συλλογή δεδομένων από διάφορες πηγές, τη μετατροπή τους σε Domain Models που μπορούν να χρησιμοποιηθούν από το

View Layer για την παρουσίασή τους στο UI, καθώς και το φιλτράρισμα και την παραμετροποίηση τους σύμφωνα με τις ανάγκες της εφαρμογής.

Το HomeViewModel διαχειρίζεται όλα τα στάδια της επεξεργασίας δεδομένων, διασφαλίζοντας ότι το View ενημερώνεται κατάλληλα όταν ολοκληρωθούν αυτές οι διαδικασίες. Αυτό επιτυγχάνεται μέσω της χρήσης Observable μεταβλητών, συγκεκριμένα με την @Published ιδιότητα του SwiftUI. Οι μεταβλητές αυτές παρακολουθούνται από το View, επιτρέποντάς του να αντιδρά άμεσα σε αλλαγές, όπως όταν νέα δεδομένα είναι διαθέσιμα ή όταν ολοκληρωθεί μια λειτουργία.

Κάθε ViewModel στην εφαρμογή, συμπεριλαμβανομένου του HomeViewModel, υλοποιεί το ViewStateEnum, το οποίο διαχειρίζεται την κατάσταση των δεδομένων και συνεπώς την κατάσταση του View. Αυτό είναι ιδιαίτερα χρήσιμο για την εμφάνιση ενός loader όταν εκτελείται μια λειτουργία που απαιτεί αναμονή από τον χρήστη, όπως η λήψη δεδομένων από το δίκτυο. Για παράδειγμα, όταν το HomeViewModel ξεκινά να ανακτά δεδομένα για τα χιονοδρομικά κέντρα, η κατάσταση του ViewStateEnum αλλάζει σε loading, ενεργοποιώντας την εμφάνιση ενός loader στο HomeView μέχρι να ολοκληρωθεί η διαδικασία και να είναι διαθέσιμα τα δεδομένα για προβολή.

Αυτό το μοτίβο επιτρέπει τη δημιουργία μιας εφαρμογής που είναι ταυτόχρονα ευέλικτη και αποκριτική, διασφαλίζοντας ότι οι χρήστες λαμβάνουν πάντα τις πιο ενημερωμένες και σχετικές πληροφορίες, χωρίς να αντιμετωπίζουν καθυστερήσεις ή άλλα προβλήματα.

```swift
import Foundation

@MainActor
final class HomeViewModel: ObservableObject {

    // MARK: - INJECTED 🔗 VALUES
    @Injected(\.getResortUseCase) var getResortUseCase: GetResortUseCaseProtocol
    @Injected(\.getResortsUseCase) var getResortsUseCase: GetResortsUseCaseProtocol
    @Injected(\.getImagesUseCase) var getImagesUseCase: GetImagesUseCaseProtocol

    // MARK: - PROPERTIES 🌐 PUBLIC
    @Published var resorts: [Resort] = []
    @Published var filteredResorts: [Resort] = []
    @Published var carouselImages: [String] = []

    // MARK: - PROPERTIES 🔰 PRIVATE
    @Published private(set) var viewStateEnum: ViewStateEnum<[Resort]> = .idle

    // MARK: - METHODS 🌐 PUBLIC
    func setUp(favorites: [UserFavorite], isLoggedIn: Bool) async {
        await getResorts(favorites: favorites, isLoggedIn: isLoggedIn)
        await getImages()
    }

    // MARK: - METHODS 🔰 PRIVATE
    private func getResorts(favorites: [UserFavorite], isLoggedIn: Bool) async {
        self.viewStateEnum = .loading
        let result = await getResortsUseCase.execute(favorites: favorites, isLoggedIn: isLoggedIn)
        switch result {
        case .success(let resorts):
            self.viewStateEnum = .loaded(resorts)
            self.resorts = resorts
        case .failure(let getResortsUseCaseError):
            guard case let .unableToRetrieveData(.some(error)) = getResortsUseCaseError else { return }
            self.viewStateEnum = .failed(error)
        }
    }

    private func getImages() async {
        self.viewStateEnum = .loading
        let result = await getImagesUseCase.execute()
        switch result {
        case .success(let images):
            self.carouselImages = images
        case .failure(let getImagesUseCaseError):
            guard case let .unableToRetrieveData(.some(error)) = getImagesUseCaseError else { return }
        }
    }
}
```

Σχήμα 3.6 HomeViewModel

Με σκοπό την καλύτερη οργάνωση του κώδικα, οι παραπάνω λειτουργίες στο ViewModel χωρίζονται με τη μορφή των UseCases. Ένα UseCase είναι υπεύθυνο για να εκτελέσει όλες τις απαραίτητες ενέργειες για την επίτευξη ενός συγκεκριμένου σκοπού. Έτσι ένα ViewModel αποτελείται από ένα ή περισσότερα UseCase που στο σύνολο τους διεκπεραιώνουν όλες τις λειτουργίες για τις οποίες το ViewModel είναι υπεύθυνο.

Σε μια τυπική ροή μεταξύ των UseCases, του ViewModel και του View, το View καλεί μια μέθοδο setUp του ViewModel, η οποία είναι υπεύθυνη για την εκτέλεση ενός UseCase. Το αποτέλεσμα του UseCase αυτού μετά την εκτέλεση του επιστρέφονται στο ViewModel και από εκεί με τη χρήση μιας Observable μεταβλητής στο View, το οποίο εκτελεί τις απαραίτητες ενέργειες ενημέρωσης του UI.

Το παρακάτω κομμάτια κώδικα αποτελούν παραδείγματα της ροής αυτής στην οθόνη Home. Αρχικά η HomeView κάθε φορά που εμφανίζεται εκτελεί την μέθοδο setUp η οποία με την σειρά της καλεί το viewModel να καλέσει την δικη του setUp μεθοδο.

3.2.2.2 Domain Layer

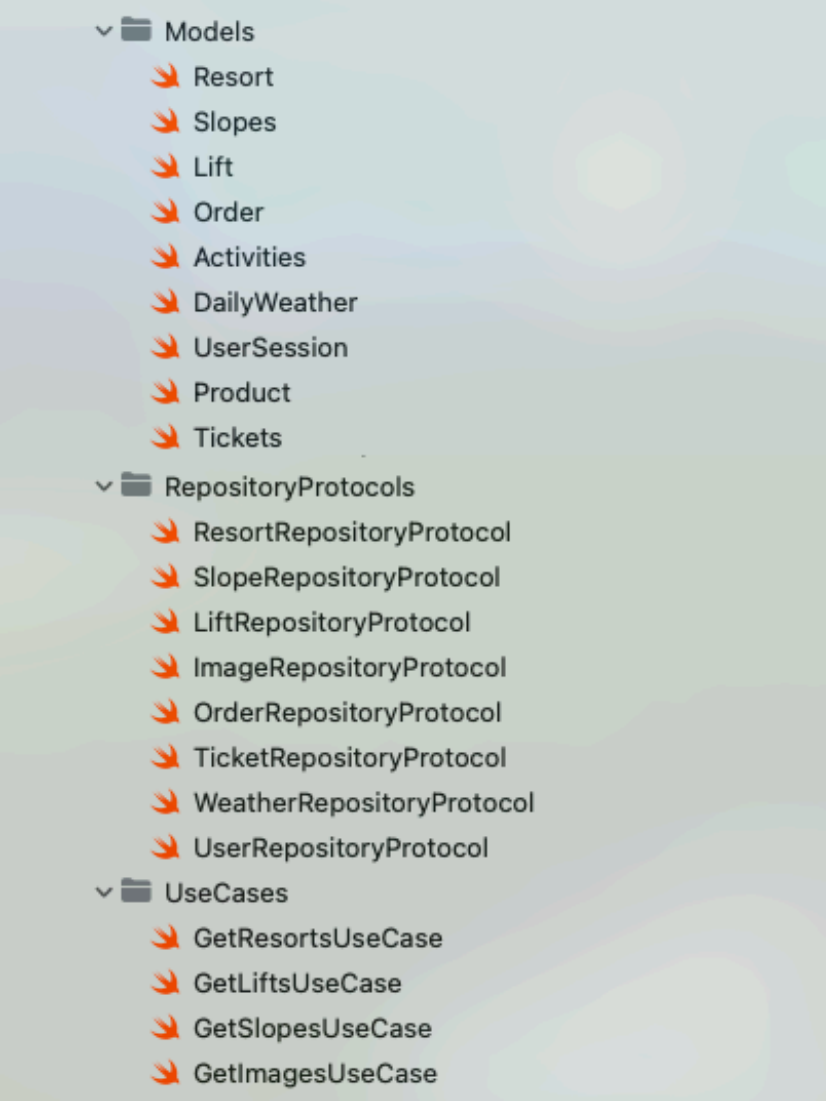
Στο Domain Layer της εφαρμογής "SnowHub", το οποίο είναι θεμελιώδες στη Clean Architecture, περιλαμβάνονται όλα τα στοιχεία που σχετίζονται με την επιχειρηματική λογική **(business logic)** και τους κανόνες της εφαρμογής. Στο επίπεδο αυτό δεν υπάρχει γνώση των λεπτομερειών υλοποίησης των υπόλοιπων επιπέδων, γεγονός που διασφαλίζει την ανεξαρτησία και την ευκολία στη συντήρηση και επαναχρησιμοποίηση του κώδικα.

Τα βασικά στοιχεία του Domain Layer είναι:

- **Domain Models**: Αυτά είναι τα struct ή class μοντέλα που αντιπροσωπεύουν τις βασικές οντότητες της εφαρμογής, όπως τα δεδομένα χρήστη, οι δραστηριότητες στα χιονοδρομικά κέντρα, τα εισιτήρια και άλλα αντικείμενα που αποτελούν την ουσία του SnowHub. Τα Domain Models δεν εξαρτώνται από το πώς αποθηκεύονται ή προβάλλονται τα δεδομένα, εστιάζοντας αποκλειστικά στη δομή και τη λογική που διέπει τα δεδομένα.
- **Use Cases**: Τα Use Cases αντιπροσωπεύουν τις βασικές επιχειρησιακές λειτουργίες της εφαρμογής. Είναι υπεύθυνα για την εκτέλεση συγκεκριμένων ενεργειών που σχετίζονται με τα δεδομένα της εφαρμογής, όπως η παρακολούθηση των χιονοδρομικών δραστηριοτήτων ή η επεξεργασία των κρατήσεων για τα resorts. Τα Use Cases επικοινωνούν με το Repository μέσω πρωτοκολλων **(protocols)** για να συλλέξουν τα δεδομένα που χρειάζονται και εφαρμόζουν τους κανόνες λογικής για την εφαρμογή.
- **Repository Protocols**: Τα πρωτόκολλα Repository ορίζουν τις λειτουργίες **(functions)** για την αλληλεπίδραση με τα δεδομένα, χωρίς να εξαρτώνται από την υλοποίηση της αποθήκευσης (π.χ., τοπικά ή από το διαδίκτυο). Παρέχουν έναν τρόπο πρόσβασης στα δεδομένα που επιτρέπει την ευκολία στη δοκιμή και τη συντήρηση.

Το Domain Layer είναι πλήρως ανεξάρτητο από οποιαδήποτε πλατφόρμα ή τεχνολογία, διασφαλίζοντας την ευελιξία και τη δυνατότητα επέκτασης της εφαρμογής, καθώς νέες λειτουργίες μπορούν να προστεθούν εύκολα χωρίς να επηρεάζονται τα υπόλοιπα επίπεδα.

Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.6 απεικονίζεται η δομή το Domain Layer στην εφαρμογή SnowHub με UseCases, Domain Models και Repositories Protocols.



Σχήμα 3.6 Παράδειγμα δομής Domain Layer στην εφαρμογή SnowHub

Στο παράδειγμα που απεικονίζεται στο Σχήμα 3.7 παρουσιάζεται η λειτουργία του GetResortsUseCase. Το Domain Model που χρησιμοποιείται είναι το Resort. Ο σκοπός αυτου του μοντέλου είναι να αναπαριστά τις πληροφορίες ενός χιονοδρομικού κέντρου, όπως η τοποθεσία, οι διαθεσιμότητα και οι πλαγιές**(slopes)**, και να είναι εύκολα προσβάσιμο από διάφορα επίπεδα της εφαρμογής.

Ο GetResortsUseCase αποτελεί ένα από τα πιο σημαντικά Use Cases της εφαρμογής, καθώς είναι υπεύθυνος για την ανάκτηση της λίστας των χιονοδρομικών κέντρων (Resorts). Ο ρόλος του Use Case είναι να καθορίζει τις επιχειρησιακές λογικές που αφορούν την αλληλεπίδραση με τα δεδομένα, ενώ διασφαλίζει ότι οι κανόνες του τομέα (domain) τηρούνται. Το Use Case λειτουργεί ως γέφυρα μεταξύ των δεδομένων (που παρέχονται από τα Repositories) και της επιχειρηματικής λογικής. Κάθε Use Case περιέχει μια μέθοδο **execute()** που είναι υπεύθυνη για την έναρξη της επιχειρησιακης λογικής του κάθε Use Case. Με τη κλήση απο την μέθοδο **execute(),** απο το εκάστοτε ViewModel ξεκινά την διαδικασία της επιχειρησιακής λογικής και καλεί με την σειρά του την μέθοδο **getResorts()** η οποια είναι προσβάσιμη από το repository με την χρηση πρωτοκόλλου. Η μέθοδος **execute()** είναι υπεύθυνη

να επιστρέψει στο εκαστοτε ViewModel έναν πίνακα από μοντέλα τύπου Resort, είτε error που περιγράφεται ως αδυναμία παροχής δεδομένων το οποίο έχει την ανάλογη διαχείριση στο ViewModel. Επιπλέον έχει και μια ακόμα μέθοδο **execute()** η οποία λαμβάνει και παραμέτρους, όπως το το isLoggedIn που είναι μια μεταβλητή boolean (αληθείας), με σκοπό να επιστρέψει στο εκαστοτε ViewModel έναν πίνακα με μοντέλα UserFavorite.

```swift
import Foundation

protocol GetResortsUseCaseProtocol {

    @discardableResult
    func execute() async -> Result<[Resort], GetResortsUseCaseError>

    @discardableResult
    func execute(favorites: [UserFavorite], isLoggedIn: Bool) async -> Result<[Resort], GetResortsUseCaseError>
}

enum GetResortsUseCaseError: Error {
    case unableToRetrieveData(Error?)
}

struct GetResortsUseCase: GetResortsUseCaseProtocol {

    // MARK: - INJECTED 🔗 VALUES
    @Injected(\.resortRepository) var repository: ResortRepositoryProtocol

    // MARK: - METHODS 🌐 PUBLIC
    func execute() async -> Result<[Resort], GetResortsUseCaseError>  {
        do {
            let resorts = try await repository.getResorts()
            return .success(resorts)
        } catch {
            return .failure(.unableToRetrieveData(error))
        }
    }

    func execute(favorites: [UserFavorite], isLoggedIn: Bool) async -> Result<[Resort], GetResortsUseCaseError>  {
        do {
            let resorts = try await repository.getResorts()
            let result = matchTheFavorites(favorites: favorites, resorts: resorts, isLoggedIn: isLoggedIn)
            return .success(result)
        } catch {
            return .failure(.unableToRetrieveData(error))
        }
    }

    // MARK: - METHODS 🔰 PRIVATE

    private func matchTheFavorites(favorites: [UserFavorite], resorts: [Resort], isLoggedIn: Bool) -> [Resort] {•••}

}
```

Σχήμα 3.7 Λειτουργία του GetResortsUseCase

Στο παράδειγμα που απεικονίζεται στο Σχήμα 3.8 παρουσιάζεται η λειτουργία του **Resort** ως domain model(entity).

Το **Resort** είναι το βασικό μοντέλο δεδομένων που αντιπροσωπεύει τις πληροφορίες ενός χιονοδρομικού κέντρου. Το μοντέλο αυτό παίζει καθοριστικό ρόλο στην επιχειρηματική λογική της εφαρμογής, καθώς περιέχει όλα τα σχετικά δεδομένα για κάθε χιονοδρομικό κέντρο και χρησιμοποιείται σε διάφορα Use Cases και επίπεδα της αρχιτεκτονικής. Το **Resort** είναι σχεδιασμένο με γνώμονα την ανεξαρτησία και τη σαφήνεια. Λειτουργεί ως ένα απλό, αλλά ουσιαστικό αντικείμενο (entity) που διασφαλίζει ότι όλες οι πληροφορίες που σχετίζονται με τα χιονοδρομικά κέντρα μπορούν να διαχειρίζονται αποτελεσματικά και να διαμοιράζονται μεταξύ των επιπέδων της εφαρμογής.

Το **Resort** περιλαμβάνει διάφορες ιδιότητες που αντιπροσωπεύουν τα δεδομένα που σχετίζονται με ένα χιονοδρομικό κέντρο. Αυτές οι ιδιότητες μπορεί να περιλαμβάνουν:

- **Όνομα** (name): Το όνομα του χιονοδρομικού κέντρου.
- **Τοποθεσία** (location): Γεωγραφική τοποθεσία του κέντρου, που μπορεί να περιλαμβάνει συντεταγμένες (latitude, longitude).
- **Υποδομές** (facilities): Δεδομένα σχετικά με τις υποδομές του χιονοδρομικού κέντρου, όπως πίστες σκι, lifts, καταλύματα κ.ά.
- **Καιρικές Συνθήκες** (weather): Πληροφορίες για τις τρέχουσες καιρικές συνθήκες στο χιονοδρομικό.
- **Διαθεσιμότητα** (availability): Πληροφορίες για τη διαθεσιμότητα και την κατάσταση των εγκαταστάσεων.

Χρησιμοποιείται κυρίως από τα Use Cases και τα ViewModels, τα οποία χρειάζονται τις πληροφορίες που αντιπροσωπεύει το μοντέλο για να εκτελέσουν λειτουργίες, όπως η παρουσίαση των δεδομένων στον χρήστη ή η επεξεργασία κρατήσεων και εισιτηρίων. Αποτελεί την κύρια οντότητα που ανακτάται από το σύστημα και έχει σχεδιαστεί με στόχο την ανεξαρτησία και την επεκτασιμότητα. Χάρη στον αφαιρετικό του χαρακτήρα, δεν εξαρτάται από τον τρόπο αποθήκευσης ή ανάκτησης των δεδομένων. Αυτό επιτρέπει την εύκολη ενσωμάτωση νέων λειτουργιών ή τη βελτίωση υπαρχουσών χωρίς να χρειάζονται αλλαγές στην ίδια την οντότητα.

Το **Resort** είναι ένα καίριο κομμάτι του **Domain Layer** της "**SnowHub**". Λειτουργεί ως το κεντρικό σημείο αναφοράς για όλες τις πληροφορίες που αφορούν τα χιονοδρομικά κέντρα, ενώ η ανεξάρτητη φύση του διασφαλίζει τη συντήρηση και την επεκτασιμότητα της εφαρμογής στο μέλλον. Ως **Domain Model**, το **Resort** διευκολύνει την αλληλεπίδραση μεταξύ των διαφόρων επιπέδων της εφαρμογής και προσφέρει μια σταθερή βάση για την προσθήκη νέων λειτουργιών και τη βελτίωση της συνολικής εμπειρίας του χρήστη.

```swift
struct Resort: Identifiable, Hashable {

    // MARK: - VARIABLES 🌐 PUBLIC
    var id: Int
    var name: String?
    let overview: String?
    var location: String?
    let elevation: [Int]?
    let slopes: [Slopes]?
    var lifts: [Lift]?
    var snowDepth: String?
    var lastSnowfall: String?
    var status: SkiResortStatusEnum?
    let activities: [Activities]?
    var mainImage: String?
    let thumbnailImage: String?
    let gallery: [String]?
    let site: String?
    var isLiked: Bool
    let coordinate: CLLocationCoordinate2D
    var ticket: Ticket?

    // Computed property to check if the resort is open
    var isOpen: Bool {
        return status == .open
    }

    // MARK: - INITIALIZATION
    init( ••• ) { ••• }

    init(resort: ResortEntity, coordinate: CLLocationCoordinate2D) { ••• }

    // MARK: - METHODS 🌐 PUBLIC
    static func getPreview() -> Resort {
        Resort(id: UUID().hashValue, coordinate: CLLocationCoordinate2D())
    }
}
```

Σχήμα 3.8 Λειτουργία του Resort Domain Model

Στο παράδειγμα που απεικονίζεται στο Σχήμα 3.9 παρουσιάζεται η λειτουργία του **ResortRepositoryProtocol**.

Για να διατηρείται η διαχωρισμένη αρχιτεκτονική, το **ResortRepositoryProtocol** καθορίζει το σύστημα με το οποίο το **GetResortsUseCase** αλληλεπιδρά με τα δεδομένα. Αυτό το πρωτόκολλο ορίζει τις βασικές λειτουργίες που πρέπει να υποστηρίζουν οι συγκεκριμένες υλοποιήσεις των

Repositories, όπως η ανάκτηση της λίστας των Resorts από απομακρυσμένες ή τοπικές πηγές δεδομένων.

Το μεγάλο πλεονέκτημα της χρήσης των Repository Protocols είναι η ανεξαρτησία που προσφέρουν στο **Domain Layer** από τις συγκεκριμένες λεπτομέρειες της υλοποίησης των δεδομένων. Έτσι, το **GetResortsUseCase** μπορεί να λειτουργεί ομαλά είτε τα δεδομένα προέρχονται από το διαδίκτυο μέσω API είτε από μια τοπική βάση δεδομένων. Συγκεκριμένα, η λειτουργία του **GetResortsUseCase** ακολουθεί τα εξής βήματα:

1. Καλεί το **ResortRepositoryProtocol** για να ζητήσει τα δεδομένα των χιονοδρομικών κέντρων όπως για παράδειγμα με την χρήση της μεθόδου **getResorts()**.
2. Το πρωτόκολλο αυτό, με βάση τις υλοποιήσεις του στο Data Layer, ανακτά τα δεδομένα και τα επιστρέφει στο UseCase.
3. Το **GetResortsUseCase** εφαρμόζει την ανάλογη επιχειρησιακή λογική, όπως φιλτράρισμα των δεδομένων ή εφαρμογή κανόνων, και επιστρέφει τα τελικά δεδομένα στο ViewModel για παρουσίαση στον χρήστη.

```
import Foundation

protocol ResortRepositoryProtocol {
    func getResort(id: Int) async throws -> Resort
    func getResorts() async throws -> [Resort]
    func getHottestsResorts() async throws -> [Resort]
}
```

Σχήμα 3.9 Λειτουργία του Resorts Repository Protocol

3.2.2.3 Data Layer

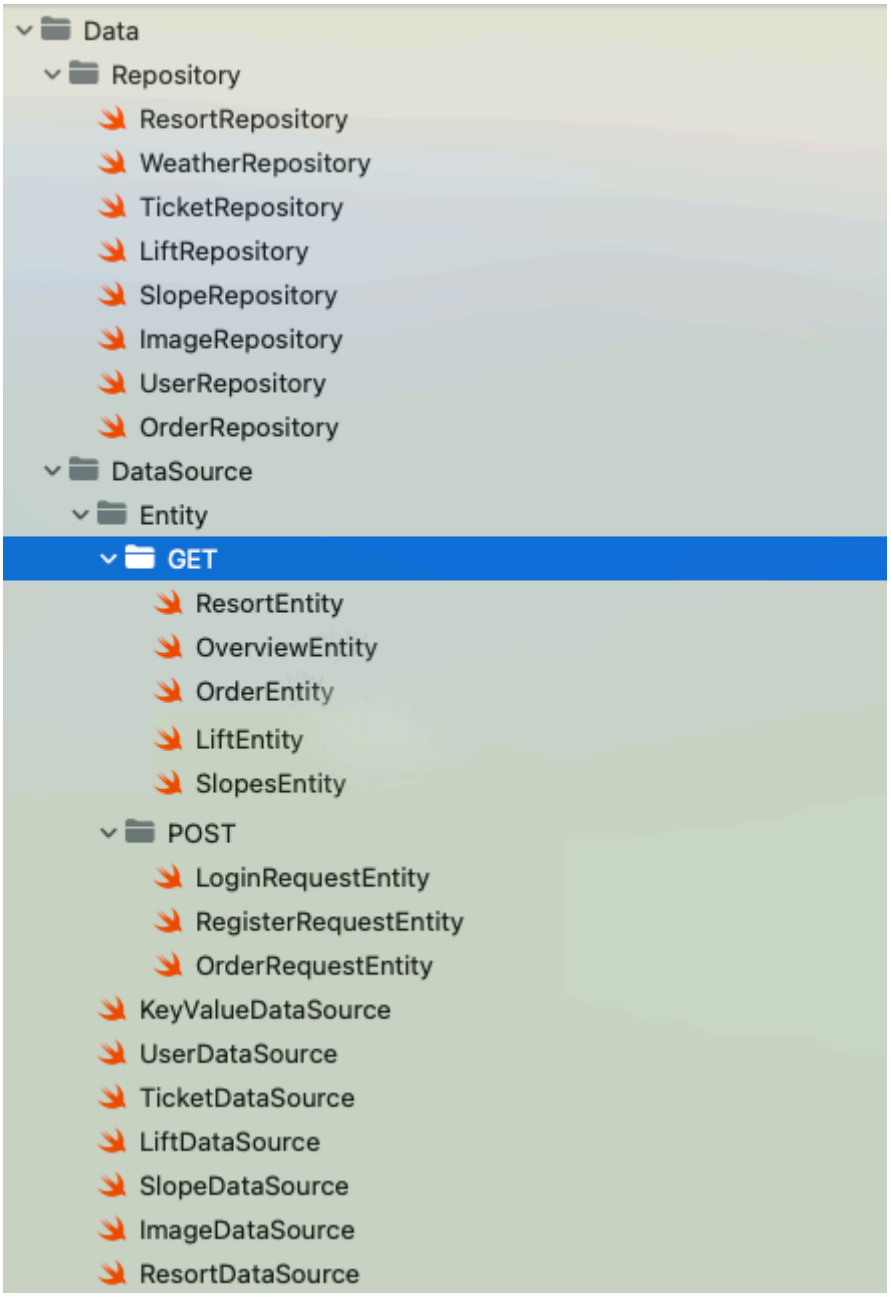
Το Data Layer της εφαρμογής “SnowHub” είναι υπεύθυνο για την αλληλεπίδραση με τις εξωτερικές πηγές δεδομένων και την αποθήκευση των πληροφοριών. Αυτή η στρώση της αρχιτεκτονικής χειρίζεται την πρόσβαση στα δεδομένα μέσω των repositories και των data sources, με σκοπό να διαχωρίσει την εφαρμογή από τις λεπτομέρειες της διαχείρισης των δεδομένων. Το Data Layer αποτελείται από τρεις κύριες συνιστώσες:

- **Data Entities**: Τα Data Entities είναι τα μοντέλα δεδομένων που χρησιμοποιούνται για την αποθήκευση και τη μεταφορά δεδομένων μεταξύ του Data Layer και των εξωτερικών πηγών (π.χ., βάσεις δεδομένων, APIs). Αυτά τα μοντέλα μπορεί να είναι παρόμοια με τα Domain Models, αλλά συνήθως περιέχουν πρόσθετες πληροφορίες που σχετίζονται με την αποθήκευση ή την αποστολή δεδομένων (π.χ., κωδικοποίηση JSON ή αποθήκευση στη βάση δεδομένων). Η βασική διαφορά με τα Domain Models είναι ότι τα Data Entities επικεντρώνονται στην αποθήκευση και μεταφορά δεδομένων, ενώ τα Domain Models στην επιχειρηματική λογική.

- **Repositories (Implementations)**: Οι υλοποιήσεις των repositories είναι υπεύθυνες για την πραγματική πρόσβαση και διαχείριση των δεδομένων. Αυτές οι κλάσεις υλοποιούν τα πρωτόκολλα των repositories που βρίσκονται στο Domain Layer, συνδέοντας τις πηγές δεδομένων (π.χ., διαδικτυακά APIs ή τοπικές βάσεις δεδομένων) με το επιχειρηματικό λογισμικό της εφαρμογής. Μέσω αυτής της υλοποίησης, το Domain Layer παραμένει ανεξάρτητο από τις λεπτομέρειες της αποθήκευσης των δεδομένων.
- **Data Sources**: Τα Data Sources είναι υπεύθυνα για την πραγματική σύνδεση με τις εξωτερικές πηγές δεδομένων, όπως τα APIs ή τις βάσεις δεδομένων. Ένα data source μπορεί να είναι ένα API client που επικοινωνεί με έναν απομακρυσμένο διακομιστή ή μια υπηρεσία αποθήκευσης για τη διαχείριση δεδομένων σε τοπικό επίπεδο. Αυτή η δομή επιτρέπει τη διαχωρισμένη πρόσβαση σε διαφορετικές πηγές δεδομένων και καθιστά ευκολότερη την αντικατάστασή τους ή την ενσωμάτωση νέων, όπως και την εύκολη μετάβαση σε άλλες τεχνολογίες αν χρειαστεί.

Το Data Layer στο SnowHub διαχειρίζεται τις εξωτερικές πηγές δεδομένων και εγγυάται την παροχή αξιόπιστων δεδομένων στο Domain Layer, διατηρώντας τη διάκριση μεταξύ του τρόπου διαχείρισης και αποθήκευσης των δεδομένων και της επιχειρηματικής λογικής της εφαρμογής.

Στο παρακάτω παράδειγμα που απεικονίζεται στο Σχήμα 3.10 απεικονίζεται η δομή το Data Layer στην εφαρμογή SnowHub με Data Sources, Data Entities και Repositories.



Σχήμα 3.10 Παράδειγμα δομής Data Layer στην εφαρμογή SnowHub

Στο παράδειγμα που απεικονίζεται στο Σχήμα 3.11 παρουσιάζεται η λειτουργία του **ResortRepository**.

Το **ResortRepository** υλοποιεί το πρωτόκολλο **ResortRepositoryProtocol** από το Domain Layer και διασφαλίζει την αλληλεπίδραση με τα δεδομένα, συνδέοντας το Domain με το Data Layer. Το **ResortRepository** είναι αυτό που λαμβάνει τα δεδομένα από το **ResortDataSource** και τα μετατρέπει σε μορφή που μπορεί να χρησιμοποιηθεί από το Domain Layer.

Η βασική λειτουργία του **ResortRepository** είναι να κρύβει την πολυπλοκότητα της πρόσβασης στα δεδομένα, εξασφαλίζοντας ότι το Domain Layer δεν χρειάζεται να γνωρίζει πώς ή από πού προέρχονται τα δεδομένα. Είτε αυτά τα δεδομένα προέρχονται από ένα απομακρυσμένο API είτε από

μια τοπική βάση δεδομένων, το **Repository** είναι υπεύθυνο για να επιστρέψει τα δεδομένα με τρόπο που να είναι συνεπής και αξιόπιστος.

Όταν στο Domain Layer καλείτε το **GetResortsUseCase**, αυτο θα απευθυνθεί στο **ResortRepository** μέσω του **ResortRepositoryProtocol** για να ανακτήσει τα δεδομένα. Το **ResortRepository** στη συνέχεια θα ζητήσει από το **ResortDataSource** να ανακτήσει τα δεδομένα από το backend, και αυτά θα μεταφερθούν ως **ResortEntity** πίσω στο Repository. Το Repository θα αναλάβει να μετατρέψει αυτά τα δεδομένα σε **Resort** (το **Domain Model**), το οποίο είναι πιο κατάλληλο για χρήση στο **Domain Layer** και το **Presentation Layer** της εφαρμογής. Παρακάτω παρουσιάζεται η λειτουργία του **ResortRepository** όπου με την κλήση της μεθόδου **getResorts()** ζητάει απο το **ResortDataSource** τα χιονοδρομικά. Στην συνέχεια τα μετατρέπει σε μια μορφή σύμφωνη με του **Resort** Domain Model(**mapping**) ωστε να ειναι ευχαριστα για το Domain & Presentation Layer

```
struct ResortRepository: ResortRepositoryProtocol {

    // MARK: - INJECTED 🔗 VALUES
    @Injected(\.resortDataSource) var datasource: ResortDataSourceProtocol
    @Injected(\.cacheDataSource) var cacheDataSource: KeyValueDataSourceProtocol


    // MARK: - METHODS 🌐 PUBLIC

    func getResort(id: Int) async throws -> Resort {
        do {
            let resort = try await datasource.getResort(id: id)
            return mapToResort(resort: resort)
        } catch {
            throw error
        }
    }

    func getResorts() async throws -> [Resort] {
        do {
            let result = try await datasource.getResorts()
            return mapToResortsArray(resorts: result)
        } catch {
            throw error
        }
    }
    // MARK: - METHODS 🔰 PRIVATE

    private func mapToResort(resort: ResortEntity) -> Resort {
        return Resort(resort: resort, coordinate: resort.slopes_map?.getCoordinate() ?? .init())
    }

    private func mapToResortsArray(resorts: [ResortEntity]) -> [Resort] {
        var resortsArray: [Resort] = []
        for resort in resorts {
            resortsArray.append(Resort(resort: resort, coordinate: resort.slopes_map?.getCoordinate() ?? .init()))
        }
        return resortsArray
    }

}
```

Σχήμα 3.11 Λειτουργία του Resorts Repository

Στο παράδειγμα που απεικονίζεται στο Σχήμα 3.12 παρουσιάζεται η λειτουργία του **ResortEntity** ως data model(entity).

Το **ResortEntity** είναι το μοντέλο που αντιπροσωπεύει τα δεδομένα όπως λαμβάνονται από μια εξωτερική πηγή, όπως ένα API. Αυτό το μοντέλο μπορεί να περιλαμβάνει δομές που είναι συμβατές με τα format των δεδομένων που επιστρέφονται από το backend (π.χ., JSON). Το ResortEntity είναι διαφορετικό από το Domain Model Resort, καθώς περιέχει συγκεκριμένες πληροφορίες για τη μεταφορά και την αποθήκευση των δεδομένων, όπως κωδικοποιήσεις, ενώ μπορεί να περιέχει επιπλέον μεταδεδομένα που σχετίζονται με την εξωτερική πηγή.

Το **ResortEntity** περιλαμβάνει πληροφορίες όπως:

- Αναγνωριστικό (ID) του χιονοδρομικού κέντρου.
- Όνομα του χιονοδρομικού κέντρου.
- Στοιχεία τοποθεσίας και πρόσθετες πληροφορίες που συνδέονται με το backend.

Αυτό το μοντέλο είναι βασικό για τη μεταφορά των δεδομένων από το επίπεδο των εξωτερικών πηγών στο **Domain Layer**.

```swift
// MARK: - VARIABLES 🌐 PUBLIC
let id: Int?
let name: NameEntity?
let description: String?
let location: String?
let slopes_map: String?
let elevation: ElevationEntity?
let slopes: [SlopesEntity]?
let status: String?
let activities: [String: [ActivityEntity]]?
let images: [RemoteImageEntity]?
let site: String?
let created_at: String?
let updated_at: String?

enum CodingKeys: String, CodingKey {
    case id = "id"
    case name = "name"
    case description = "description"
    case location = "location"
    case slopes_map = "slopes_map"
    case elevation = "elevation"
    case slopes = "slopes"
    case activities = "activities"
    case images = "images"
    case site = "site"
    case status = "status"
    case created_at = "created_at"
    case updated_at = "updated_at"
}
```

```swift
// MARK: - DECODING
init(from decoder: Decoder) throws {
    let values = try decoder.container(keyedBy: CodingKeys.self)
    id = try values.decodeIfPresent(Int.self, forKey: .id)
    name = try values.decodeIfPresent(NameEntity.self, forKey: .name)
    description = try values.decodeIfPresent(String.self, forKey: .description)
    location = try values.decodeIfPresent(String.self, forKey: .location)
    slopes_map = try values.decodeIfPresent(String.self, forKey: .slopes_map)
    elevation = try values.decodeIfPresent(ElevationEntity.self, forKey: .elevation)
    slopes = try values.decodeIfPresent([SlopesEntity].self, forKey: .slopes)

    status = try values.decodeIfPresent(String.self, forKey: .status)
    activities = try values.decodeIfPresent([String: [ActivityEntity]].self, forKey: .activities)
    images = try values.decodeIfPresent([RemoteImageEntity].self, forKey: .images)
    site = try values.decodeIfPresent(String.self, forKey: .site)
    created_at = try values.decodeIfPresent(String.self, forKey: .created_at)
    updated_at = try values.decodeIfPresent(String.self, forKey: .updated_at)
```

Σχήμα 3.12 Λειτουργία του Resorts Entity Data Response Model

Στο παράδειγμα που απεικονίζεται στο Σχήμα 3.13 παρουσιάζεται η λειτουργία του ResortDataSource.

Το ResortDataSource είναι υπεύθυνο για τη διαχείριση της πρόσβασης στις εξωτερικές πηγές δεδομένων, όπως APIs ή βάσεις δεδομένων. Διαχειρίζεται τις κλήσεις προς το backend για την ανάκτηση των δεδομένων που αφορούν τα χιονοδρομικά κέντρα και, στη συνέχεια, μετατρέπει τα δεδομένα που λαμβάνει σε ResortEntity.

```swift
import Foundation

protocol ResortDataSourceProtocol {
    func getResort(id: Int) async throws -> ResortEntity
    func getResorts() async throws -> [ResortEntity]
}

enum ResortDataSourceError: Error {
    case badURL(String)
    case mappingError(Swift.Error)
}

struct ResortAPI: ResortDataSourceProtocol, NetworkableProtocol, APIHandlerProtocol {

    // MARK: - PROPERTIES 🌐 PUBLIC

    // ...

    // MARK: - PROPERTIES 🔰 PRIVATE

    // ...

    // MARK: - METHODS 🌐 PUBLIC
    func getResort(id: Int) async throws -> ResortEntity {
        do {
            let data = try await getData(endPoint: .resort(id: id))
            let result: ResortEntity = try CustomMapper().get(from: data)
            return result
        } catch Network.Error.badUrl(let string) {
            throw ResortDataSourceError.badURL(string)
        } catch CustomMapperError.mapping(let error) {
            throw ResortDataSourceError.mappingError(error)
        } catch {
            throw error
        }
    }
}
```

Για παράδειγμα, όταν το **ResortRepository** ζητά δεδομένα, το **ResortDataSource** εκτελεί την κλήση προς το API, ανακτά τα δεδομένα και τα επιστρέφει ως **ResortEntity**, έτοιμα για χρήση από το Repository. Αυτή η διασύνδεση εξασφαλίζει ότι το Repository είναι ανεξάρτητο από τον τρόπο λήψης των δεδομένων, ενώ παράλληλα παρέχει τα απαραίτητα εργαλεία για την ανάκτηση των πληροφοριών με συνέπεια και ακρίβεια.

Σχήμα 3.13 Λειτουργία του Resort Data Source

Το **Data Layer** της εφαρμογής "**SnowHub**" παίζει τον καίριο ρόλο της διαχείρισης της πρόσβασης και της επεξεργασίας δεδομένων, συνδέοντας το **Domain Layer** με τις εξωτερικές πηγές. Μέσω του **ResortDataSource**, τα δεδομένα λαμβάνονται και μετατρέπονται σε μορφή που μπορεί να χρησιμοποιηθεί εσωτερικά, ενώ το **ResortRepository** διασφαλίζει την επικοινωνία του Domain Layer με τις πηγές δεδομένων χωρίς το Domain να χρειάζεται να γνωρίζει λεπτομέρειες για την προέλευση των δεδομένων. Αυτή η δομή επιτρέπει την ευκολία στη συντήρηση και την επέκταση της εφαρμογής, καθώς τυχόν αλλαγές στον τρόπο απόκτησης των δεδομένων δεν επηρεάζουν την επιχειρηματική λογική.

3.3 Navigation

Στην εφαρμογή "**SnowHub**", ένα από τα πιο σημαντικά και βασικά στοιχεία της εμπειρίας χρήστη είναι η πλοήγηση (**navigation**) μεταξύ των διαφόρων οθονών. Για να υλοποιηθεί ένα ευέλικτο και κλιμακούμενο σύστημα πλοήγησης, αναπτύξαμε μια **custom navigation logic**, η οποία βασίζεται στη χρήση του **SwiftUI**, σε συνδυασμό με **enums**, **NavigationPath** και **NavigationLink**.

3.3.1 Enums

Τα enumerations**(enums)** στη Swift χρησιμοποιούνται για να δηλώσουν ένα κοινό τύπο **(type)** για ένα σύνολο διαφορετικών τιμών οι οποίες σχετίζονται μεταξύ τους. Στην κάθε περίπτωση **(case)** ενός Enumeration μπορούμε προαιρετικά να θέσουμε μια τιμή είτε ως σταθερά, αναγκάζοντας έτσι και όλες τις υπόλοιπες περιπτώσεις να ακολουθούν το συγκεκριμένο τύπο τιμής, είτε ως συσχετισμένη τιμή **(associated value)** με τη μορφή παραμέτρου[11].

Η πλοήγηση στην εφαρμογή "**SnowHub**" διαχειρίζεται μέσω ενός **enum** που ορίζει τα διάφορα σημεία πλοήγησης. Αυτή η προσέγγιση εξασφαλίζει ότι η πλοήγηση παραμένει αυστηρά τυποποιημένη και διαχειρίσιμη, αφού κάθε προορισμός αντιπροσωπεύεται από μια συγκεκριμένη περίπτωση (case) του **enum**. Για παράδειγμα, το enum μπορεί να περιλαμβάνει περιπτώσεις όπως φαίνονται στο παρακάτω Σχήμα 3.14:

```
enum AppRoute {
    case home
    case resortDetail(resortId: String)
    case settings
}
```

Σχήμα 3.14 Αναπαράσταση AppRoute Enum

Αυτός ο τρόπος χρήσης των **enums** επιτρέπει τον εύκολο και κατανοητό χειρισμό των διαδρομών της εφαρμογής. Επιπλέον, επειδή το **enum** μπορεί να περιέχει συσχετισμένα δεδομένα (όπως το **resortId**), μπορούμε να περάσουμε παραμέτρους από μία οθόνη στην άλλη με ασφάλεια και ακρίβεια.

3.3.2 NavigationPath

Στο **SwiftUI**, χρησιμοποιήθηκε το **NavigationPath** για τη διαχείριση της πλοήγησης πολλαπλών επιπέδων, επιτρέποντας τη στοίβαξη των προβολών καθώς ο χρήστης μεταβαίνει από τη μία οθόνη στην άλλη. Το **NavigationPath** επιτρέπει την καταγραφή της τρέχουσας κατάστασης της στοίβας πλοήγησης, διασφαλίζοντας ότι η εφαρμογή μπορεί να επαναφέρει εύκολα την τρέχουσα θέση του χρήστη ή να μεταβεί απευθείας σε συγκεκριμένη προβολή όταν χρειαστεί.

Στο παρακάτω Σχήμα 3.14 έχουμε ένα παράδειγμα χρήσης του **NavigationPath** στο **NavigationStack** που χρησιμοποιείται στην εφαρμογή.

```
@StateObject private var coordinator = AppCoordinator()

// MARK: - VIEW
var body: some View {
    NavigationStack(path: $coordinator.path) {
        coordinator.build(screen: .splash)
            .navigationDestination(for: Screen.self) { screen in
                coordinator.build(screen: screen)
            }
            .sheet(item: $coordinator.sheet) { sheet in
                coordinator.build(sheet: sheet)
            }
            .fullScreenCover(item: $coordinator.fullScreenCover) { fullScreenCover in
                coordinator.build(fullScreenCover: fullScreenCover)
            }
            .fullScreenCover(item: $coordinator.errorWrapper) { error in
                coordinator.build(fullScreenCover: .error(error))
            }
            .onAppear {
                if NetworkService.networkReachabilityManager?.status == .notReachable {

                }
            }
    }
```

Σχήμα 3.15 Αναπαράσταση Navigation Stack με χρήση NavigationPath

Αυτή η προσέγγιση με το **NavigationStack** επιτρέπει την εύκολη προσθήκη ή αφαίρεση προβολών από τη στοίβα της πλοήγησης, διατηρώντας την πλοήγηση συνεπή και ευέλικτη.

3.4 Observation Pattern

Σε αντίθεση με το παραδοσιακό Observable pattern που χρησιμοποιούμε με το UIKit, η SwiftUI παρέχει έναν πιο άμεσο και απλό τρόπο παρακολούθησης των αλλαγών στα δεδομένα μέσω των **Property Wrappers**. Οι wrappers όπως **@State**, **@Published**, **@EnvironmentObject** και άλλα, παρέχουν έναν μηχανισμό παρατήρησης **(observation)** των αλλαγών στα δεδομένα και επιτρέπουν την αυτόματη ενημέρωση του UI όταν χρειάζεται, χωρίς την ανάγκη για χειροκίνητη διαχείριση της ενημέρωσης καθιστώντας την εφαρμογή πιο αποκριτική και ευέλικτη.

Συγκεκριμένα οι λειτουργίες του κάθε **PropertyWrapper** που χρησιμοποιείται στη **SwiftUI** για **Observation** είναι οι εξής:

@State

- Χρησιμοποιείται για να αποθηκεύσει την τοπική κατάσταση ενός View. Είναι κατάλληλο για απλά δεδομένα που ανήκουν αποκλειστικά στο View.
- Όταν αλλάζει η τιμή μιας @State μεταβλητής, το SwiftUI ενημερώνει το UI που σχετίζεται με αυτήν τη μεταβλητή.

@StateObject

- Χρησιμοποιείται για να δημιουργήσει και να διαχειριστεί ένα **ObservableObject** μέσα σε ένα View. Είναι υπεύθυνο για την αρχικοποίηση και τη διατήρηση της κατάστασης του αντικειμένου σε όλη τη διάρκεια ζωής του View.

@Published

- Χρησιμοποιείται μέσα σε κλάσεις που υλοποιούν το **ObservableObject**. Όταν αλλάζει η τιμή μιας **@Published** μεταβλητής, ενημερώνει αυτόματα τα Views που παρακολουθούν (**observe**) το αντικείμενο αυτό.

@EnvironmentObject

- Χρησιμοποιείται για να διανείμει ένα **ObservableObject** μέσα σε πολλά Views, χωρίς να χρειάζεται να το περάσετε ρητά μέσω παραμέτρων. Όταν αλλάζουν οι τιμές του **EnvironmentObject**, ενημερώνονται όλα τα Views που το χρησιμοποιούν.

@Environment

- Επιτρέπει την πρόσβαση σε τιμές που διανέμονται από το περιβάλλον (**environment**) του SwiftUI, όπως το **colorScheme** ή το **locale**. Χρησιμοποιείται για πρόσβαση σε συστηματικά ή καθολικά δεδομένα της εφαρμογής.

@ObservedObject

- Χρησιμοποιείται για να παρακολουθεί (**observe**) ένα **ObservableObject** που δημιουργήθηκε από κάπου αλλού (όχι μέσα στο View). Δεν είναι υπεύθυνο για την αρχικοποίηση ή τη διατήρηση της κατάστασης, απλώς παρακολουθεί τις αλλαγές.

@Binding

- Χρησιμοποιείται για τη δημιουργία συνδέσμων (**bindings**) σε δεδομένα που ανήκουν σε άλλο View. Επιτρέπει σε δύο Views να μοιράζονται και να ενημερώνουν τα ίδια δεδομένα.

Συνοπτικά:

- **@State**: Τοπική κατάσταση σε ένα View.
- **@StateObject**: Διαχείριση ενός ObservableObject που δημιουργείται μέσα στο View.
- **@Published**: Παρακολούθηση αλλαγών μέσα σε ObservableObject.
- @**EnvironmentObject**: Διαμοιρασμός ενός ObservableObject σε πολλά Views μέσω του περιβάλλοντος.
- **@Environment**: Πρόσβαση σε περιβαλλοντικές τιμές του SwiftUI.
- **@ObservedObject**: Παρακολούθηση ενός ObservableObject που διαχειρίζεται αλλού.
- **@Binding**: Σύνδεση δεδομένων μεταξύ διαφορετικών Views.

3.5 Managers

Στην αρχιτεκτονική λογισμικού Swift, οι managers, που υλοποιούνται ως κλάσεις, αναλαμβάνουν τον ρόλο διαχείρισης συγκεκριμένων επιχειρηματικών λογικών. Αυτή η οργάνωση επιτρέπει την ομαδοποίηση συναφών λειτουργιών, βελτιώνοντας την αναγνωσιμότητα, τη συντήρηση και την επεκτασιμότητα του κώδικα. Παρακάτω στο Σχήμα 3.16 έχουμε τμήμα του **NetworkManager**, ο οποίος είναι **final class** κλάση και συγκεντρώνει κώδικα που είναι υπεύθυνος για τη διαχείριση όλων των δικτυακών αιτημάτων **(network requests)** της εφαρμογής. Αυτή η κλάση έχει σχεδιαστεί με τη χρήση της νέας προσέγγισης **async/await** της Swift, που εισάγει τη λογική της **structured concurrency**, επιτρέποντας την ασύγχρονη εκτέλεση δικτυακών αιτημάτων με έναν πιο απλό και κατανοητό τρόπο[20].

Παρακάτω στο Σχήμα 3.16 παρουσιάζεται η βασική μέθοδος που εκτελεί ένα δικτυακό αίτημα για την ανάκτηση δεδομένων από ένα URL. Η μέθοδος **makeRequest** πραγματοποιεί ένα αίτημα στο δίκτυο χρησιμοποιώντας το πρότυπο **async/await** για να επιτρέψει ασύγχρονες κλήσεις χωρίς την πολυπλοκότητα των **closures** και των callbacks. Η μέθοδος δέχεται ως παράμετρο ένα αντικείμενο με το πρωτόκολλο **RequestConvertible** προέκταση της βιβλιοθήκης **Alamofire**(για την οποια εγινε αναφορά παραπάνω) για την καλυτερη δημιουργία του **URLRequest** που απαιτει η Swift, το οποίο χρησιμοποιείται για να κατασκευάσει το αίτημα (**request**) που θα σταλεί στο δίκτυο.

```
/// Makes a network request using async/await. •••
func makeRequest(endpoint: RequestConvertible) async throws -> Data {
    return try await withCheckedThrowingContinuation { continuation in
        session.request(endpoint)
            .validate()
            .responseData { response in
                /// Log the response
                NetworkLogger.logResponse(response)
                /// Handle the response result
                switch response.result {
                case .success(let data):
                    continuation.resume(returning: data)
                case .failure(let error):
                    /// Handle failure scenarios
                    if let data = response.data {
                        if let json = try? JSONSerialization.jsonObject(with: data, options: .mutableContainers) {
                            print("Response JSON: \(json)")
                        } else {
                            print("Failed to serialize response data to JSON.")
                        }
                    } else {
                        print("No response data.")
                    }
                    continuation.resume(throwing: self.mapError(error: error))
                }
            }
    }
}
```

Σχήμα 3.16 Αναπαράσταση makeRequest() του Network Manager

Ο **NetworkManager** στην εφαρμογή "SnowHub" αναλαμβάνει διάφορα καθήκοντα που σχετίζονται με την επικοινωνία με το δίκτυο, όπως:

- **Ανάκτηση δεδομένων από APIs**: Καλεί εξωτερικές υπηρεσίες μέσω **HTTP** αιτημάτων για την ανάκτηση δεδομένων, όπως πληροφορίες για χιονοδρομικά κέντρα, καιρικές συνθήκες ή πληροφορίες χρηστών.
- **Επεξεργασία των αποτελεσμάτων**: Τα δεδομένα που λαμβάνονται από τα δικτυακά αιτήματα επεξεργάζονται, μετατρέπονται σε κατάλληλα μοντέλα δεδομένων της εφαρμογής, και στη συνέχεια διανέμονται στα υπόλοιπα μέρη της εφαρμογής για χρήση.
- **Διαχείριση σφαλμάτων (Error handling)**: Χρησιμοποιεί τη δυνατότητα throws για την ασφαλή διαχείριση σφαλμάτων, όπως προβλήματα συνδεσιμότητας ή μη έγκυρες απαντήσεις από τον server. Αυτό επιτρέπει την καθαρή διαχείριση σφαλμάτων και την εύκολη αναφορά τους στο UI.

Ο **NetworkManager** στην εφαρμογή "**SnowHub**" προσφέρει μια καθαρή και σύγχρονη λύση για τη διαχείριση δικτυακών αιτημάτων, βασισμένη στο **async/await** και το **structured concurrency**. Αυτή η προσέγγιση καθιστά τη διαχείριση των δικτυακών κλήσεων πιο κατανοητή και ευέλικτη, ενώ εξασφαλίζει ότι η εφαρμογή παραμένει αποκριτική και αξιόπιστη κατά τη διάρκεια της επικοινωνίας με εξωτερικές υπηρεσίες.

3.6 Extensions

Οι επεκτάσεις (**extensions**) στη Swift παρέχουν έναν τρόπο να προσθέσουμε επιπλέον λειτουργικότητα σε υπάρχουσες κλάσεις, δομές, ή πρωτόκολλα χωρίς να χρειάζεται να αλλάξουμε τον αρχικό κώδικα αυτών. Μας επιτρέπουν να εμπλουτίσουμε τις δυνατότητες των τύπων προσθέτοντας νέες ιδιότητες και μεθόδους. Αυτό είναι ιδιαίτερα χρήσιμο όταν οι ενσωματωμένες λειτουργίες μιας κλάσης δεν είναι επαρκείς για τις ανάγκες μας και θέλουμε να προσαρμόσουμε τη συμπεριφορά ή να προσθέσουμε χρήσιμες βοηθητικές μεθόδους.

Ένα χαρακτηριστικό παράδειγμα επέκτασης είναι η προσθήκη μεθόδων στην κλάση String, για να διευκολύνουμε την επαναλαμβανόμενη χρήση κώδικα, όπως η τοπικοποίηση (**localization**). Στο παρακάτω παράδειγμα στο Σχήμα 3.17, η επέκταση προσθέτει τη δυνατότητα διαχείρισης ενός String με βάση ένα συγκεκριμένο αρχείο **Localizable**

```
extension String {

    // MARK: - Localization

    func localized(tableName: String = "Localizable") -> String {
        return NSLocalizedString(self, tableName: tableName, value: "**\(self)**", comment: "")
    }

}
```

Σχήμα 3.17 Αναπαράσταση localized στο String Extension

Αυτή η επέκταση διευκολύνει την τοπικοποίηση των κειμένων στην εφαρμογή. Αντί να χρησιμοποιούμε συνεχώς τη συνάρτηση **NSLocalizedString** κάθε φορά που θέλουμε να τοπικοποιήσουμε μια συμβολοσειρά, μπορούμε απλά να καλούμε τη μέθοδο localized πάνω σε οποιοδήποτε αντικείμενο τύπου **String**, καθιστώντας τον κώδικα πιο καθαρό και ευανάγνωστο.

3.7 Protocols

Τα πρωτόκολλα (**protocols**) στη Swift είναι ένα είδος **blueprint** (**πρότυπο**), το οποίο καθορίζει τις ιδιότητες, τις μεθόδους και άλλες λειτουργίες που πρέπει να υιοθετήσει μια κλάση, δομή, ή enum που το ακολουθεί. Είναι ένας ισχυρός τρόπος για να ορίσουμε ένα σύνολο κανόνων που επιβάλλονται σε πολλαπλές οντότητες, διασφαλίζοντας ότι αυτές ακολουθούν μια συγκεκριμένη δομή και συμπεριφορά[12].

Ένα παράδειγμα χρήσης πρωτοκόλλου είναι το **LocalizableProtocol**, το οποίο διαχειρίζεται τη διαδικασία τοπικοποίησης για οντότητες που υιοθετούν το πρωτόκολλο. Αυτό επιτρέπει στις διάφορες οντότητες να αποκτούν τοπικοποιημένες συμβολοσειρές χωρίς να χρειάζεται να επαναλαμβάνουμε τη λογική της τοπικοποίησης σε κάθε σημείο του κώδικα.

Παρακάτω στο Σχήμα 3.18 έχουμε το παράδειγμα του **LocalizableProtocol**

```
import Foundation

protocol LocalizableProtocol {
    var tableName: String { get }
    var localized: String { get }
}

extension LocalizableProtocol where Self: RawRepresentable, Self.RawValue == String {
    var localized: String {
        return rawValue.localized(tableName: tableName)
    }
}
```

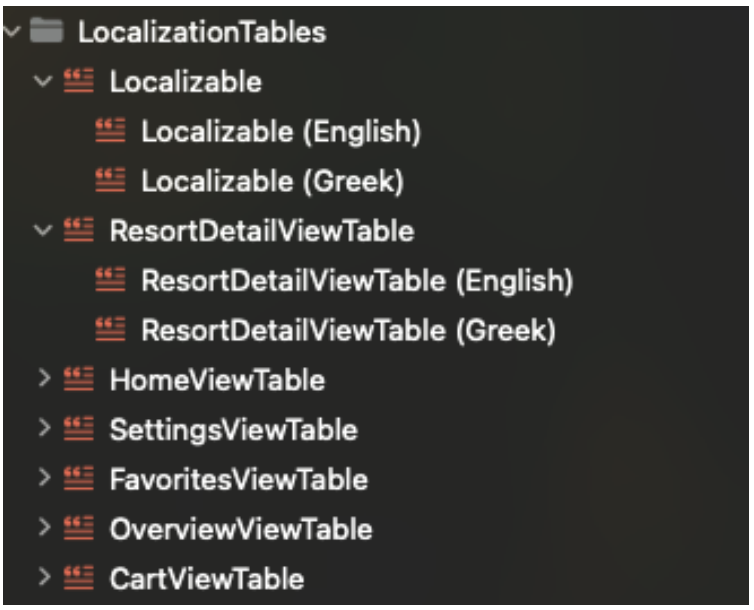
Σχήμα 3.18 Αναπαράσταση του LocalizableProtocol

Σε αυτό το παράδειγμα, το πρωτόκολλο **LocalizableProtocol** μπορεί να υιοθετηθεί από τύπους που είναι **RawRepresentable** και έχουν ως **RawValue** μια **String**. Το πρωτόκολλο απαιτεί την ύπαρξη ενός πίνακα τοπικοποίησης (**tableName**) και της μεθόδου **localized**. Χρησιμοποιώντας αυτήν την επέκταση, οποιοσδήποτε τύπος (π.χ. **enum**) μπορεί εύκολα να αποκτήσει δυνατότητες τοπικοποίησης, χωρίς να χρειάζεται να επαναλάβουμε τη λογική.

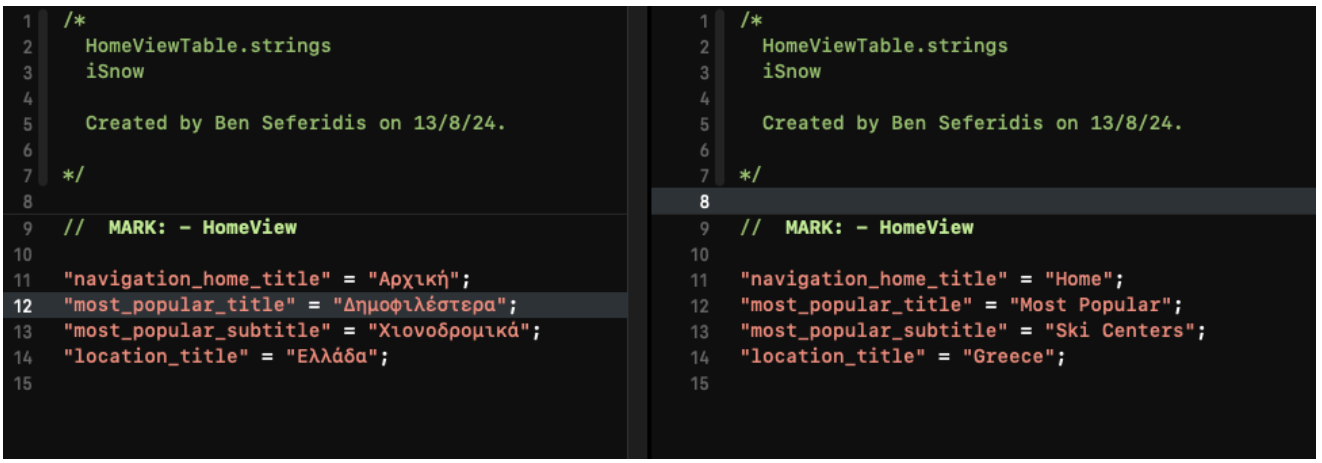
3.8 Localization

Η διαδικασία προσαρμογής μιας εφαρμογής σε διαφορετικές γλώσσες και περιοχές ονομάζεται Localization. Για να γίνει αυτό, δημιουργούμε ένα ειδικό αρχείο που ονομάζεται Localizable. Αυτό το αρχείο χωρίζεται σε μικρότερα κομμάτια, ένα για κάθε γλώσσα που θέλουμε να υποστηρίξουμε (όπως

φαίνεται στο Σχήμα 3.19). Για παράδειγμα, στο SnowHub app μας, υπάρχει ένα κομμάτι για τα Αγγλικά και ένα για τα Ελληνικά. Όταν ανοίγεις την εφαρμογή, η γλώσσα προσαρμόζεται αυτόματα στη γλώσσα που έχεις επιλέξει στο τηλέφωνό σου. Αλλά μπορείς και εσύ να την αλλάξεις από το προφίλ σου. Στο Σχήμα 3.20 μπορείς να δεις ένα παράδειγμα από αυτό το αρχείο, τόσο στα Ελληνικά όσο και στα Αγγλικά. Πέραν του βασικού Localizable αρχείου, για την καλύτερη διαχείριση των λεκτικών, η κάθε οθόνη έχει τα δικά της 2 αρχεία που περιέχουν αγγλικά και ελληνικά λεκτικά.



Σχήμα 3.19 Διαχωρισμός Localizable αρχείων

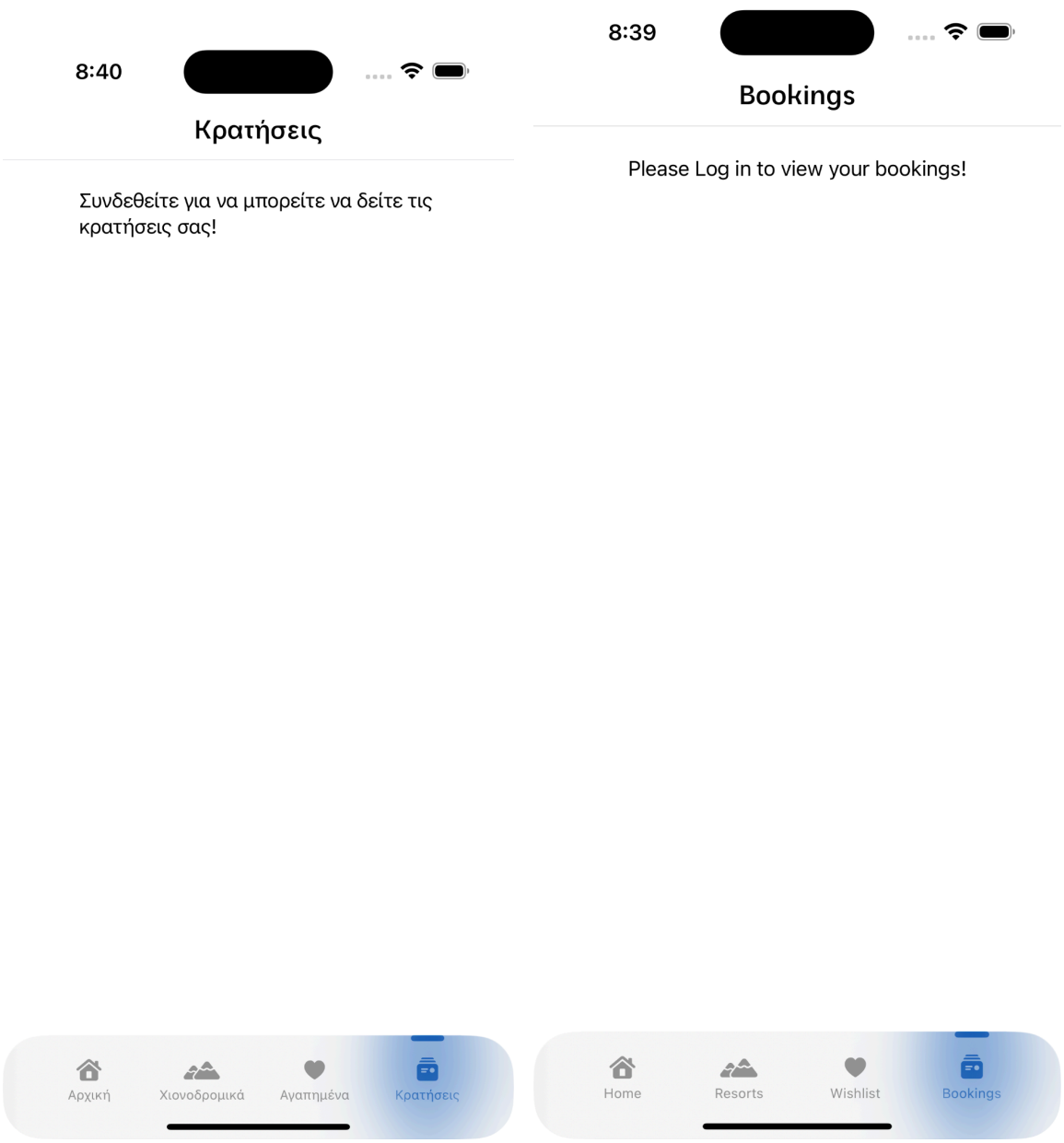


Σχήμα 3.20 Αρχείο Localizable στα Ελληνικά και στα Αγγλικά

Κεφάλαιο 4ο: Παρουσίαση Οθονών Διεπαφής Χρηστών

Στο συγκεκριμένο κεφάλαιο, παρουσιάζεται η τελική έκδοση της εφαρμογής, όπως διαμορφώθηκε μέσω της χρήσης των παραπάνω τεχνολογιών και εργαλείων. Μέσα από συγκεκριμένα παραδείγματα χρήσης και στιγμιότυπα (screenshots) των οθονών της εφαρμογής, θα γίνει μια αναλυτική περιήγηση στις λειτουργίες και δυνατότητες που προσφέρει. Προκειμένου να αποφευχθεί ο πλατειασμός και η επανάληψη τοσο σε επεξηγησεις οσο και εικόνες με παρόμοιο περιεχόμενο, στα Σχήματα 4.1 και 4.2 απεικονίζονται η συμπεριφορά της διεπαφής χρήστη σε light και dark mode καθώς και οι υποστηριζόμενες γλώσσες της εφαρμογής (Ελληνικά και Αγγλικά) αντίστοιχα. Όλες οι οθόνες της εφαρμογής υποστηρίζουν αυτές τις λειτουργίες και αντιδρούν με παρόμοιο τρόπο, όπως φαίνεται στα εν λόγω σχήματα.

Σχήμα 4.1 Light - Dark Mode



Σχήμα 4.2 Ελληνικό - Αγγλικό Localization

4.1 Οθόνη Σύνδεσης / Εγγραφής - Login / Register Screen

Οι οθόνη σύνδεσης και εγγραφής στην εφαρμογή **SnowHub** είναι ένα κοινό container view που περιέχει τις δύο οθόνες σύνδεσης και εγγραφή μαζί, προσφέροντας στους χρήστες τη δυνατότητα να επιλέξουν είτε να συνδεθούν στον υπάρχοντα λογαριασμό τους, είτε να δημιουργήσουν έναν νέο. Αυτός ο σχεδιασμός προσφέρει μια ενιαία εμπειρία χρήστη, όπου και οι δύο επιλογές παρουσιάζονται με συνεπή και φιλική προς τον χρήστη διάταξη διατηρώντας το ίδιο παρουσιαστικό και εστιάζοντας στην καλύτερη πλοήγησή του μεταξυ των δυο αυτών επιλογών. Οι οθόνες μοιράζονται παρόμοια δομή **και στυλ, εξασφαλίζοντας μια ομαλή και ενιαία εμπειρία κατά την είσοδο στην εφαρμογή.**

Στοιχεία του Κοινού Container View:

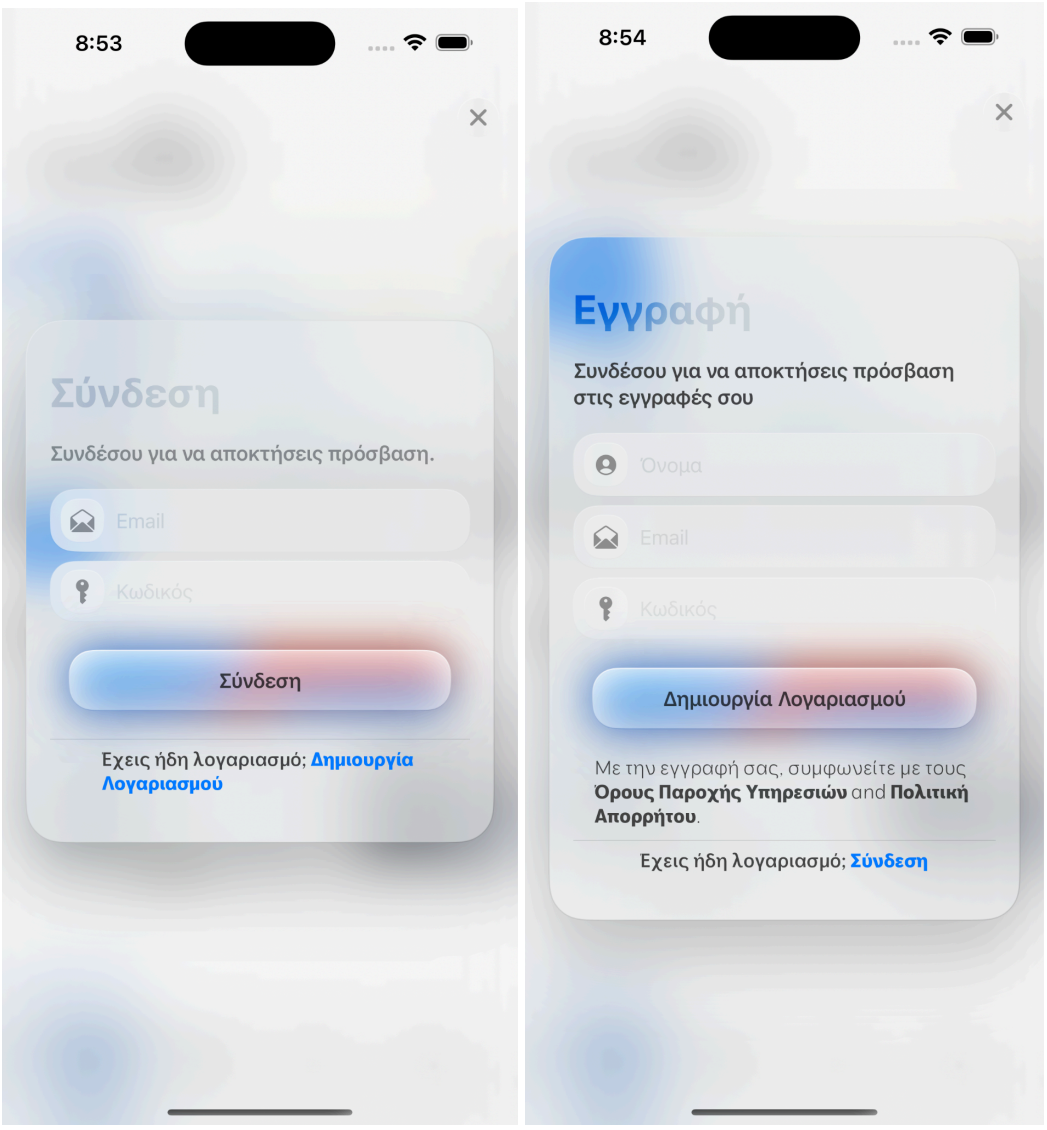
- **Πεδία Εισαγωγής(Text Fields)**: Και οι δύο οθόνες περιλαμβάνουν πεδία για την εισαγωγή του email και του κωδικού, με την οθόνη εγγραφής να έχει επιπλέον πεδίο για την εισαγωγή ονόματος. Τα πεδία είναι καθαρά και ευδιάκριτα, συνοδευόμενα από εικονίδια που αντιπροσωπεύουν τη λειτουργία κάθε πεδίου (π.χ. εικονίδιο φακέλου για το email, κλειδί για τον κωδικό πρόσβασης).
- **Κουμπί Δράσης (Action Button)**: Στο κάτω μέρος και των δύο οθονών, υπάρχει ένα κύριο κουμπί δράσης. Στην οθόνη σύνδεσης, το κουμπί αναγράφει "**Σύνδεση**", ενώ στην οθόνη εγγραφής, το κουμπί αναγράφει "**Δημιουργία Λογαριασμού**". Τα κουμπιά είναι ευδιάκριτα και εύκολα στη χρήση, προσφέροντας σαφήνεια στην ενέργεια που θα πραγματοποιηθεί.
- **Σύνδεσμοι Μετάβασης**: Σε κάθε οθόνη υπάρχει ένας σύνδεσμος που επιτρέπει τη γρήγορη εναλλαγή μεταξύ σύνδεσης και εγγραφής. Στην οθόνη σύνδεσης, ο σύνδεσμος κατευθύνει τους χρήστες στην εγγραφή και στην οθόνη εγγραφής, ο σύνδεσμος κατευθύνει τους χρήστες πίσω στη σύνδεση. Οι σύνδεσμοι εμφανίζονται με μπλε χρώμα για να τραβούν την προσοχή και να διευκολύνουν την πλοήγηση και παράλληλα στοχεύουν στο να διατηρήσουν μια ομοιομορφία στα χρώματα που χρησιμοποιεί η εφαρμογή.
- **Κοινή Εμφάνιση**: Και οι δύο οθόνες χρησιμοποιούν το ίδιο στυλ παρουσίασης, με απαλούς χρωματικούς τόνους και διαφάνεια στο background, δημιουργώντας μια ευχάριστη και μοντέρνα αίσθηση στον χρήστη. Ο ενιαίος σχεδιασμός εξασφαλίζει μια συνεπή εμπειρία και δίνει την αίσθηση συνέχειας κατά την εναλλαγή μεταξύ των οθονών.

Λειτουργίες του Κοινού Container View:

- **Σύνδεση**: Στην οθόνη σύνδεσης, ο χρήστης μπορεί να εισαγάγει το **email** και τον **κωδικό** πρόσβασής του για να αποκτήσει πρόσβαση στον λογαριασμό του. Η διαδικασία αυτή είναι άμεση και γρήγορη, επιτρέποντας στον χρήστη να συνδεθεί και να χρησιμοποιήσει τις εξατομικευμένες λειτουργίες της εφαρμογής.
- **Εγγραφή**: Στην οθόνη εγγραφής, οι νέοι χρήστες μπορούν να δημιουργήσουν έναν νέο λογαριασμό εισάγοντας **όνομα**, **email** και **κωδικό**. Μόλις συμπληρωθούν τα πεδία και πατηθεί το κουμπί "**Δημιουργία Λογαριασμού**", η εφαρμογή δημιουργεί τον λογαριασμό και ο χρήστης μπορεί να συνδεθεί αμέσως για να ξεκινήσει να χρησιμοποιεί την εφαρμογή.
- **Εναλλαγή Μεταξύ Σύνδεσης και Εγγραφής**: Μέσω των συνδέσμων "Έχεις ήδη λογαριασμό; Σύνδεση" και "Δημιουργία Λογαριασμού", ο χρήστης μπορεί εύκολα να εναλλάσσεται μεταξύ των δύο οθονών χωρίς να χρειάζεται να επιστρέψει στο αρχικό μενού. Αυτή η λειτουργία προσφέρει ευελιξία και βελτιώνει τη συνολική εμπειρία πλοήγησης.
- **Όροι Παροχής Υπηρεσιών και Πολιτική Απορρήτου**: Στην οθόνη εγγραφής, υπάρχει υπενθύμιση στον χρήστη ότι, δημιουργώντας λογαριασμό, συμφωνεί με τους όρους χρήσης

και την πολιτική απορρήτου. Αυτό προσθέτει διαφάνεια και ενημερώνει τον χρήστη για τις νομικές προϋποθέσεις της εφαρμογής.

Ο σχεδιασμός αυτού του κοινόχρηστου container view προσφέρει μια ευχάριστη και συνεπή εμπειρία στον χρήστη, είτε πρόκειται για τη σύνδεση σε έναν υπάρχοντα λογαριασμό είτε για τη δημιουργία νέου λογαριασμού. Η σαφής διάταξη και η εύκολη εναλλαγή μεταξύ σύνδεσης και εγγραφής βοηθούν στη μείωση της πολυπλοκότητας και ενισχύουν την αμεσότητα, καθιστώντας τη διαδικασία πλοήγησης απλή και ευχάριστη. Στο παρακάτω Σχήμα 4.3 απεικονίζεται το συγκεκριμένο Container View που φιλοξενεί τις οθόνες σύνδεσης και εγγραφής.

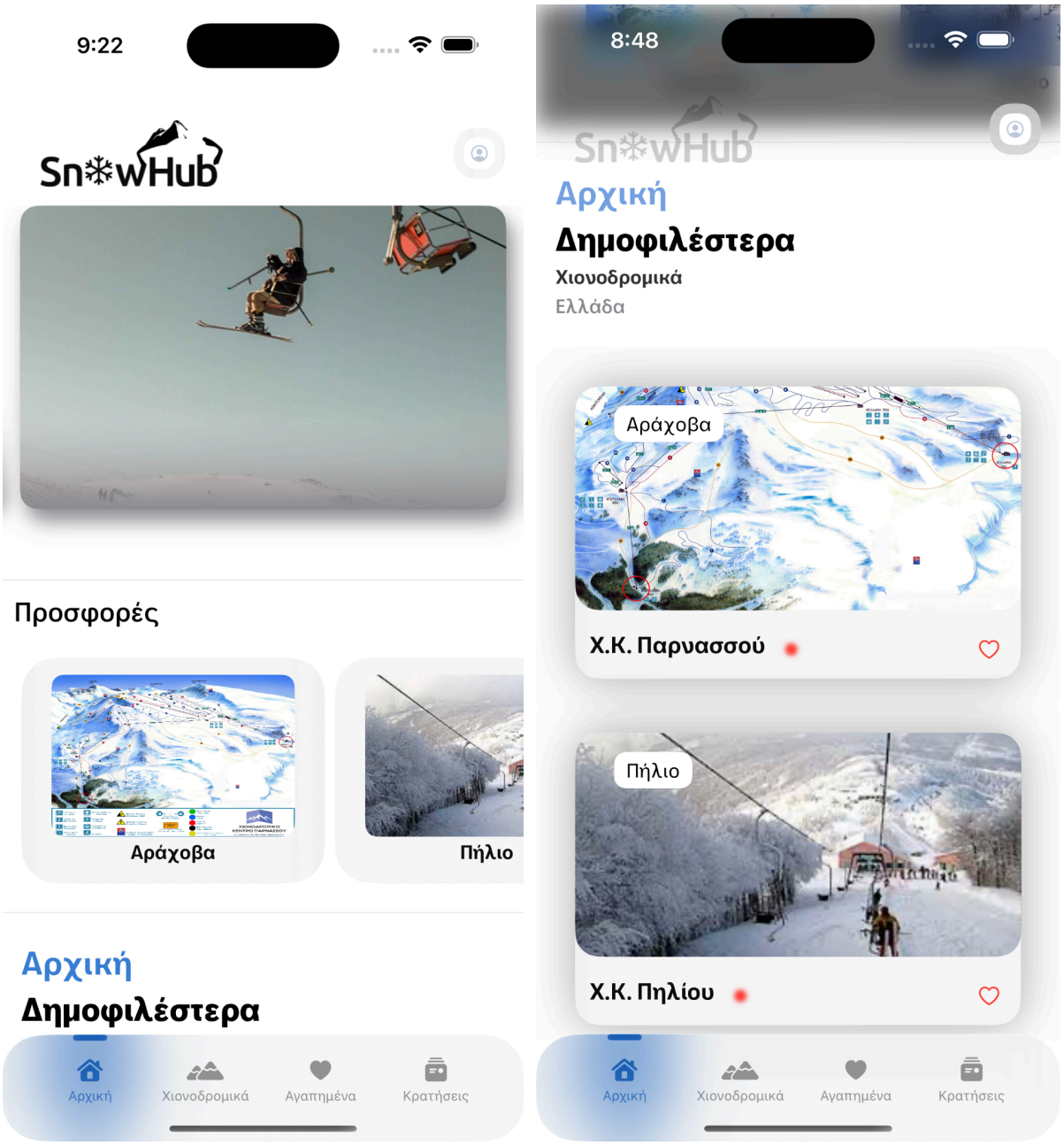


Σχήμα 4.3 Οθόνη Σύνδεσης / Εγγραφής - Login / Register Screen

4.2 Οθόνη Αρχική - Home Screen

Η Αρχική Οθόνη της εφαρμογής SnowHub προσφέρει μια εισαγωγή στα δημοφιλέστερα χιονοδρομικά κέντρα. Όπως απεικονίζεται και στο Σχήμα 4.4, βλέπουμε μια συλλογή απο φωτογραφίες απο τα χιονοδρομικά με τυχαία σειρά, ακολουθούμενη από προσφορές για χιονοδρομικά, όπως η Αράχωβα και το Πήλιο. Ο χρήστης μπορεί να περιηγηθεί σε αυτά τα χιονοδρομικά μέσω των καρτελών που παρουσιάζονται και των προσφορών.

Στη συνέχεια, παρουσιάζονται οι δημοφιλέστεροι προορισμοί που προτείνονται στον χρήστη, όπως το Χ.Κ. Παρνασσού και το Χ.Κ. Πηλίου, σε μια καρτέλα. Στο κάτω μέρος της οθόνης υπάρχει το μενού το οποίο εξυπηρετεί την πλοήγηση του χρήστη στις διάφορες οθόνες τις εφαρμογής. Επιλέγοντας την δεύτερη επιλογή των χιονοδρομικών από το μενού, ο χρήστης μπορεί να μεταφερθεί στην οθόνη των χιονοδρομικών που απεικονίζονται στο χάρτη της Ελλάδος.



Σχήμα 4.4 Αρχική Οθόνη - Home Screen

4.3 Οθόνη Πληροφοριών Χιονοδρομικού - Resorts Informations Screen

Συλλογή Εικόνων Χιονοδρομικού

Στην κορυφή της οθόνης εμφανίζεται μια μεγάλη **αρχική εικόνα** από το χιονοδρομικό κέντρο, η οποία λειτουργεί ως εξώφυλλο και βρίσκεται σε μια **συλλογή εικόνων** από το εκάστοτε χιονοδρομικό. η χρήση αρχικής εικόνας δίνει στον χρήστη μια πρώτη εντύπωση της τοποθεσίας και δημιουργεί μια πιο οπτική σύνδεση με το μέρος. Η εικόνα μπορεί να είναι από τα χιονισμένα τοπία του χιονοδρομικού, τις εγκαταστάσεις ή τις πίστες

Καρτέλες Πλοήγησης

Ακριβώς κάτω από την συλλογή εικόνων υπάρχουν καρτέλες πλοήγησης που επιτρέπουν στον χρήστη να εναλλάσσεται ανάμεσα σε διαφορετικές κατηγορίες πληροφοριών. Οι επιλογές που εμφανίζονται είναι οι εξής:

- **Πληροφορίες**: Παρουσιάζει βασικά δεδομένα για το χιονοδρομικό.
- **Πίστες**: Παρέχει πληροφορίες για τις διαθέσιμες πίστες και το επίπεδο δυσκολίας τους.
- **Αναβατήρες**: Ενημερώνει τον χρήστη για τους αναβατήρες που είναι διαθέσιμοι.
- **Κάμερες**: Εδώ εμφανίζονται live εικόνες από τις κάμερες του χιονοδρομικού, επιτρέποντας στους χρήστες να δουν τις συνθήκες σε πραγματικό χρόνο.

4.3.1 Καρτέλα Πληροφοριών (Overview tab)

Περιγραφή και Δραστηριότητες

Στην καρτέλα "**Πληροφορίες**" εμφανίζεται μια μικρή περιγραφή του χιονοδρομικού κέντρου, που ενημερώνει τον χρήστη για τη γεωγραφική τοποθεσία του, την υψομετρική διαφορά, και τον συνολικό αριθμό των πιστών που διαθέτει. Επίσης, εμφανίζονται **σημαντικές δραστηριότητες** που μπορεί να πραγματοποιήσει ο χρήστης στο χιονοδρομικό, όπως:

- Σκι
- Snowboard
- Έλκηθρο
- Snowmobile
- Σαλέ

Αυτές οι δραστηριότητες είναι παρουσιασμένες με τη μορφή μικρών εικονιδίων, καθιστώντας την πληροφορία πιο προσιτή και κατανοητή.

Πρόσθετες Πληροφορίες

Στο κάτω μέρος της ενότητας, παρουσιάζονται πληροφορίες που αφορούν τη συνολική κατάσταση του χιονοδρομικού κέντρου, όπως η **τελευταία χιονόπτωση**, το **ύψος του χιονιού** και ο **συνολικός αριθμός πιστών**. Αυτές οι πληροφορίες είναι κρίσιμες για τον επισκέπτη που θέλει να σχεδιάσει την επίσκεψή του με βάση τις τρέχουσες συνθήκες. Παράλληλα, προσφέρεται και η δυνατότητα αγοράς εισιτηρίου μέσα από το σχετικό κουμπί το οποίο μας μεταφέρει στην οθόνη αγοράς εισιτηρίου.

Χάρτες και Οδηγίες

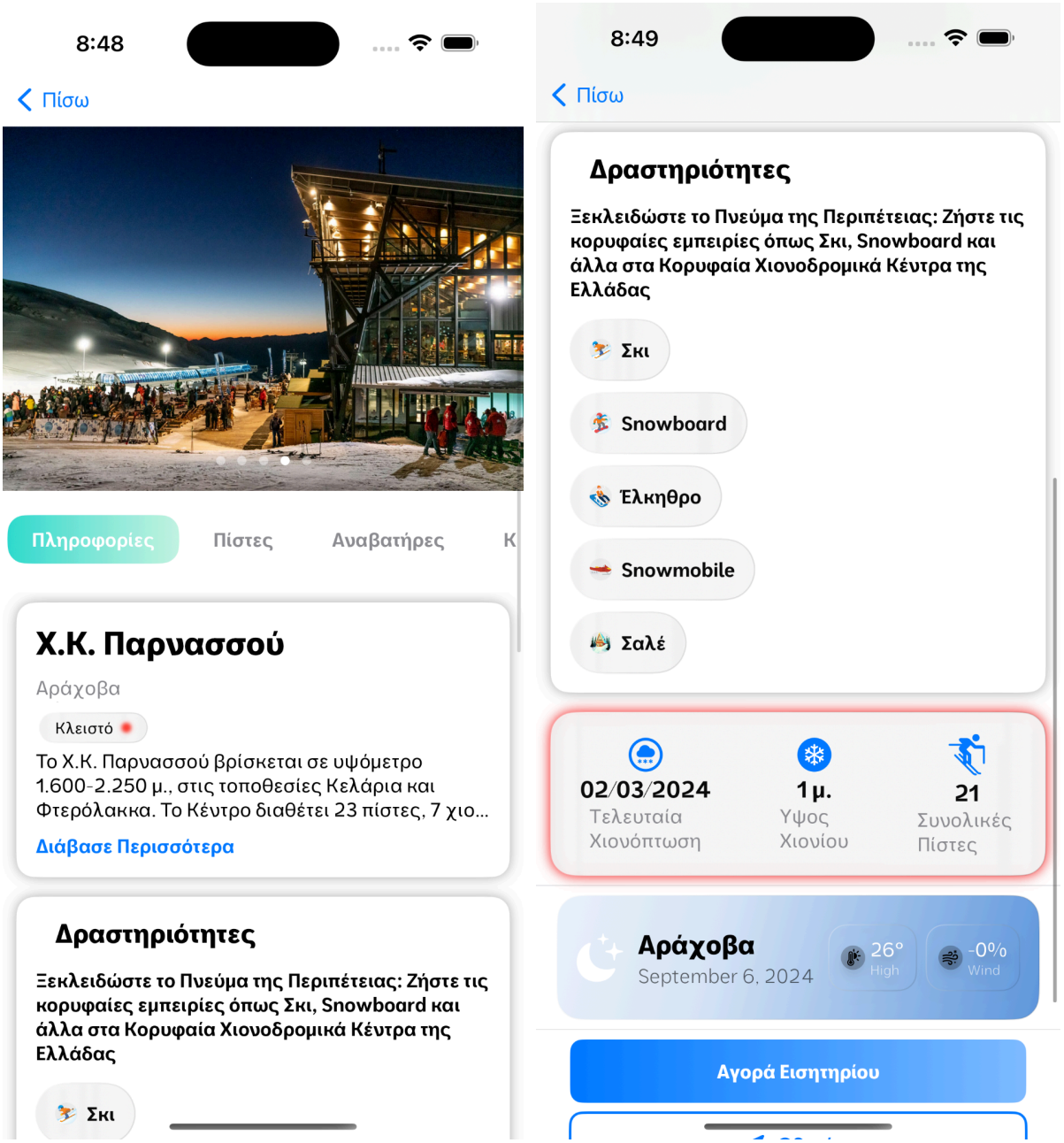
Παρέχεται η δυνατότητα να ανοίξει κάποιος χάρτες με οδηγίες για την πρόσβαση στο χιονοδρομικό μέσω των εφαρμογών **Apple Maps** ή **Google Maps**. Η λειτουργία αυτή διευκολύνει τον χρήστη να βρει το δρόμο προς το χιονοδρομικό με ακρίβεια, ενώ ταυτόχρονα προσφέρεται η επιλογή μεταξύ δύο δημοφιλών εφαρμογών πλοήγησης. Στο παρακάτω Σχήμα 4.7 απεικονίζεται η καρτέλα επιλογής χάρτη πλοήγησης.

Καιρικές Συνθήκες

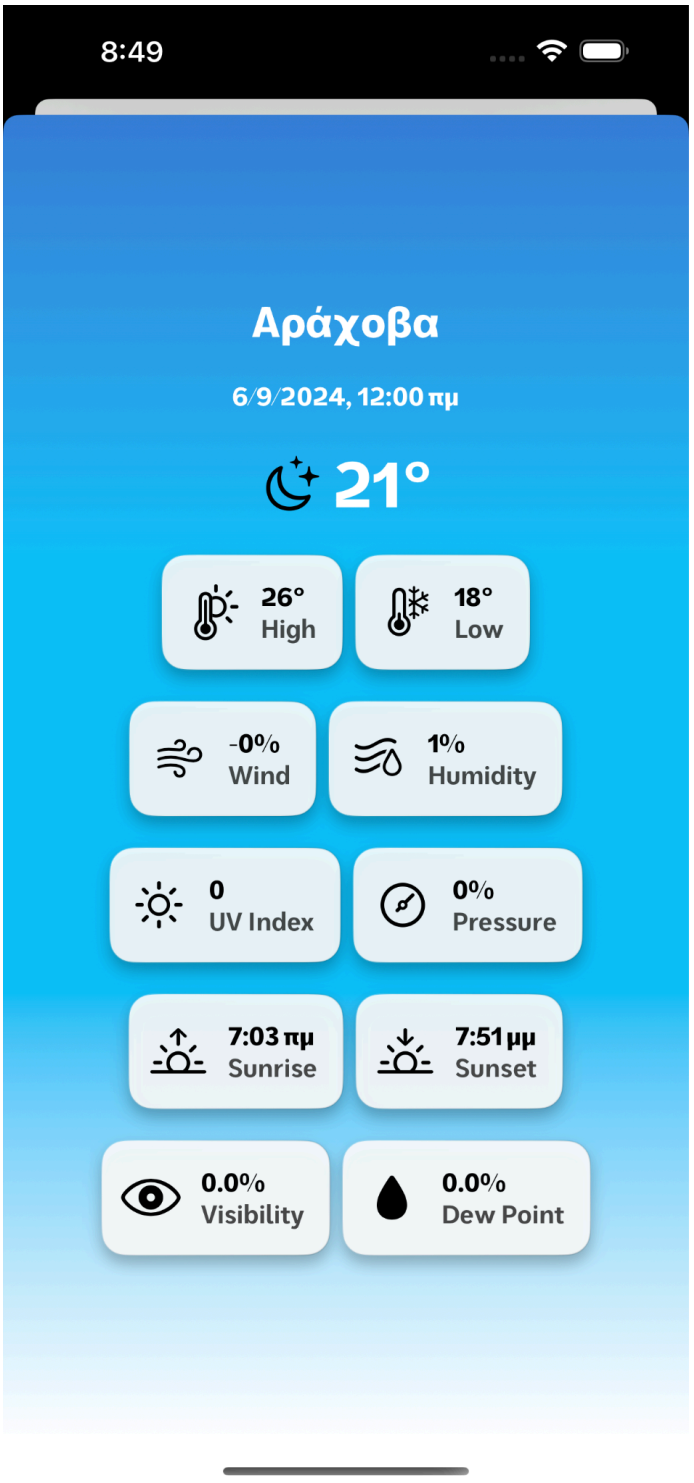
Παρουσιάζονται αναλυτικά οι τρέχουσες καιρικές συνθήκες, όπως η θερμοκρασία, η ταχύτητα και κατεύθυνση του ανέμου, η υγρασία, η πίεση και η ορατότητα. Ο χρήστης έχει στη διάθεσή του πλήρη πρόβλεψη των συνθηκών για να προγραμματίσει την επίσκεψή του με βάση τον καιρό. Στο παρακάτω Σχήμα 4.6 απεικονίζεται η καρτέλα καιρικών συνθηκών.

Η **Οθόνη Πληροφοριών Χιονοδρομικού** στην εφαρμογή **SnowHub** προσφέρει μια πλήρη και πολυδιάστατη ενημέρωση για το κάθε χιονοδρομικό κέντρο, παρέχοντας όλες τις απαραίτητες πληροφορίες για την καλύτερη εμπειρία του χρήστη. Η οθόνη έχει σχεδιαστεί με σκοπό να είναι ευέλικτη, άμεση και κατανοητή, εξασφαλίζοντας ότι ο επισκέπτης του χιονοδρομικού θα έχει στη διάθεσή του ό,τι χρειάζεται.

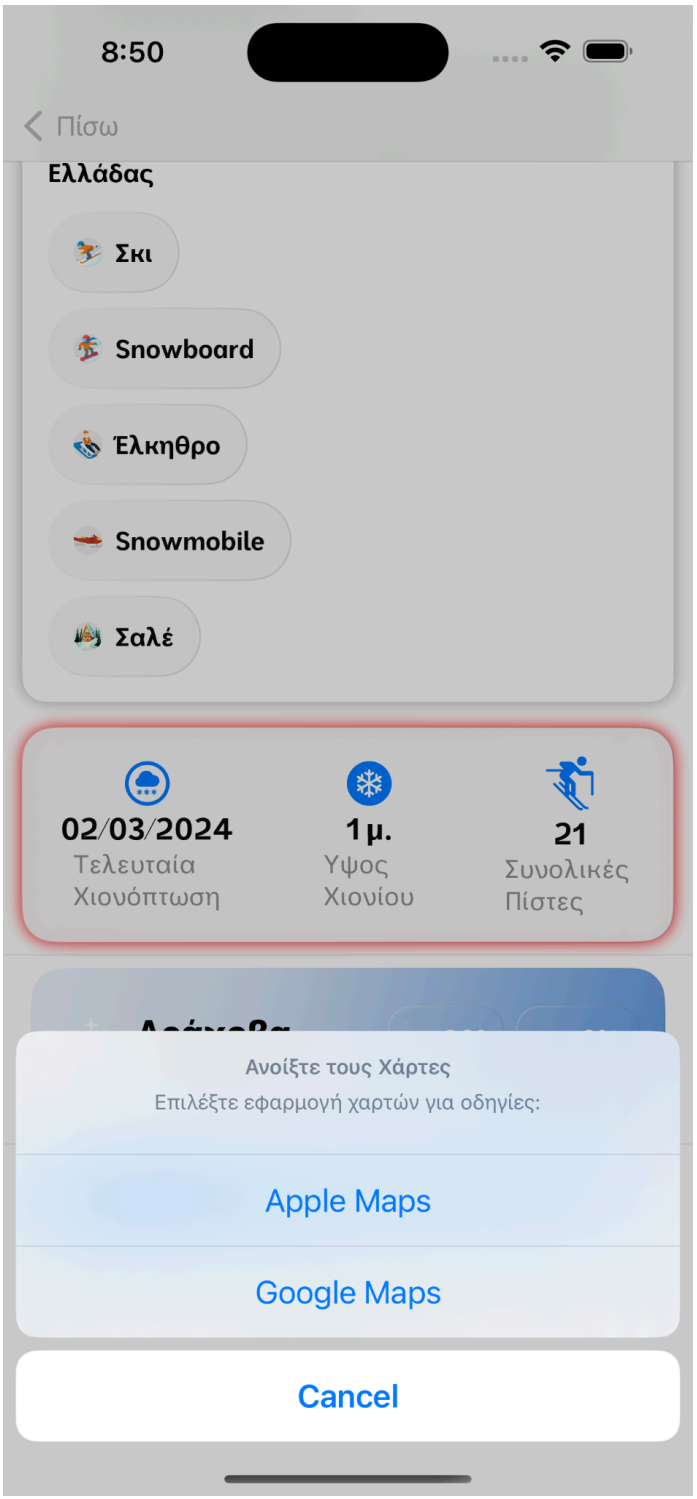
Στο παρακάτω Σχήμα 4.5 απεικονίζονται όλα όσα περιγράφονται παραπάνω για το περιεχόμενο της καρτέλας πληροφοριών.



Σχήμα 4.5 Οθόνη Πληροφοριών Χιονοδρομικού - Καρτέλα Πληροφοριών



Σχήμα 4.6 Οθόνη Πληροφοριών Χιονοδρομικού - Καρτέλα Πληροφοριών - Καιρικές Συνθήκες



Σχήμα 4.7 Οθόνη Πληροφοριών Χιονοδρομικού - Καρτέλα Πληροφοριών - Επιλογή Χάρτη

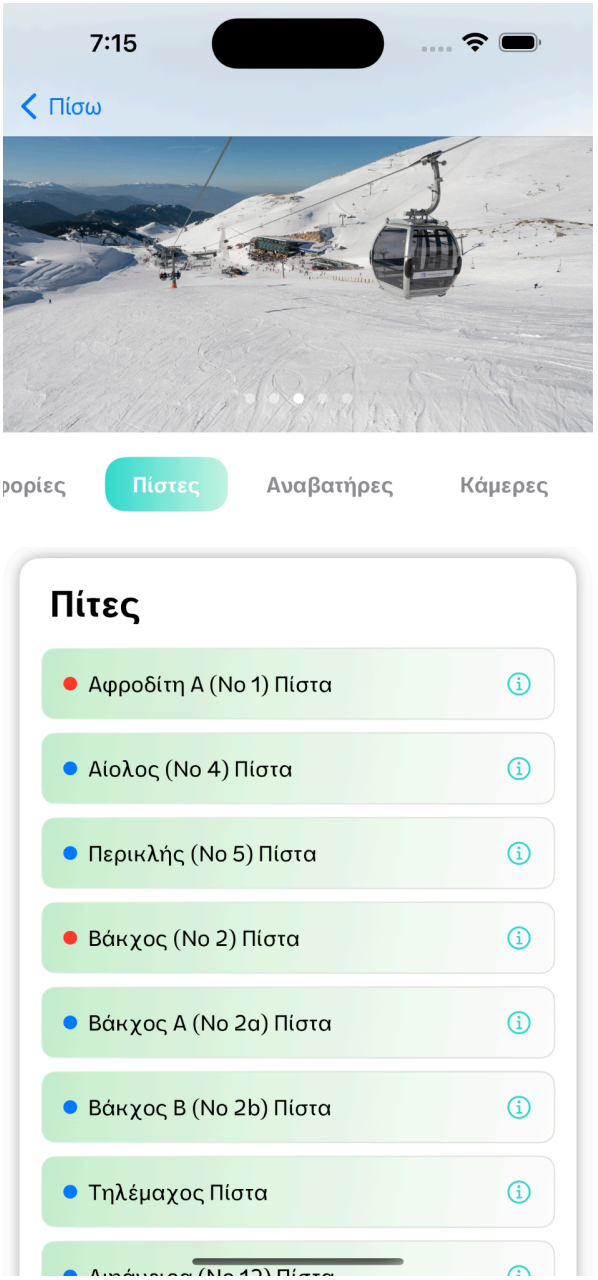
4.3.2 Καρτέλα Πιστών (Slopes Tab)

Η καρτέλα των πιστών παρέχει πληροφορίες σχετικά με τις πίστες στο χιονοδρομικό. Κάθε πίστα εμφανίζεται με το όνομα της, καθώς και το επίπεδο δυσκολίας της, που υποδεικνύεται με χρωματικούς δείκτες δίπλα από το όνομα της πίστας:

- **Πράσινος Δείκτης:** Εύκολη
- **Μπλε Δείκτης:** Μέτρια.
- **Κόκκινος Δείκτης:** Δύσκολη.
- **Μαύρος Δείκτης:** Πολύ Δύσκολη

Για κάθε πίστα παρέχονται πληροφορίες όπως ο αριθμός της (π.χ. “Αφροδίτη Α” ή “Αίολος Νο.4”). Όπως και στην καρτέλα των αναβατήρων, δίπλα σε κάθε πίστα υπάρχει το εικονίδιο πληροφοριών (i), που μπορεί να παρέχει επιπλέον πληροφορίες όπως το επίπεδο δυσκολίας ή άλλες λεπτομέρειες.

Η καρτέλα αυτή είναι σημαντική για τους χρήστες που επιθυμούν να γνωρίζουν ποιες πίστες είναι διαθέσιμες, ώστε να επιλέξουν την κατάλληλη διαδρομή ανάλογα με τις ικανότητές τους και τις προτιμήσεις τους. Στο παρακάτω Σχήμα 4.8 απεικονίζεται η καρτέλα πιστών.



Σχήμα 4.8 Οθόνη Πληροφοριών Χιονοδρομικού - Καρτέλα Πιστών

4.3.3 Καρτέλα Αναβατήρων (Lifts Tab)

Στην καρτέλα των αναβατήρων, εμφανίζονται οι πληροφορίες σχετικά με τους αναβατήρες που βρίσκονται στο χιονοδρομικό κέντρο. Κάθε αναβατήρας εμφανίζεται με το όνομα του, καθώς και την κατάστασή του (ανοιχτός ή κλειστός), που υποδεικνύεται με χρωματικούς δείκτες δίπλα από το όνομα του κάθε αναβατήρα:

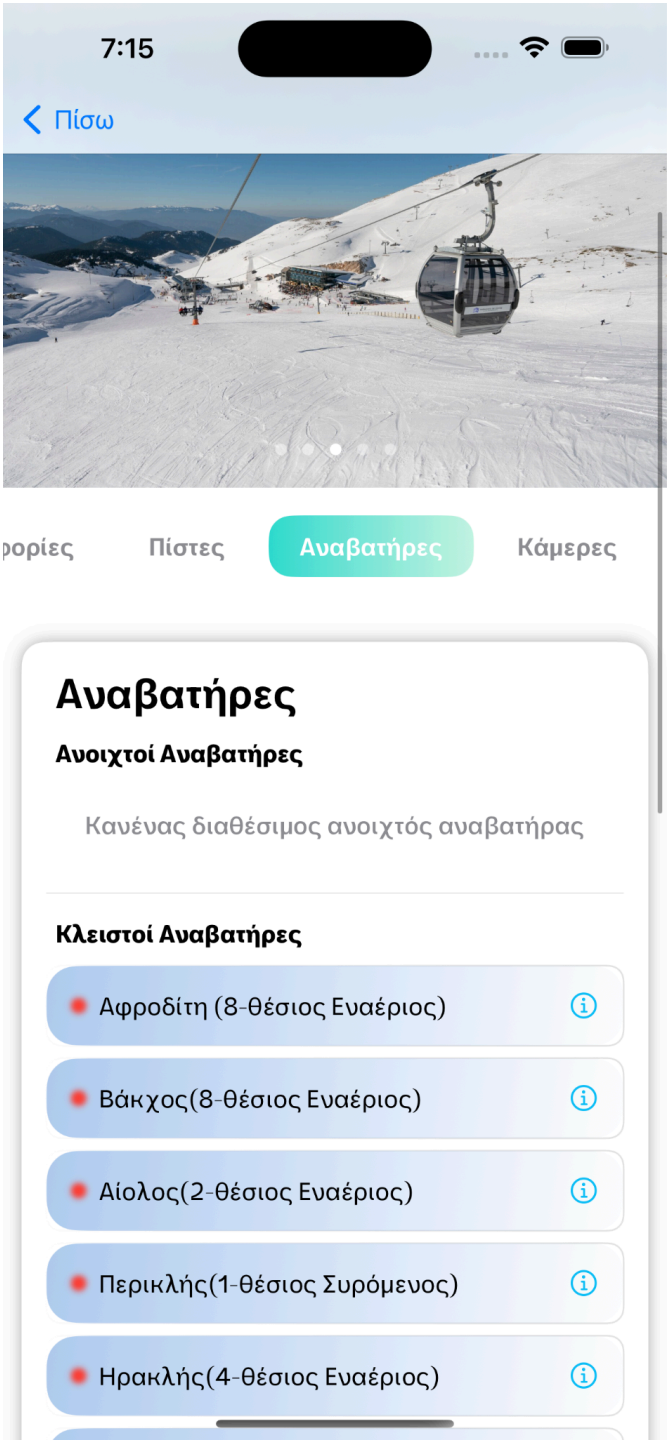
- **Πράσινος Δείκτης**: Ο αναβατήρας είναι ανοιχτός και προσβάσιμος.
- **Κόκκινος Δείκτης**: Ο αναβατήρας είναι κλειστός.

Ο διαχωρισμος των αναβατήρων γίνεται σε δύο κατηγορίες:

1. **Ανοιχτοί Αναβατήρες:** Η κατηγορία αυτή παρουσιάζει τους αναβατήρες που είναι ανοιχτοί και σε λειτουργία. Στην εικόνα δεν εμφανίζονται αναβατήρες, καθώς φαίνεται να μην υπάρχει διαθέσιμος αναβατήρας σε λειτουργία εκείνη τη στιγμή.

2. **Κλειστοί Αναβατήρες:** Στην κατηγορία αυτή παρατίθενται οι αναβατήρες που είναι κλειστοί και δεν λειτουργούν, με κάθε αναβατήρα να περιλαμβάνει πληροφορίες για τον τύπο του (π.χ. 8-θέσιος εναέριος, 1-θέσιος συρόμενος). Δίπλα σε κάθε αναβατήρα υπάρχει ένα εικονίδιο πληροφοριών (i) που, όταν πατηθεί, εμφανίζει πρόσθετες πληροφορίες σχετικά με τον αναβατήρα, όπως τεχνικά χαρακτηριστικά ή λόγους για τους οποίους είναι εκτός λειτουργίας.

Αυτή η καρτέλα δίνει στους χρήστες την δυνατότητα να γνωρίζουν ποιοι αναβατήρες είναι διαθέσιμοι κατά τη διάρκεια της ημέρας, επιτρέποντάς τους να σχεδιάσουν τη διαδρομή τους και την εμπειρία τους στο χιονοδρομικό. Στο παρακάτω Σχήμα 4.9 απεικονίζεται η καρτέλα αναβατήρων.

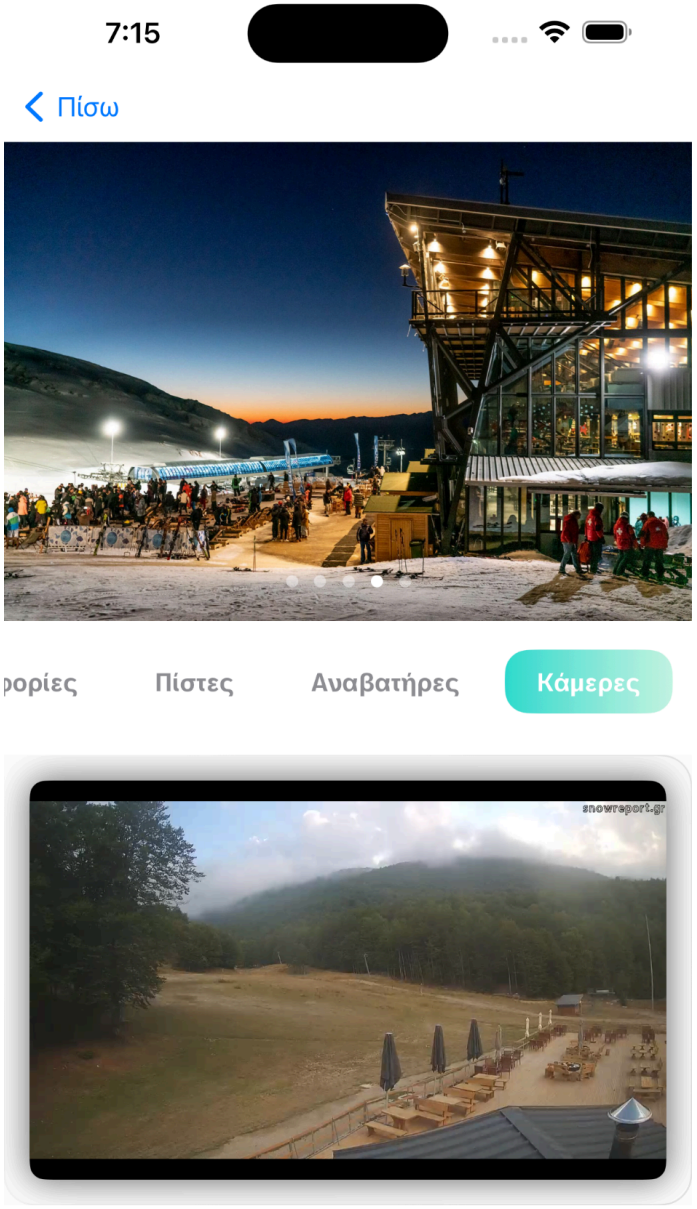


Σχήμα 4.9 Οθόνη Πληροφοριών Χιονοδρομικού - Καρτέλα Αναβατήρων

4.3.4 Καρτέλα Καμερών (Webcams tab)

Κάμερες

Η καρτέλα **Κάμερες,** όπως φαίνεται και στο παρακάτω Σχήμα 4.10 επιτρέπει στους χρήστες να παρακολουθήσουν ζωντανές εικόνες από τις κάμερες του χιονοδρομικού. Αυτή η λειτουργία προσφέρει μια ρεαλιστική απεικόνιση των συνθηκών στην τοποθεσία σε πραγματικό χρόνο, όπως η ορατότητα, η κίνηση ή η πυκνότητα του χιονιού.

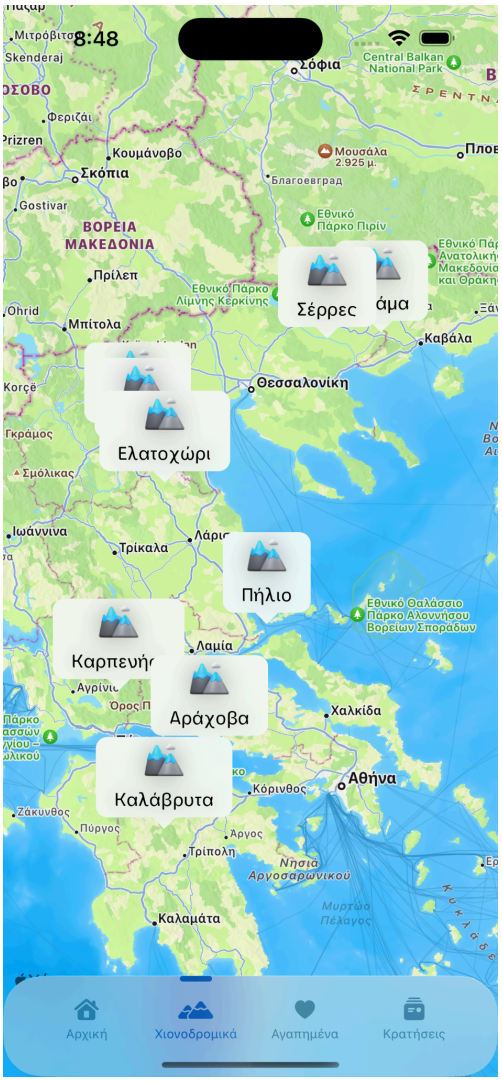


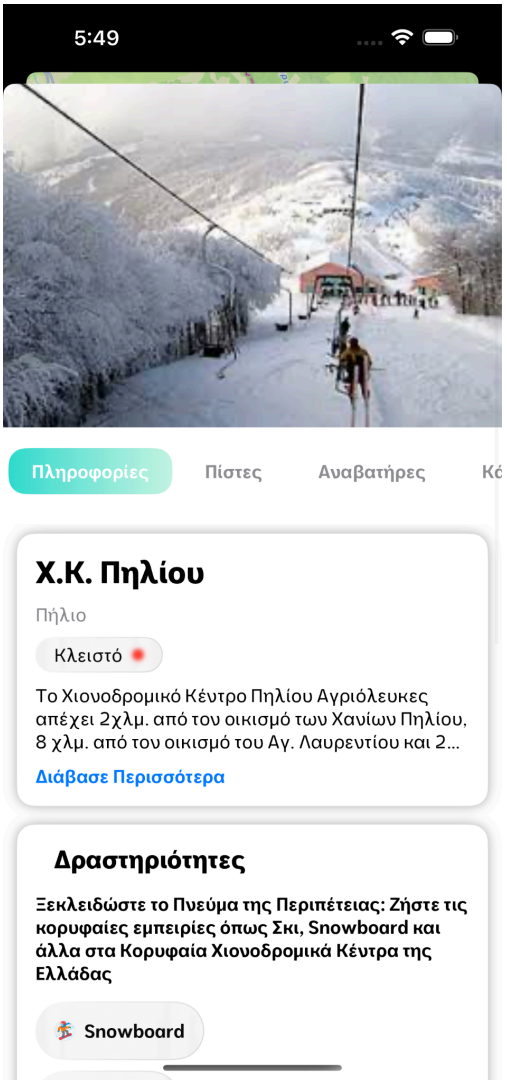
Σχήμα 4.10 Οθόνη Πληροφοριών Χιονοδρομικού - Καρτέλα Καμερών

4.4 Οθόνη Χιονοδρομικών - Resorts Screen

Η οθόνη των χιονοδρομικών παρουσιάζει έναν χάρτη της Ελλάδας που υποδεικνύει τις τοποθεσίες απο τα διάφορα χιονοδρομικά κέντρα, όπως φαίνεται στο Σχήμα 4.11. Κάθε χιονοδρομικό είναι τοποθετημένο στο χάρτη της Ελλάδας με βάση τη γεωγραφική του θέση, και οι χρήστες μπορούν να επιλέξουν να δουν πληροφορίες για το καθένα από αυτά.

Στην Σχήμα 4.11, φαίνεται η καρτέλα για το Χ.Κ. Πηλίου, όπου παρέχονται βασικές πληροφορίες για το χιονοδρομικό κέντρο, όπως περιγραφή και διαθέσιμες δραστηριότητες, όπως Snowboard. Επίσης, υπάρχουν καρτέλες επιλογών στο πάνω μέρος (Πληροφορίες, Πίστες, Αναβατήρες, Κάμερες) για πλοήγηση σε επιπλέον πληροφορίες.



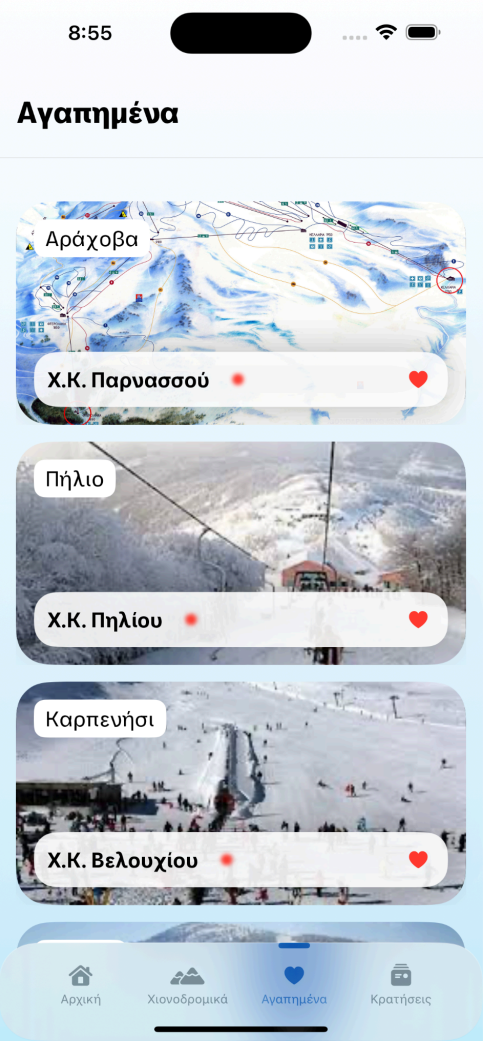


Σχήμα 4.11 Οθόνη Χιονοδρομικών - Resorts Screen

4.5 Οθόνη Αγαπημένων - Favorites Screen

Η οθόνη Αγαπημένα που παρουσιάζεται στο παρακάτω Σχήμα 4.12, απεικονίζει μια λίστα με τα χιονοδρομικά κέντρα που ο χρήστης έχει προσθέσει στα αγαπημένα του. Σε αυτή την περίπτωση, ο χρήστης έχει προσθέσει την Αράχωβα, το Πήλιο και το Καρπενήσι στη λίστα των αγαπημένων του χιονοδρομικών κέντρων. Βασική προϋπόθεση για την ύπαρξη αγαπημένων είναι ο χρήστης να έχει συνδεθεί στον προσωπικό του λογαριασμό διαφορετικά δεν έχει την δυνατότητα να προσθέσει κάποιο χιονοδρομικό κέντρο στη λίστα των αγαπημένων του. Η ευελιξία που παρέχει η λιστα με τα αγαπημένα του κάθε χρήστη είναι καίριας σημασίας, καθώς προσφέρει μια αλληλεπίδραση του χρήστη με την εφαρμογή, δινοντας του την δυνατότητα να εξατομίκευση συγκεκριμένα σημεια της εφαρμογής, όπως για παράδειγμα να προσθέσει και να αφαιρέσει αγαπημένα χιονοδρομικά κέντρα απο την λιστα αγαπημένων του.

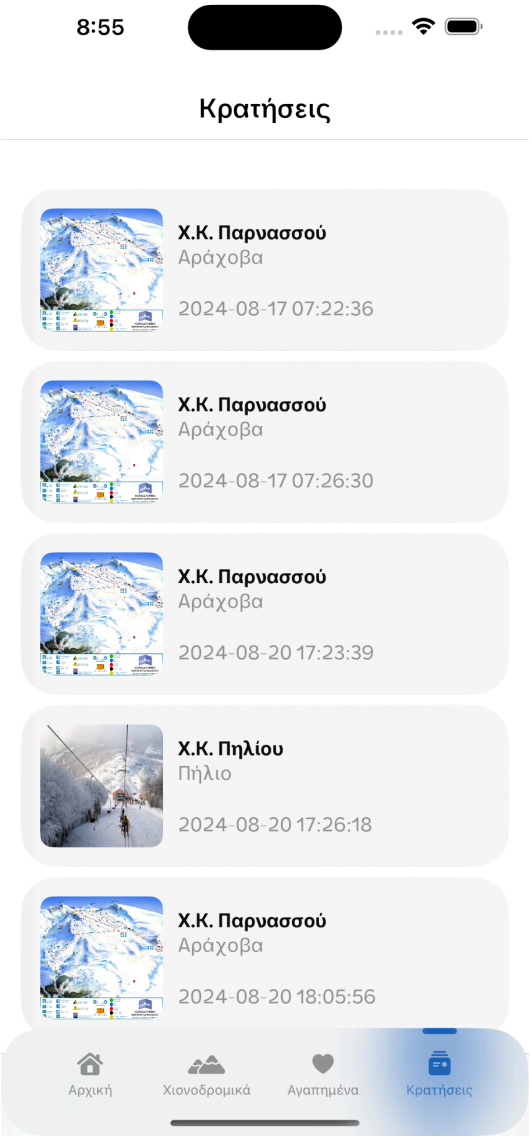
Στο συγκεκριμένο παράδειγμα με την προσθήκη αγαπημένων στην λίστα ο χρήστης μπορεί να ανατρέχει πιο αμεσα και γρηγορα στα αγαπημενα του χιονοδρομικά για να αντλήσει πληροφορίες για αυτα χωρις να χρειάζεται να ψάχνει στην αρχική οθόνη.



Σχήμα 4.12 Οθόνη Αγαπημένων - Favorites Screen

4.6 Οθόνη Κρατήσεων - Bookings Screen

Στην οθόνη Κρατήσεις, που παρουσιάζεται στο παρακάτω Σχήμα 4.13, ο χρήστης μπορεί να δει όλες τις ενεργές και προηγούμενες κρατήσεις του. Στην εικόνα, φαίνονται κρατήσεις για το Χ.Κ. Παρνασσού και το Χ.Κ. Πηλίου, με τις αντίστοιχες ημερομηνίες και ώρες των κρατήσεων να εμφανίζονται κάτω από την κάθε κράτηση. Βασική προϋπόθεση, όπως και στην οθόνη αγαπημένων είναι ο χρήστης να έχει συνδεθεί στον προσωπικό του λογαριασμό διαφορετικά δεν έχει την δυνατότητα να δει τη λίστα των κρατήσεών του. Αυτή η λειτουργία επιτρέπει στον χρήστη να παρακολουθεί τις κρατήσεις του σε διάφορα χιονοδρομικά και αν το θελήσει με την επιλογή κάποιας κράτησης να μεταφερθεί στην σελιδα πληροφοριών του ανάλογου χιονοδρομικού κέντρου.



Σχήμα 4.13 Οθόνη Κρατήσεων - Bookings Screen

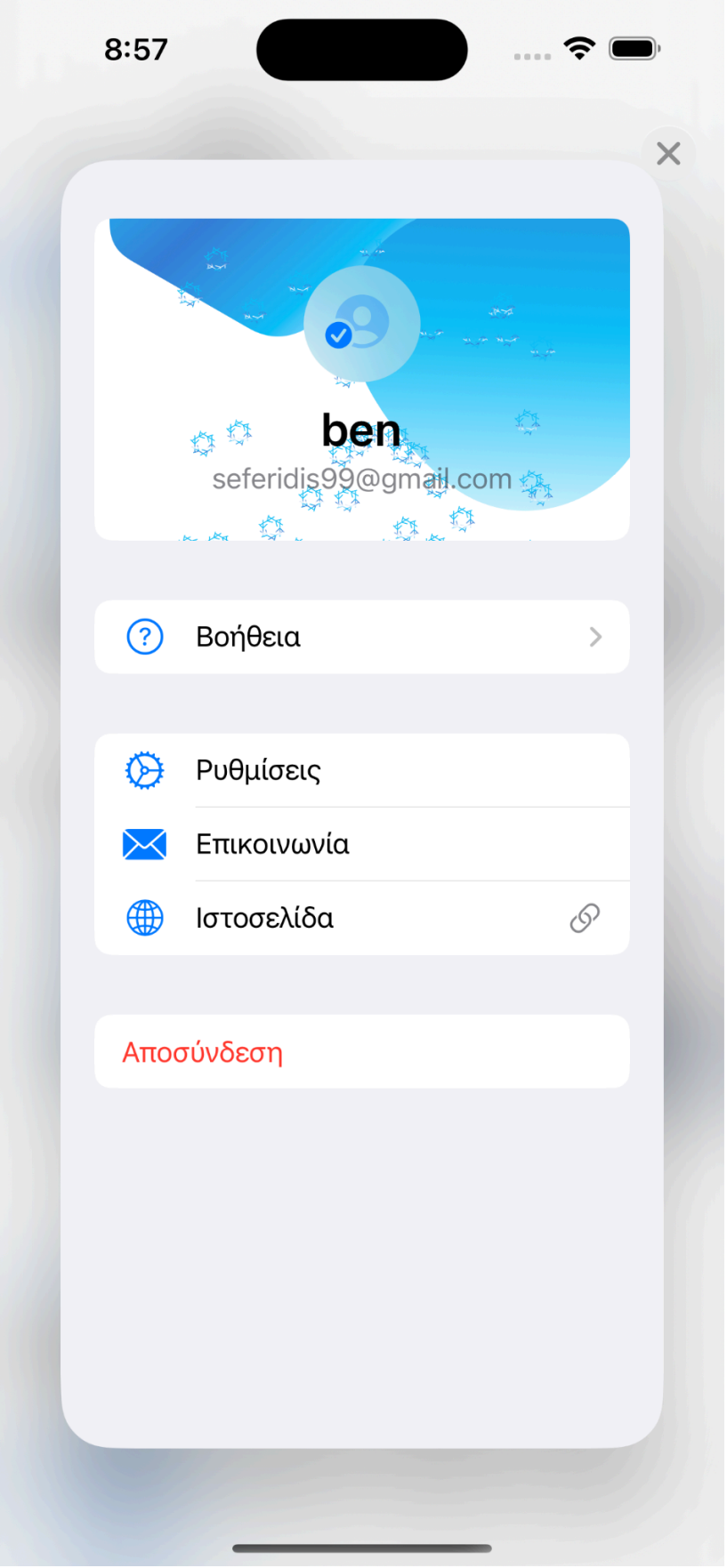
4.7 Οθόνη Λογαριασμού Χρήστη - User Account Screen

Η οθόνη λογαριασμού χρήστη στην εφαρμογή **SnowHub** παρέχει στον χρήστη τη δυνατότητα διαχείρισης και προβολής των προσωπικών πληροφοριών του, καθώς και πρόσβαση σε σημαντικές λειτουργίες σχετικές με τον λογαριασμό του. Στην κορυφή της οθόνης, εμφανίζεται το όνομα του χρήστη, η διεύθυνση email του, και το icon του προφίλ του. Η οθόνη αυτή επιτρέπει στον χρήστη να διαχειρίζεται τη σύνδεσή του, να λαμβάνει υποστήριξη και να έχει πρόσβαση σε βασικές ρυθμίσεις.

Στοιχεία της Οθόνης:

- Profile icon και Όνομα Χρήστη: Στο πάνω μέρος, εμφανίζεται το profile icon του χρήστη μαζί με το όνομά του και τη διεύθυνση email. Αυτό παρέχει μια γρήγορη επισκόπηση των βασικών πληροφοριών του λογαριασμού.
- Βοήθεια: Υπάρχει ένα κουμπί βοήθειας, το οποίο οδηγεί τον χρήστη σε μια σελίδα υποστήριξης, όπου μπορεί να λάβει πληροφορίες ή βοήθεια για τη χρήση της εφαρμογής.
- Ρυθμίσεις: Ένα κουμπί για πρόσβαση στις γενικές ρυθμίσεις της εφαρμογής, όπως αλλαγή της γλώσσας.
- Επικοινωνία: Παρέχει τη δυνατότητα στον χρήστη να επικοινωνήσει με την υποστήριξη πελατών μέσω email, για να επιλύσει προβλήματα ή να ζητήσει βοήθεια.
- Ιστοσελίδα: Παρέχεται ένας σύνδεσμος για την επίσημη ιστοσελίδα της εφαρμογής, ώστε ο χρήστης να ενημερώνεται για νέα και ανακοινώσεις ή να αποκτήσει επιπλέον πληροφορίες.
- Αποσύνδεση: Στο κάτω μέρος της οθόνης, υπάρχει ένα κουμπί Αποσύνδεσης (Log out), το οποίο επιτρέπει στον χρήστη να αποσυνδεθεί από την εφαρμογή με ασφάλεια. Αυτή η λειτουργία είναι ιδιαίτερα σημαντική για την προστασία των προσωπικών δεδομένων του χρήστη.

Η οθόνη αυτή προσφέρει άμεση πρόσβαση σε κρίσιμες λειτουργίες όπως η αποσύνδεση, οι ρυθμίσεις και η βοήθεια. Οι χρήστες μπορούν να δουν τα στοιχεία του λογαριασμού τους ή να λάβουν υποστήριξη χωρίς να περιηγηθούν σε πολύπλοκα μενού. Στο παρακάτω Σχήμα 4.14 απεικονίζεται η οθόνη λογαριασμού χρήστη.



Σχήμα 4.14 Οθόνη Λογαριασμού Χρήστη - User Account Screen

4.8 Οθόνη Ρυθμίσεων - Settings Screen

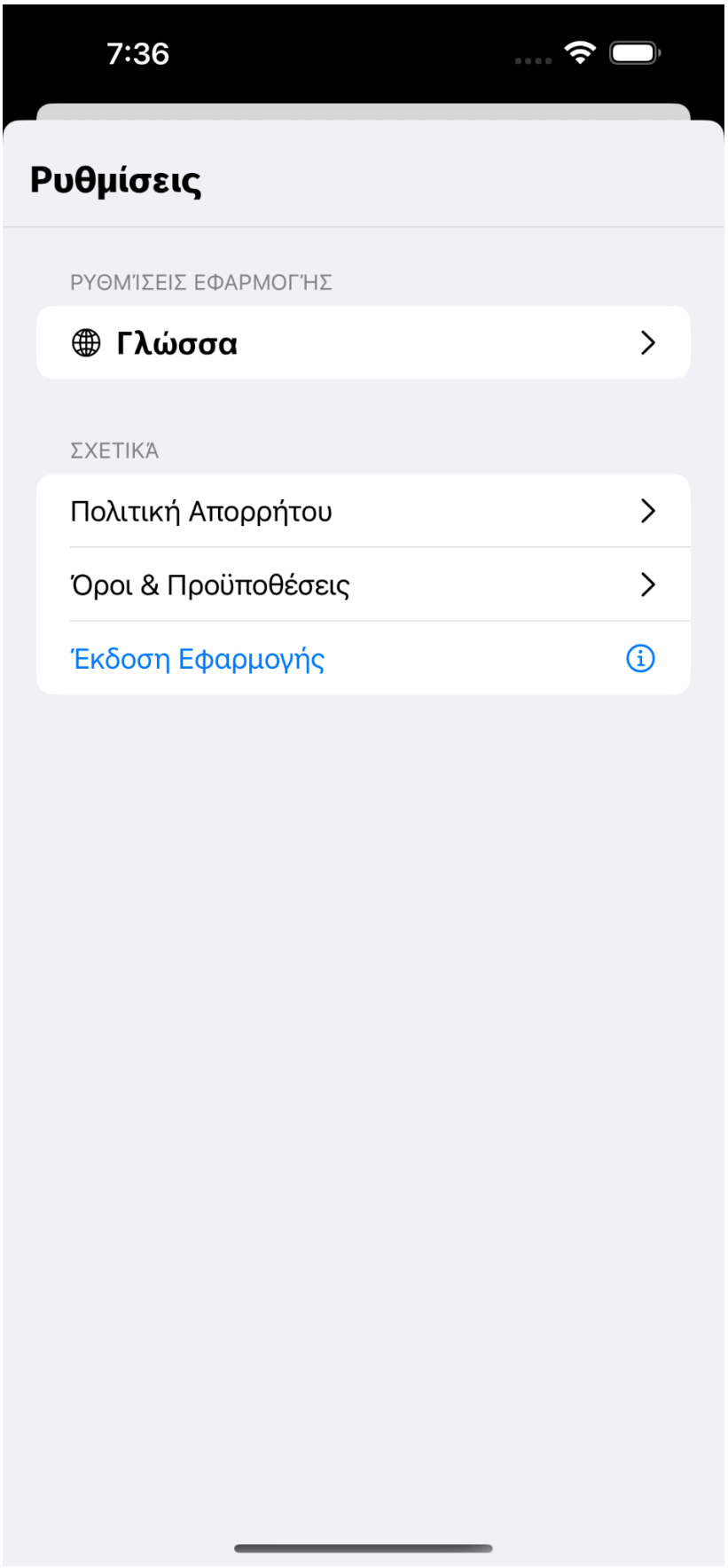
Περιγραφή:

Η οθόνη Ρυθμίσεων επιτρέπει στον χρήστη να προσαρμόσει βασικές λειτουργίες της εφαρμογής SnowHub σύμφωνα με τις προτιμήσεις του. Οι ρυθμίσεις είναι ομαδοποιημένες σε τμήματα, διευκολύνοντας την πλοήγηση και την επιλογή των επιθυμητών ρυθμίσεων. Η διαχείριση των ρυθμίσεων είναι ουσιώδης για την παροχή μιας εξατομικευμένης εμπειρίας χρήστη, καθώς προσφέρονται επιλογές σχετικά με τη γλώσσα, την ιδιωτικότητα, και τους όρους χρήσης.

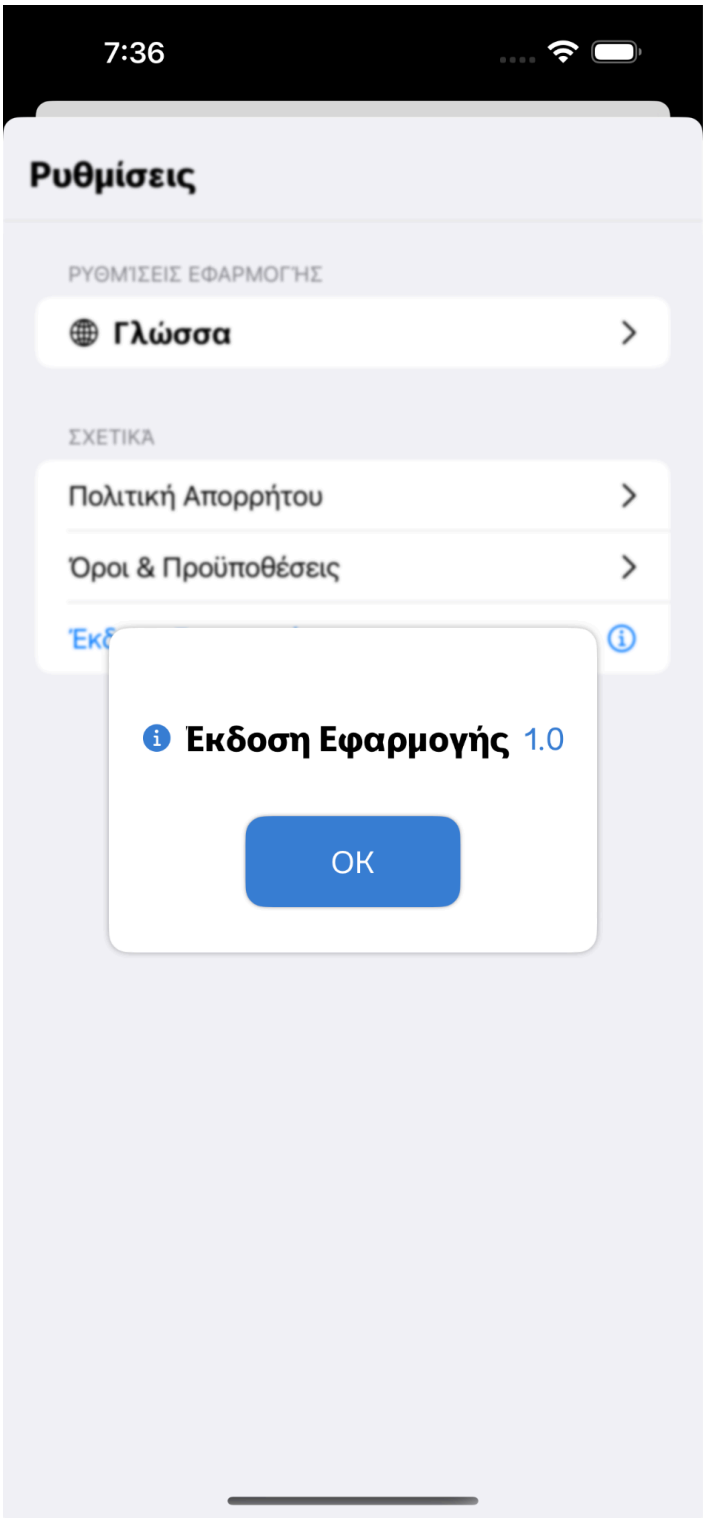
Στοιχεία της Οθόνης:

- **Language (Γλώσσα)**: Στο πάνω μέρος της οθόνης, υπάρχει μια επιλογή για την αλλαγή της γλώσσας της εφαρμογής. Ο χρήστης μπορεί να επιλέξει την προτιμώμενη γλώσσα του, κάτι που επηρεάζει άμεσα τη διεπαφή χρήστη. Συγκεκριμένα αυτή η ρύθμιση είναι απαραίτητη για την παροχή μιας πολυγλωσσικής εμπειρίας χρήστη, ειδικά για χρήστες που επιθυμούν να χρησιμοποιήσουν την εφαρμογή στη παγκόσμια γλώσσα τα Αγγλικά είτε στην βασική γλώσσα της εφαρμογής τα Ελληνικά.
- **Privacy & Policy (Πολιτική Απορρήτου)**: Η Συγκεκριμένη επιλογή παρέχει πρόσβαση στην πολιτική απορρήτου της εφαρμογής. Ο χρήστης μπορεί να ενημερωθεί για τον τρόπο με τον οποίο η εφαρμογή συλλέγει, χρησιμοποιεί, και αποθηκεύει τα προσωπικά του δεδομένα.
- **Terms & Conditions (Όροι Χρήσης)**: Παρέχει λεπτομέρειες για τους όρους και τις προϋποθέσεις χρήσης της εφαρμογής. Αυτή η ενότητα εξηγεί τους κανονισμούς που διέπουν τη χρήση της πλατφόρμας, κάτι που είναι ιδιαίτερα σημαντικό για τη διαφάνεια προς τον χρήστη.
- **Version Info (Πληροφορίες Έκδοσης)**: Παρέχει πληροφορίες για την τρέχουσα έκδοση της εφαρμογής που χρησιμοποιεί ο χρήστης.

Η οθόνη ρυθμίσεων προσφέρει στον χρήστη τον πλήρη έλεγχο της εμπειρίας του μέσα στην εφαρμογή. Οι χρήστες μπορούν να προσαρμόσουν τις επιλογές που τους αφορούν, όπως τη γλώσσα και να ενημερωθούν για πολιτική ιδιωτικότητας και τους όρους χρήσης της εφαρμογής. Στο παρακάτω Σχήμα 4.15 απεικονίζεται η οθόνη ρυθμίσεων εφαρμογής και στο Σχήμα 4.16 απεικονίζεται στην οθόνη ρυθμίσεων εφαρμογής η έκδοση της εφαρμογής.



Σχήμα 4.15 Οθόνη Ρυθμίσεων - Settings Screen



Σχήμα 4.16 Οθόνη Ρυθμίσεων - Settings Screen - Version

4.9 Οθόνη Πληρωμής - Payment Screen

Η οθόνη **Payment Screen** στην εφαρμογή **SnowHub** είναι το τελικό βήμα για την ολοκλήρωση μιας κράτησης εισιτηρίου για το χιονοδρομικό κέντρο. Πρόκειται για μια διεπαφή όπου ο χρήστης μπορεί να επιβεβαιώσει τις λεπτομέρειες της παραγγελίας του και να προχωρήσει στην πληρωμή μέσω **Apple Pay**. Παρακάτω περιγράφονται αναλυτικά τα στοιχεία και οι λειτουργίες της οθόνης:

Λεπτομέρειες Παραγγελίας

Στην κορυφή της οθόνης, ο χρήστης βλέπει το συνολικό ποσό του καλαθιού, το οποίο αφορά το κόστος του εισιτηρίου για το συγκεκριμένο χιονοδρομικό κέντρο. Στο παράδειγμα, το χιονοδρομικό που εμφανίζεται είναι το "Χ.Κ. Παρνασσού" με την τιμή του εισιτηρίου να είναι 13,00 €. Παρέχεται πλήρης αναφορά για το όνομα του χιονοδρομικού και τον αριθμό των εισιτηρίων που αγοράζονται.

Επιλογή Πληρωμής μέσω Apple Pay

Η επιλογή πληρωμής γίνεται κυρίως μέσω της πλατφόρμας **Apple Pay**, η οποία εμφανίζεται ξεκάθαρα στο κάτω μέρος της οθόνης με το κουμπί "**Πληρωμή μέσω Apple Pay**". Η χρήση του **Apple Pay** προσφέρει μια γρήγορη και ασφαλή εμπειρία πληρωμής, εκμεταλλευόμενη τις ενσωματωμένες δυνατότητες των συσκευών iOS.

Επιβεβαίωση Πληρωμής

Μόλις ο χρήστης επιλέξει την πληρωμή μέσω **Apple Pay**, εμφανίζεται ένα νέο παράθυρο που περιέχει τις πληροφορίες της πληρωμής:

- Η προσυμπληρωμένη κάρτα πληρωμής (π.χ. "Προσομοιωμένη κάρτα - 1234").
- Οι λεπτομέρειες παράδοσης, όπου φαίνεται η διεύθυνση αποστολής ή παραλαβής του εισιτηρίου (προσομοιωμένα δεδομένα για δοκιμή).
- Πληροφορίες για την εκτιμώμενη άφιξη της παραγγελίας (στη συγκεκριμένη περίπτωση προσομοιώνεται η ημερομηνία 12-17 Σεπτεμβρίου).

Συνολικό Κόστος

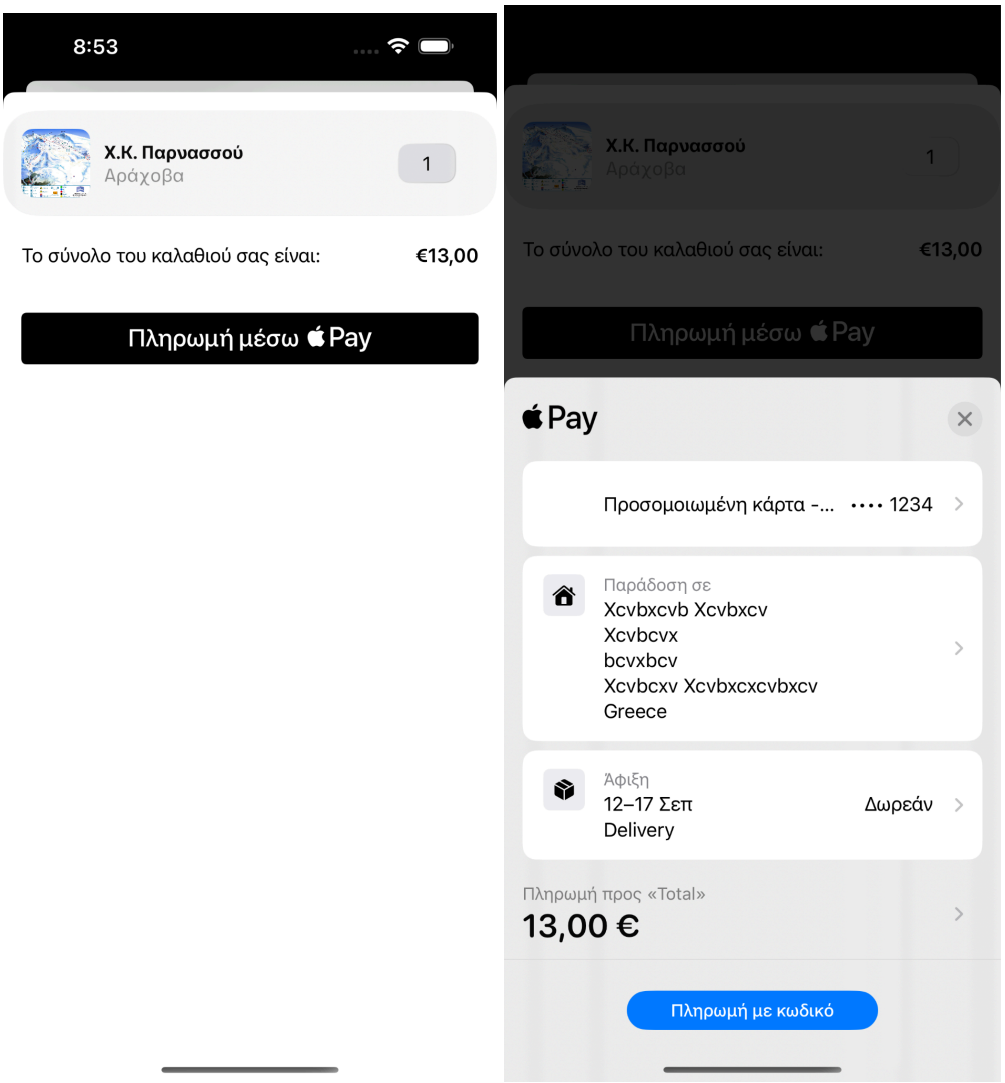
Το συνολικό κόστος της συναλλαγής αναγράφεται στο κάτω μέρος της οθόνης με την τιμή των 13,00 €, η οποία αναφέρεται στην αγορά του εισιτηρίου για το χιονοδρομικό.

Επιλογή Πληρωμής

Εάν ο χρήστης αποφασίσει να ολοκληρώσει την πληρωμή, μπορεί να το κάνει είτε επιλέγοντας το κουμπί "**Πληρωμή με κωδικό**" είτε επιβεβαιώνοντας με το **Face ID**, το **Touch ID** ή τον **κωδικό πρόσβασης** της συσκευής του, ανάλογα με τις ρυθμίσεις ασφαλείας της συσκευής.

Η οθόνη αυτή εξυπηρετεί την εύκολη και ασφαλή ολοκλήρωση της διαδικασίας κράτησης εισιτηρίων για τα χιονοδρομικά κέντρα, προσφέροντας ταυτόχρονα μια μοντέρνα και απλή διεπαφή πληρωμής μέσω της δημοφιλούς πλατφόρμας **Apple Pay.**

Στο παρακάτω Σχήμα 4.17 απεικονίζεται η οθόνη Πληρωμής - Payment Screen



Σχήμα 4.17 Οθόνη Πληρωμής - Payment Screen

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Εξέλιξη

5.1 Συμπεράσματα

Η δημιουργία της προτεινόμενης iOS εφαρμογής για τα χιονοδρομικά κέντρα της Ελλάδας αποτελεί μια ουσιαστική καινοτομία που απαντά σε μια σημαντική ανάγκη της αγοράς. Σε μια εποχή όπου η ψηφιακή τεχνολογία έχει διεισδύσει σε κάθε πτυχή της ζωής μας, η απουσία μιας τέτοιας εφαρμογής στο ελληνικό app marketplace είναι αισθητή, ιδιαίτερα όταν αντίστοιχες λύσεις είναι ήδη διαδεδομένες στο εξωτερικό.

Η εφαρμογή αυτή προσφέρει μια ολοκληρωμένη πλατφόρμα πληροφόρησης και αλληλεπίδρασης για τους λάτρεις των χειμερινών σπορ. Με την παροχή άμεσης και αξιόπιστης ενημέρωσης σχετικά με τις καιρικές συνθήκες, την κατάσταση των πιστών, τις τιμές και τις διαθέσιμες υπηρεσίες, διευκολύνει τον προγραμματισμό και την οργάνωση των επισκέψεων στα χιονοδρομικά κέντρα.

Επιπλέον, η δυνατότητα κρατήσεων και πλοήγησης μέσα από την εφαρμογή βελτιώνει σημαντικά την εμπειρία του χρήστη.

Η σημασία της εφαρμογής γίνεται ακόμη μεγαλύτερη αν λάβουμε υπόψη την αυξανόμενη ενασχόληση της Gen Z με τα χειμερινά σπορ μετά την πανδημία του COVID-19. Η γενιά αυτή, όντας τεχνολογικά εγγράμματη και αναζητώντας άμεση και εύκολη πρόσβαση σε πληροφορίες, θα βρει στην εφαρμογή ένα εργαλείο που ανταποκρίνεται πλήρως στις ανάγκες και τις προσδοκίες της.

Πέρα από τα οφέλη για τους χρήστες, η εφαρμογή μπορεί να λειτουργήσει ως μοχλός ανάπτυξης για τον χειμερινό τουρισμό στην Ελλάδα. Με την ενίσχυση της προβολής των χιονοδρομικών κέντρων και την αύξηση της προσβασιμότητας των πληροφοριών, αναμένεται να αυξηθεί η επισκεψιμότητα και η οικονομική δραστηριότητα στις ορεινές περιοχές. Η συγκέντρωση των πληροφοριών σε μια ενιαία πλατφόρμα μειώνει την κατακερματισμένη ενημέρωση που αποτελεί σήμερα εμπόδιο για πολλούς ενδιαφερόμενους.

Η εφαρμογή SnowHub, όπως είναι σήμερα, αποτελεί μια ολοκληρωμένη λύση για την παρακολούθηση και τη βελτίωση της εμπειρίας των χρηστών στα χιονοδρομικά κέντρα. Με τη χρήση της εφαρμογής, οι χρήστες έχουν τη δυνατότητα να παρακολουθούν τις ενημερωμένες πληροφορίες των χιονοδρομικών, όπως την δυνατότητα να κάνουν σκι και snowboard, καθώς και να λαμβάνουν εξατομικευμένο περιεχόμενο που ενισχύει την εμπειρία τους στο βουνό. Χάρη στη χρήση του σχεδιαστικού προτύπου Clean Architecture και της τεχνολογίας SwiftUI, η εφαρμογή SnowHub είναι χτισμένη με τρόπο που επιτρέπει την εύκολη επέκταση και βελτίωση, διασφαλίζοντας ότι μπορεί να υποστηρίξει μελλοντικές λειτουργίες και να παραμείνει ενημερωμένη με τις τελευταίες τεχνολογικές εξελίξεις.

Η χρήση του SwiftUI framework δίνει την δυνατότητα να υλοποιηθούν μοντέρνοι σχεδιασμοί στην εφαρμογή με βάση τις νέες τάσεις του mobile design προσφέροντας στους χρήστες μια μοναδική εμπειρία πλοήγησης.

Η δημιουργία του SnowHub ήταν μια ιδιαίτερα επιτυχημένη και διαφωτιστική διαδικασία. Μέσα από την ανάπτυξη της εφαρμογής, κατάφερα να αποκτήσω μια βαθύτερη κατανόηση των προκλήσεων που προκύπτουν κατά τη δημιουργία ενός τέτοιου έργου, ιδιαίτερα σε τομείς όπως ο σχεδιασμός, ο καθορισμός προδιαγραφών, η διαχείριση χρόνου και η εκπλήρωση προθεσμιών. Αυτά τα μαθήματα θα είναι πολύτιμα τόσο για την περαιτέρω εξέλιξη του SnowHub όσο και για την ανάπτυξη μελλοντικών έργων λογισμικού.

Συνοψίζοντας, η ανάπτυξη αυτής της εφαρμογής προσφέρει μια σύγχρονη και πρακτική λύση σε ένα υπαρκτό πρόβλημα, ενισχύοντας την ανταγωνιστικότητα του ελληνικού χειμερινού τουρισμού. Αποτελεί ένα βήμα προς τον ψηφιακό μετασχηματισμό του κλάδου, προωθώντας την καινοτομία και συμβάλλοντας στην οικονομική και κοινωνική ανάπτυξη των περιοχών που φιλοξενούν χιονοδρομικά κέντρα. Είναι μια επένδυση που υπόσχεται να αποδώσει σημαντικά οφέλη τόσο στους χρήστες όσο και στην ευρύτερη τουριστική βιομηχανία της χώρας.

5.2 Μελλοντική Εξέλιξη και Βελτίωση

Παρόλο που το SnowHub είναι ήδη μια πλήρως λειτουργική εφαρμογή, υπάρχουν πολλές δυνατότητες για μελλοντικές βελτιώσεις και προσθήκες που θα εμπλουτίσουν την εμπειρία των χρηστών. Ένας σημαντικός στόχος είναι η ανάπτυξη μιας εφαρμογής για το Apple Watch, η οποία θα παρακολουθεί τις χιονοδρομικές δραστηριότητες, όπως το σκι και το snowboard, σε πραγματικό χρόνο και θα συνεργάζεται άψογα με την κύρια εφαρμογή SnowHub στο iOS. Αυτό θα επιτρέψει στους χρήστες να έχουν άμεση πρόσβαση στα δεδομένα τους και να παρακολουθούν την απόδοσή τους απευθείας από το ρολόι τους.

Η μεγαλύτερη μελλοντική βελτίωση που σχεδιάζεται είναι η δυνατότητα χρήσης του SnowHub για την αγορά και χρήση εισιτηρίων σε lifts στα χιονοδρομικά κέντρα μέσω NFC. Αυτή η λειτουργία θα ενσωματωθεί με ένα νέο σύστημα κρατήσεων και πληρωμών που θα αναπτυχθεί σε συνεργασία με τα χιονοδρομικά κέντρα, προσφέροντας στους χρήστες μια ολοκληρωμένη εμπειρία από την κράτηση έως και την πρόσβαση στις εγκαταστάσεις. Παράλληλα θα υλοποιηθεί λειτουργία ανίχνευσης κινητικότητας(traffic)στα lifts ώστε να μπορεί ο κάθε χρήστης να ενημερωνεται για τον χρόνο αναμονης που έχει κάθε αναβατήρας και έτσι να γνωρίζει ποιους να αποφύγει ωστε να έχει καλυτερη εμπειρια στο βουνό με λιγότερες αναμονές και περισσότερες δραστηριότητες.

Με την προσθήκη αυτών των λειτουργιών, το SnowHub θα επεκτείνει σημαντικά τις δυνατότητές του και θα ενισχύσει την αξία που προσφέρει στους χρήστες. Οι νέες δυνατότητες θα συμβάλλουν στην περαιτέρω ανάπτυξη της κοινότητας των χιονοδρόμων, παρέχοντας παράλληλα μια σύγχρονη, άνετη και ασφαλή εμπειρία χρήσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Apple Human Interface Guidelines – SF Symbol Animations [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/sf-symbols#Animations

[2] Apple Developer Documentation – Swift [Online]. Available: https://developer.apple.com/swift/

[3] Apple Developer Documentation – Xcode [Online]. Available: https://developer.apple.com/xcode/

[4] Apple Developer Documentation – SwiftUI [Online]. Available: https://developer.apple.com/documentation/swiftui/

[5] Imperative Vs Declarative Programming in Swift [Online]. Available: https://medium.com/macoclock/imperative-vs-declarative-programming-swift-fa538e01a7ba

[6] Implementing Dark Mode on iOS [Online]. Available: https://developer.apple.com/videos/play/wwdc2019/214

[7] Git [Online]. Available: https://git-scm.com

[8] Git Documentation – Reference Manual [Online]. Available: https://git-scm.com/docs

[8] GitHub [Online]. Available: https://github.com

[10] Robert Martin, *Clean Architecture - A Craftsman's Guide to Software Structure and Design, PEARSON, 2017* [Online]. Available: https://agorism.dev/book/software-architecture/%28Robert%20C.%20Martin%20Series%29%20Robert%20C.%20Martin%20-%20Clean%20Architecture_%20A%20Craftsman%E2%80%99s%20Guide%20to%20Software%20Structure%20and%20Design-Prentice%20Hall%20%282017%29.pdf

[11] Swift Documentation – Enumerations [Online]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/enumerations/

[12] Swift Documentation – Protocols [Online]. Available: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/protocols/

[13] GitHub – Alamofire [Online]. Available: https://github.com/Alamofire/Alamofire

[14] GitHub – Kingfisher [Online]. Available: https://github.com/onevcat/Kingfisher

[15] GitHub – IOSSecuritySuite [Online]. Available: https://github.com/securing/IOSSecuritySuite

[16] GitHub – Firebase iOS sdk [Online]. Available: https://github.com/firebase/firebase-ios-sdk

[17] GitHub – Swift crypto [Online]. Available: https://github.com/apple/swift-crypto

[18] Medium Article - S.O.L.I.D. PRINCIPLES [Online]. Available: https://medium.com/@nishant.kumbhare4/solid-principles-in-swift-73b505d3c63f

[19] Medium Article - Clean Architecture [Online]. Available: https://tech.olx.com/clean-architecture-and-mvvm-on-ios-c9d167d9f5b3

[20] Article Avander Lee - Async Await [Online]. Available: https://www.avanderlee.com/swift/async-await/