ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σχεδιασμός & Ανάπτυξη Web Εφαρμογής για χιονοδρομικά στην Ελλάδα»

Του φοιτητή
Μεϊμάρογλου Ανδρέα
Αρ. Μητρώου: 185225

Επιβλέπων
Χαράλαμπος Μπράτσας
Επίκουρος καθηγητής

Σεπτέμβριος 2024

Τίτλος Δ.Ε. **Σχεδιασμός & Ανάπτυξη Web Εφαρμογής για χιονοδρομικά στην Ελλάδα**
Κωδικός Δ.Ε. **24111**
Φοιτητής: **Μεϊμάρογλου Ανδρέας**
Εισηγητής: **Χαράλαμπος Μπράτσας**
Ημερομηνία ανάληψης Δ.Ε. **25-01-2024**
Ημερομηνία περάτωσης Δ.Ε. **10-09-2024**

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως ς 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.

«Στην οικογένειά μου και σε όλους μου τους φίλους»

Πρόλογος

Η επιλογή της παρούσας πτυχιακής εργασίας ήταν αποτέλεσμα της ακαδημαϊκής αλλά και της επαγγελματικής μου πορείας στην ανάπτυξη web εφαρμογών και του προσωπικού μου ενδιαφέροντος για τα χιονοδρομικά κέντρα στην Ελλάδα. Έχοντας ασχοληθεί εκτενώς με τον προγραμματισμό και ειδικότερα με την Angular, είδα την ανάπτυξη μιας εφαρμογής για τα χιονοδρομικά κέντρα ως μια ευκαιρία να συνδυάσω τις γνώσεις μου στην τεχνολογία με την αγάπη μου για τον χειμερινό αθλητισμό. Η ιδέα για την εφαρμογή προέκυψε από την παρατήρηση της έλλειψης ψηφιακών εργαλείων που παρέχουν ολοκληρωμένες πληροφορίες και υπηρεσίες για τους λάτρεις των χειμερινών σπορ στην Ελλάδα. Στόχος μου με αυτήν τη πτυχιακή εργασία είναι να εξελιχθώ περαιτέρω στον τομέα της ανάπτυξης web εφαρμογών και να συμβάλω στη βελτίωση της εμπειρίας των χρηστών που επιθυμούν να εξερευνήσουν τα χιονοδρομικά κέντρα της χώρας μας. Μέσω αυτής της εργασίας, επιδιώκω να αναδείξω τη σημασία της τεχνολογίας στον τουρισμό, ενώ παράλληλα να εξελίξω τις δεξιότητές μου στον προγραμματισμό.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στον σχεδιασμό και την ανάπτυξη μιας διαδικτυακής εφαρμογής για τα χιονοδρομικά κέντρα στην Ελλάδα. Στόχος της εφαρμογής είναι να παρέχει στους χρήστες εύκολη και γρήγορη πρόσβαση σε πληροφορίες σχετικά με τα χιονοδρομικά κέντρα, όπως τρέχουσες καιρικές συνθήκες, κατάσταση χιονόπτωσης, διαθέσιμες πίστες, τιμές εισιτηρίων, καθώς και δυνατότητα κράτησης αυτών. Η εφαρμογή στοχεύει στη βελτίωση της εμπειρίας του χρήστη, κάνοντας την αναζήτηση πληροφοριών και τον προγραμματισμό των εξορμήσεων πιο άμεσο και αποτελεσματικό. Για την υλοποίηση της εφαρμογής, έγινε χρήση σύγχρονων εργαλείων ανάπτυξης και βέλτιστων πρακτικών προγραμματισμού, εστιάζοντας στη λειτουργικότητα, την ασφάλεια και την ευχρηστία. Η Angular επιλέχθηκε ως το κύριο framework λόγω της ευελιξίας της, της δυνατότητας δημιουργίας διαδραστικών διεπαφών χρήστη και της ισχυρής υποστήριξης από την κοινότητα προγραμματιστών. Τα αποτελέσματα της εργασίας περιλαμβάνουν μια πλήρως λειτουργική και διαδραστική web εφαρμογή, η οποία έχει δοκιμαστεί σε πραγματικές συνθήκες και προσφέρει μια ολοκληρωμένη εμπειρία στους χρήστες της. Η εφαρμογή καταφέρνει να καλύψει τις ανάγκες του κοινού, προσφέροντας εύκολη πρόσβαση σε όλες τις απαραίτητες πληροφορίες για τα χιονοδρομικά κέντρα της Ελλάδας, βελτιώνοντας παράλληλα την αποτελεσματικότητα της πλοήγησης και της χρήσης υπηρεσιών. Συνολικά, η εργασία αποδεικνύει τη σημασία της τεχνολογίας στη βελτίωση του τουριστικού προϊόντος της χώρας και προτείνει μελλοντικές επεκτάσεις για την περαιτέρω βελτίωση και εμπλουτισμό της εφαρμογής με νέες λειτουργίες και δυνατότητες.

«Design & Development of a Web Application

for Ski Resorts in Greece»

«Andreas Meimaroglou»

Abstract

This thesis focuses on designing and developing an online application for ski resorts in Greece. The application aims to provide users with easy and quick access to information about ski resorts, such as current weather conditions, snow conditions, available slopes, and ticket prices, and the ability to reserve them. The application aims to improve the user experience, making searching for information and planning trips more direct and efficient. Modern development tools and best programming practices were used to implement the application, focusing on functionality, security, and usability. Angular was chosen as the main framework because of its flexibility, ability to create interactive user interfaces, and strong support from the developer community. The results of the work include a fully functional and interactive web application, which has been tested in real conditions and offers a comprehensive experience to its users. The application manages to meet the needs of the public, offering easy access to all the necessary information about the ski resorts of Greece, while improving the efficiency of navigation and the use of services. Overall, the paper proves the importance of technology in improving the tourism product of the country and suggests future extensions to further improve and enrich the application with new functions and features.

Ευχαριστίες

Ξεκινώντας θα ηθελα να ευχαριστήσω την οικογένεια μου, η οποία με στήριξε από την αρχή της εκπόνησης της πτυχιακής μου εργασίας έως και το τέλος αυτής. Η κατανόηση, η στήριξη και η υπομονή που μου έδωσαν αποτέλεσαν σημαντικά στοιχεία για την επιτυχία ολοκλήρωση της εργασίας μου.

Στην συνέχεια θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Χαράλαμπο Μπράτσα για την εμπιστοσύνη του σε εμένα και την καθοδήγησή του σε όλη την ανάπτυξη της εργασίας, καθιστώντας ομαλή όλη την διαδικασία από τις χρήσιμες συμβουλές του για την διαχείριση του χρόνου έως και τον ξεκάθαρο προσδιορισμό των στόχων της εργασίας αυτής.

Επίσης θα ήθελα να ευχαριστήσω τους φίλους αλλά και συμφοιτητές, Βενιαμίν Σεφερίδη και Ξενοφών Νικόλαο Πάντσο, οι οποίοι δημιούργησαν αντίστοιχη πτυχιακή εργασία σε iOS και Backend αντίστοιχα, δίνοντας έτσι την δυνατότητα να δημιουργήσουμε μια ολοκληρωμένη εφαρμογή.

Τέλος, η συμπαράσταση όλων των φίλων μου και συναδέλφων, αποτέλεσε σημαντικό παράγοντα για την υλοποίηση αυτής της εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

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

Κεφάλαιο 1ο: Εισαγωγή

1.1 Ιστορία των Web Εφαρμογών

- **Αρχές του Παγκόσμιου Ιστού και Στατικές Ιστοσελίδες (1990s)**: Η ιστορία των web εφαρμογών ξεκινά από τις αρχές του παγκόσμιου ιστού τη δεκαετία του 1990, όταν οι ιστοσελίδες ήταν απλές στατικές σελίδες HTML που προσέφεραν βασικές πληροφορίες στους χρήστες. Με την εξέλιξη των τεχνολογιών του Διαδικτύου και την αύξηση των αναγκών των χρηστών, οι στατικές ιστοσελίδες δεν ήταν πλέον αρκετές. Έτσι, άρχισε να εμφανίζεται η ανάγκη για πιο δυναμικές και διαδραστικές εφαρμογές που θα μπορούσαν να προσφέρουν περισσότερες λειτουργίες και εξατομικευμένες εμπειρίες.
- **Διαδραστικότητα και JavaScript (τέλη 1990s - αρχές 2000s)**: Στα τέλη της δεκαετίας του 1990 και στις αρχές της δεκαετίας του 2000, με την ανάπτυξη της γλώσσας προγραμματισμού JavaScript και την ενσωμάτωσή της στους browsers, οι web εφαρμογές έγιναν πιο διαδραστικές. Τεχνολογίες όπως τα cookies και τα session management επέτρεψαν την αποθήκευση δεδομένων και την εξατομίκευση του περιεχομένου, ενώ η χρήση των server-side γλωσσών προγραμματισμού, όπως η PHP, η ASP και η Java, επέτρεψε την ανάπτυξη πιο σύνθετων δυναμικών εφαρμογών.
- **Εισαγωγή του AJAX και Εξέλιξη των Web Εφαρμογών (μέσα 2000s)**: Η σημαντικότερη εξέλιξη ήρθε με την εισαγωγή του AJAX (Asynchronous JavaScript and XML) στα μέσα της δεκαετίας του 2000, το οποίο επέτρεψε την αλληλεπίδραση με τον server χωρίς την ανάγκη ανανέωσης της σελίδας. Αυτό άνοιξε το δρόμο για την ανάπτυξη πιο εξελιγμένων web εφαρμογών όπως τα πρώτα social media, τα ηλεκτρονικά καταστήματα και οι υπηρεσίες cloud.
- **Front-End Frameworks και Μοντέρνες Web Εφαρμογές (2010s και μετέπειτα)**: Με την εισαγωγή των front-end frameworks, όπως η Angular, η React και η Vue.js, οι web εφαρμογές έγιναν ακόμη πιο ισχυρές και μοντέρνες. Οι εφαρμογές αυτές υιοθέτησαν σχεδιαστικές αρχές όπως το Model-View-Controller (MVC) και το Model-View-ViewModel (MVVM), που επέτρεψαν τη διαχείριση της πολυπλοκότητας και την ανάπτυξη κώδικα που είναι πιο εύκολα διαχειρίσιμος και επεκτάσιμος.
- **Σημερινή Κατάσταση και Μελλοντικές Προοπτικές**: Σήμερα, οι web εφαρμογές αποτελούν κεντρικό μέρος της καθημερινής μας ζωής, καλύπτοντας ένα ευρύ φάσμα αναγκών, από την εργασία και την επικοινωνία μέχρι την ψυχαγωγία και το ηλεκτρονικό εμπόριο. Η συνεχής εξέλιξη των τεχνολογιών και των εργαλείων ανάπτυξης web εφαρμογών υπόσχεται να κάνει αυτές τις εφαρμογές ακόμα πιο ισχυρές,

1.2 Χειμερινός Τουρισμός την περίοδο του COVID-19

Η πανδημία του COVID-19 είχε σημαντική επίδραση σε πολλούς τομείς της καθημερινής ζωής, με τον χειμερινό τουρισμό να μην αποτελεί εξαίρεση. Τα χιονοδρομικά κέντρα, τα οποία αποτελούν αγαπημένα σημεία για τους λάτρεις των χειμερινών σπορ, βρέθηκαν αντιμέτωπα με σοβαρές προκλήσεις. Οι αυστηρές οδηγίες κοινωνικής αποστασιοποίησης, τα περιοριστικά μέτρα μετακίνησης, και οι ανησυχίες για την εξάπλωση του ιού οδήγησαν σε δραματική μείωση της επισκεψιμότητας.

Η αναστολή δραστηριοτήτων και η μειωμένη ζήτηση είχαν ως αποτέλεσμα την ακύρωση πολλών επισκέψεων και εκδηλώσεων, επηρεάζοντας αρνητικά τα χιονοδρομικά κέντρα και την τοπική οικονομία που εξαρτάται από τον τουρισμό. Οι λάτρεις των σπορ αντιμετώπισαν την αδυναμία να απολαύσουν τις αγαπημένες τους δραστηριότητες, γεγονός που ανέδειξε την ανάγκη για νέα εργαλεία επικοινωνίας και ενημέρωσης.

Για την κάλυψη αυτής της ανάγκης, δημιουργήθηκε η εφαρμογή "SnowHub". Στόχος της είναι να παρέχει στους επισκέπτες των χιονοδρομικών κέντρων μια ολοκληρωμένη και ενημερωμένη πλατφόρμα, επιτρέποντάς τους να προγραμματίζουν τις επισκέψεις τους με ασφάλεια. Η εφαρμογή προσφέρει άμεση πρόσβαση σε κρίσιμες πληροφορίες όπως καιρικές συνθήκες, κατάσταση πιστών, και διαθέσιμες υπηρεσίες.

Η ανάπτυξη της εφαρμογής "SnowHub" ανταποκρίνεται στις προκλήσεις που δημιουργήθηκαν κατά τη διάρκεια της πανδημίας και έχει ως στόχο να βελτιώσει την εμπειρία των επισκεπτών, όχι μόνο κατά την περίοδο του COVID-19 αλλά και για το μέλλον.

1.3 Εισαγωγή στο Θέμα

Τα χιονοδρομικά κέντρα αποτελούν σημαντικό κομμάτι του χειμερινού τουρισμού στην Ελλάδα, προσελκύοντας χιλιάδες επισκέπτες κάθε χρόνο. Παρά την αυξανόμενη δημοτικότητα των χειμερινών σπορ, παρατηρείται έλλειψη ολοκληρωμένων ψηφιακών εργαλείων που να παρέχουν στους επισκέπτες εύκολη πρόσβαση σε χρήσιμες πληροφορίες, όπως κατάσταση των πιστών, καιρικές συνθήκες, τιμές εισιτηρίων και δυνατότητες κράτησης. Η ανάγκη για βελτίωση της εμπειρίας των χρηστών και για παροχή ολοκληρωμένων υπηρεσιών μέσω διαδικτύου είναι επιτακτική, ιδιαίτερα σε έναν κόσμο που βασίζεται όλο και περισσότερο στις ψηφιακές τεχνολογίες.

Η παρούσα εργασία επικεντρώνεται στη δημιουργία μιας web εφαρμογής που θα καλύψει αυτό το κενό, προσφέροντας στους χρήστες μια πλατφόρμα όπου μπορούν να βρουν όλες τις απαραίτητες πληροφορίες για τα χιονοδρομικά κέντρα της Ελλάδας. Η εφαρμογή αναπτύχθηκε χρησιμοποιώντας την Angular, ένα δημοφιλές framework για την ανάπτυξη web εφαρμογών, το οποίο προσφέρει ισχυρά εργαλεία για τη δημιουργία δυναμικών και διαδραστικών διεπαφών χρήστη.

Σκοπός της εφαρμογής είναι να ενισχύσει τη χρηστικότητα και την αποτελεσματικότητα στην αναζήτηση πληροφοριών, ενώ ταυτόχρονα να βελτιώσει την εμπειρία του χρήστη. Επιπλέον, η ανάπτυξη αυτής της εφαρμογής παρέχει την ευκαιρία να εξεταστούν σύγχρονες πρακτικές προγραμματισμού και να αξιολογηθεί η συμβολή της τεχνολογίας στον τομέα του τουρισμού.

1.4 Σκοπός και Στόχοι της Εργασίας

Ο βασικός σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης web εφαρμογής που θα εξυπηρετεί τις ανάγκες των χρηστών για πληροφορίες σχετικά με τα χιονοδρομικά κέντρα στην Ελλάδα. Η εφαρμογή αυτή στοχεύει στην παροχή εύχρηστων και άμεσων υπηρεσιών, καλύπτοντας μια ευρεία γκάμα αναγκών των επισκεπτών, από την αναζήτηση πληροφοριών μέχρι την κράτηση εισιτηρίων και την ενημέρωση για τις συνθήκες στα χιονοδρομικά κέντρα.

Οι κύριοι στόχοι της εργασίας είναι οι εξής:

1. **Σχεδιασμός και Υλοποίηση Web Εφαρμογής:** Δημιουργία μιας εύχρηστης και λειτουργικής πλατφόρμας που θα παρέχει στους χρήστες πληροφορίες για τα χιονοδρομικά κέντρα της Ελλάδας. Η εφαρμογή θα πρέπει να διαθέτει διαδραστική διεπαφή χρήστη (UI) και να παρέχει δυναμικές πληροφορίες, όπως τις συνθήκες χιονόπτωσης και τις τιμές των εισιτηρίων.

2. **Βελτίωση Εμπειρίας Χρήστη (UX):** Στόχος είναι η δημιουργία μιας ευχάριστης και απλής εμπειρίας πλοήγησης για τους χρήστες, μειώνοντας τον χρόνο που απαιτείται για την αναζήτηση πληροφοριών και την ολοκλήρωση ενεργειών, όπως οι κρατήσεις.

3. **Ανάπτυξη Με Σύγχρονες Τεχνολογίες:** Η εφαρμογή αναπτύσσεται με την Angular, ένα σύγχρονο και ισχυρό framework που επιτρέπει την ταχεία ανάπτυξη και την υλοποίηση διαδραστικών λειτουργιών. Παράλληλα, εξετάζονται βέλτιστες πρακτικές προγραμματισμού για την επίτευξη υψηλής απόδοσης και ασφάλειας.

4. **Αξιολόγηση και Βελτιστοποίηση:** Η εργασία περιλαμβάνει τη δοκιμή της εφαρμογής σε πραγματικές συνθήκες και την αξιολόγηση της απόδοσης και της χρηστικότητάς της. Βασικός στόχος είναι να διασφαλιστεί ότι η εφαρμογή καλύπτει πλήρως τις ανάγκες των χρηστών και προσφέρει αξιόπιστες υπηρεσίες.

5. **Συμβολή στον Τουριστικό Τομέα:** Μέσω της υλοποίησης αυτής της εφαρμογής, επιδιώκεται η ενίσχυση του χειμερινού τουρισμού στην Ελλάδα, προσφέροντας ένα καινοτόμο εργαλείο που θα προσελκύσει περισσότερους επισκέπτες και θα διευκολύνει την εμπειρία τους.

Η επίτευξη των παραπάνω στόχων θα συμβάλει στην ολοκληρωμένη αξιολόγηση της σημασίας της τεχνολογίας για την ανάπτυξη του τουριστικού τομέα και στη βελτίωση της αλληλεπίδρασης μεταξύ των χρηστών και των χιονοδρομικών κέντρων στην Ελλάδα.

1.5 Δομή της Εργασίας

Στο παρων κεφάλαιο έγινε μια σύντομη παρουσίαση του θέματος και των στόχων της εργασίας. Περιγράφεται το υπόβαθρο και η σημασία της ανάπτυξης της εφαρμογής για τα χιονοδρομικά κέντρα, ενώ προσδιορίζονται οι στόχοι και τα επιδιωκόμενα αποτελέσματα.

Στο δεύτερο κεφάλαιο ακολουθεί η παρουσίαση αλλά και η επεξήγηση των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Πιο συγκεκριμένα, αναλύεται η επιλογή της Angular ως βασικό framework και περιγράφονται και οι υπόλοιπες τεχνολογίες όπως οι βιβλιοθήκες και τα εργαλεία ανάπτυξης.

Στο τρίτο κεφάλαιο περιγράφεται η διαδικασία σχεδιασμού της εφαρμογής, από την αρχιτεκτονική της μέχρι της ανάπτυξη του Front-End. Αναφέρεται η σχεδίαση της διεπαφής χρήστη (UI/UX), η ενσωμάτωση των λειτουργιών και η διαχείριση της ασφάλειας.

Στο τέταρτο κεφάλαιο γίνεται η παρουσίαση της τελικής εφαρμογής, με λεπτομερή ανάλυση των λειτουργιών, των χαρακτηριστικών αλλα και των δυνατοτήτων απο την μεριά του χρήστη.

Τέλος, στο πέμπτο κεφάλαιο αναφέρονται τα συμπεράσματα απο την εκπόνηση αυτής της πτυχιακής εργασίας. Κλείνοντας θα προταθούν ορισμένες προτάσεις και ιδεές για μελλοντική επέκταση του συστήματος καθιστώντας το πιο εύχρηστο με ακόμα περισσότερες λειτουργίες.

Κεφάλαιο 2ο: Τεχνολογίες και Εργαλεία

2.1 Angular

Η **Angular** (Angular 2.x και μεταγενέστερες εκδόσεις) είναι ένα web framework το οποίο χρησιμοποιείται για την δημιουργία Single Page Application (SPA). Η Angular 2 χρησιμοποιεί την TypeScript ως κύρια γλώσσα προγραμματισμού, προσφέροντας στατική τυποποίηση και καλύτερη διαχείριση του κώδικα.

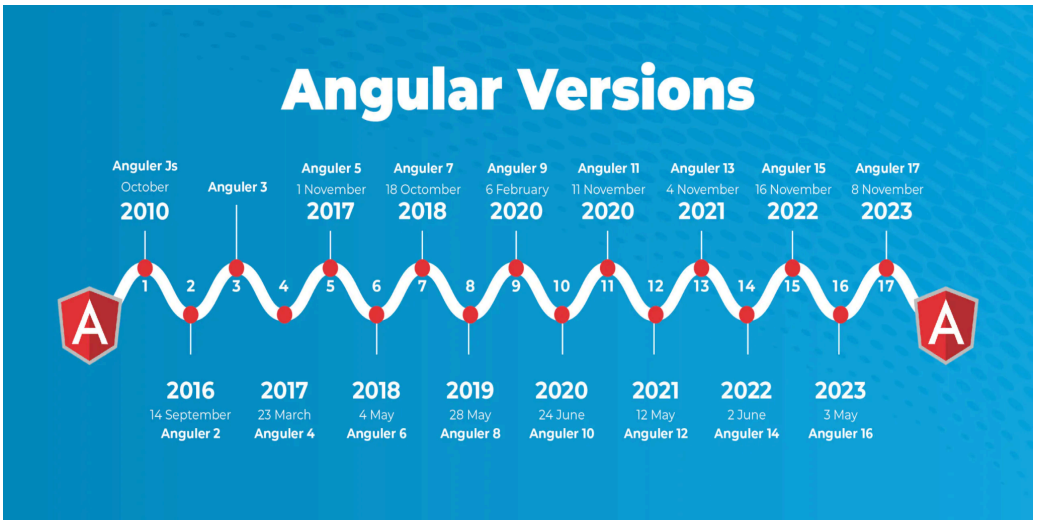
Σχήμα 2.1: Λογότυπο Angular

2.1.1 Ιστορία της Angular

- **AngularJS (2009)**: Η αρχική έκδοση της Angular, γνωστή ως AngularJS, δημιουργήθηκε το 2009 από τους Misko Hevery και Adam Abrons. Ο σκοπός ήταν να απλοποιηθεί η διαδικασία δημιουργίας δυναμικών web εφαρμογών. Πριν από την AngularJS, οι περισσότερες ιστοσελίδες βασίζονταν σε server-side rendering, όπου κάθε αλλαγή απαιτούσε φόρτωση ολόκληρης της σελίδας από τον server. Η AngularJS εισήγαγε το μοντέλο MVC (Model-View-Controller), επιτρέποντας στους προγραμματιστές να διαχωρίσουν την επιχειρηματική λογική (business logic) από την παρουσίαση (view). Με τη χρήση των directives, έδινε τη δυνατότητα για την επέκταση του HTML με συμπεριφορές και δεδομένα, καθιστώντας τις εφαρμογές πιο διαδραστικές και γρήγορες.

- **Angular 2 (2016)**: Το 2016, η ομάδα της Angular ανακοίνωσε μια πλήρη επανεγγραφή του framework, γνωστή ως Angular 2. Σε αυτή τη φάση, το "JS" αφαιρέθηκε από το όνομα, και το πλαίσιο ονομάστηκε απλά Angular. Το Angular 2 ήταν μια σημαντική αναθεώρηση και αλλαγή προσέγγισης. Εισήγαγε τη γλώσσα TypeScript, που είναι ένα υπερσύνολο της JavaScript, παρέχοντας δυνατότητες όπως στατική τυποποίηση, κλάσεις και decorators, βελτιώνοντας έτσι την εμπειρία ανάπτυξης και τη συντηρησιμότητα του κώδικα. Η Angular 2 επέτρεψε την ανάπτυξη πιο δομημένων και ευέλικτων εφαρμογών, υιοθετώντας έννοιες όπως τα components, services, και dependency injection.
- **Angular 4 (2017)**: Αν και δεν υπήρξε Angular 3, η Angular 4 κυκλοφόρησε το 2017 με βελτιώσεις στην απόδοση και νέες δυνατότητες, όπως τα βελτιωμένα animation modules και την υποστήριξη για το TypeScript 2.1 και 2.2. Η επιλογή του αριθμού 4 αντί του 3 έγινε για να συγχρονιστεί με την έκδοση του router package. Η Angular συνέχισε να εξελίσσεται, προσφέροντας εργαλεία όπως το Angular CLI (Command Line Interface), που απλοποίησε τη διαδικασία ανάπτυξης, δοκιμής, και διαχείρισης των εφαρμογών.
- **Angular 5 (2017)**: Η Angular 5 επικεντρώθηκε στη βελτιστοποίηση της απόδοσης, με μικρότερα bundle sizes και ταχύτερο rendering. Εισήγαγε το Angular Universal State Transfer API, επιτρέποντας την αποθήκευση της κατάστασης των εφαρμογών για καλύτερη απόδοση σε server-side rendering. Επίσης, βελτίωσε τη διαχείριση των forms και τα εργαλεία για τη μετάφραση και διεθνοποίηση των εφαρμογών (i18n).
- **Angular 6 (2018)**: Αυτή η έκδοση ενίσχυσε τη modularity, εισάγοντας τα Angular Elements, που επιτρέπουν τη μετατροπή των Angular components σε ανεξάρτητα HTML elements, τα οποία μπορούν να χρησιμοποιηθούν σε μη-Angular εφαρμογές. Επίσης, το Service Worker API ενσωματώθηκε πιο ομαλά, βελτιώνοντας την υποστήριξη για progressive web applications (PWAs). Το Angular CLI αναβαθμίστηκε με τη δυνατότητα προσθήκης (add) χαρακτηριστικών στις εφαρμογές, κάνοντας την ανάπτυξη ακόμα πιο εύκολη.
- **Angular 7 και 8 (2018-2019)**: Αυτές οι εκδόσεις συνέχισαν να βελτιώνουν την απόδοση και τη χρήση της Angular. Η Angular 7 εισήγαγε το Virtual Scrolling και το Drag and Drop με το Angular Material, προσφέροντας πιο φιλικές και γρήγορες εμπειρίες χρήστη. Η Angular 8, από την άλλη, εισήγαγε το Ivy, έναν νέο renderer, που βελτίωσε σημαντικά την απόδοση και τη μεγέθυνση των εφαρμογών. Παράλληλα, πρόσθεσε το Differential Loading για καλύτερη διαχείριση των browsers και πιο αποτελεσματική φόρτωση των εφαρμογών.
- **Angular 9 και Ivy (2020):** Η Angular 9 ήταν μια σημαντική αναβάθμιση, κυρίως λόγω του Ivy, που έγινε πλέον ο προεπιλεγμένος renderer. Το Ivy έφερε σημαντικές βελτιώσεις στην απόδοση, μειώνοντας το μέγεθος των bundles και επιταχύνοντας το rendering. Επίσης, έκανε τον κώδικα πιο προβλέψιμο και ευκολότερο στη συντήρηση. Η μετάβαση στο Ivy σηματοδότησε μια πιο ώριμη φάση για την Angular, καθιστώντας την πιο αποτελεσματική και αξιόπιστη.
- **Angular 10 και Μετέπειτα (2020-Σήμερα):** Με την κυκλοφορία της Angular 10 και των μετέπειτα εκδόσεων, το framework επικεντρώνεται στη βελτίωση της σταθερότητας, της απόδοσης και της ευκολίας χρήσης. Οι εκδόσεις αυτές εισάγουν εργαλεία και βελτιώσεις που βοηθούν τους προγραμματιστές να εντοπίζουν και να διορθώνουν προβλήματα, να δημιουργούν καλύτερα optimized εφαρμογές, και να υιοθετούν τις τελευταίες τεχνολογίες.
- **Σημερινή Χρήση και Μέλλον:** Η Angular σήμερα αποτελεί ένα από τα κορυφαία πλαίσια ανάπτυξης web εφαρμογών, χάρη στην ευελιξία, τη δομή, και την απόδοσή της. Χρησιμοποιείται από μεγάλες επιχειρήσεις και κοινότητες προγραμματιστών για την κατασκευή σύνθετων και υψηλής απόδοσης εφαρμογών. Η συνεχής εξέλιξη της Angular, με

τις τακτικές ενημερώσεις και την υποστήριξη νέων εργαλείων και τεχνολογιών, διασφαλίζει ότι παραμένει στην πρώτη γραμμή της ανάπτυξης web εφαρμογών, προσαρμοσμένη στις σύγχρονες ανάγκες και απαιτήσεις.



Σχήμα 2.2: Χρονοδιάγραμμα εκδόσεων Angular

2.1.2 Βασικά Στοιχεία

Κάποια βασικά στοιχεία της Angular:

1. **Components**: Τα components είναι τα βασικά δομικά στοιχεία της Angular εφαρμογής. Κάθε component αποτελείται από ένα κομμάτι κώδικα (TypeScript) και ένα πρότυπο (HTML), που καθορίζουν τη λογική και την εμφάνιση της εφαρμογής.
2. **Modules**: Η Angular οργανώνει τον κώδικα σε modules, τα οποία ομαδοποιούν τα components, τις υπηρεσίες και άλλα στοιχεία της εφαρμογής, διευκολύνοντας τη διαχείριση και την επαναχρησιμοποίηση του κώδικα.
3. **Services** και **Dependency Injection**: Οι υπηρεσίες (services) είναι κλάσεις που παρέχουν συγκεκριμένες λειτουργίες όπως πρόσβαση σε δεδομένα και επεξεργασία αυτών καθώς και την επικοινωνία με εξωτερικά APIs. Το Dependency Injection επιτρέπει την εύκολη διαχείριση και χρήση αυτών των υπηρεσιών σε όλη την εφαρμογή.
4. **Routing**: Η Angular παρέχει ένα ισχυρό σύστημα πλοήγησης (routing) που επιτρέπει την κατασκευή SPA, όπου η πλοήγηση μεταξύ διαφορετικών σελίδων γίνεται χωρίς ανανέωση της σελίδας.
5. **Directives**: Τα directives είναι ειδικές οδηγίες που επεκτείνουν τη λειτουργικότητα του HTML, επιτρέποντας τη δημιουργία προσαρμοσμένων στοιχείων και την τροποποίηση της συμπεριφοράς των υπαρχόντων στοιχείων. Μερικά από τα πιο βασικά directives που παρέχει η Angular είναι το ngIf και το ngFor.

2.2 Typescript

Η **Typescript** είναι ένα υπερσύνολο της JavaScript που αναπτύχθηκε από τη Microsoft. Προσθέτει στατική τυποποίηση στον κώδικα, επιτρέποντας στους προγραμματιστές να ορίζουν τύπους δεδομένων για τις μεταβλητές, τις συναρτήσεις και τις παραμέτρους. Αυτό βοηθά στον εντοπισμό σφαλμάτων κατά τη διάρκεια της ανάπτυξης, κάνοντας τον κώδικα πιο ασφαλή και ευκολότερο στη

συντήρηση. Επιπλέον, η Typescript υποστηρίζει χαρακτηριστικά όπως κλάσεις, interfaces και enums, τα οποία διευκολύνουν την ανάπτυξη μεγάλων και πολύπλοκων εφαρμογών. Η Typescript μεταγλωττίζεται σε JavaScript, καθιστώντας την πλήρως συμβατή με όλες τις πλατφόρμες που υποστηρίζουν JavaScript. Χρησιμοποιείται ευρέως σε μοντέρνα frameworks όπως το Angular, προσφέροντας βελτιωμένη εμπειρία ανάπτυξης.

Σχήμα 2.3: Λογότυπο Typescript

2.3 LESS CSS

Η **LESS** (Leaner Style Sheets) είναι μια προεπεξεργαστική γλώσσα για το CSS, η οποία προσθέτει λειτουργίες που δεν είναι διαθέσιμες στο παραδοσιακό CSS, όπως μεταβλητές, συναρτήσεις, και ενσωματωμένες μαθηματικές πράξεις. Αναπτύχθηκε για να διευκολύνει τη διαχείριση και τη συντήρηση μεγάλων CSS αρχείων, επιτρέποντας στους προγραμματιστές να γράφουν πιο οργανωμένο και επαναχρησιμοποιήσιμο κώδικα. Με τη χρήση μεταβλητών, οι προγραμματιστές μπορούν να ορίζουν κοινές τιμές, όπως χρώματα και μεγέθη, σε ένα σημείο και να τις επαναχρησιμοποιούν σε ολόκληρο το αρχείο CSS. Οι συναρτήσεις επιτρέπουν τη δημιουργία πιο σύνθετων στυλ, ενώ οι ενσωματωμένες μαθηματικές πράξεις βοηθούν στον καθορισμό σχετικών μεγεθών και αποστάσεων. Η LESS μεταγλωττίζεται σε κανονικό CSS, και μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον όπου υποστηρίζεται το CSS, προσφέροντας μεγαλύτερη ευελιξία και παραγωγικότητα στη διαχείριση των στυλ.

Σχήμα 2.4: Λογότυπο LESS

2.4 Βιβλιοθήκες

Για την υλοποίηση της παρούσας εργασίας χρησιμοποιήθηκαν αρκετές βιβλιοθήκες ανοιχτού κώδικα, οι οποίες συνέβαλαν σημαντικά στην ανάπτυξη της εφαρμογής, διευκολύνοντας την υλοποίηση ορισμένων χαρακτηριστικών και μειώνοντας τον απαιτούμενο κώδικα. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι κυρίως βασισμένες στην JavaScript και στην Typescript, υποστηρίζοντας τη λειτουργικότητα του frontend της εφαρμογής, καθώς και την ενσωμάτωση διαφόρων συμπληρωματικών εργαλείων και τεχνολογιών.

2.4.1 Angular Material

Το **Angular Material** είναι μια βιβλιοθήκη UI components για την Angular, σχεδιασμένη σύμφωνα με τις αρχές του Material Design της Google. Προσφέρει μια μεγάλη ποικιλία από έτοιμα components, όπως κουμπιά, φόρμες, πίνακες κ.ά., που διευκολύνουν τη δημιουργία μοντέρνων user interfaces. Το Angular Material υποστηρίζει responsive design, εξασφαλίζοντας ότι οι εφαρμογές προσαρμόζονται ομαλά σε διάφορες συσκευές. Επίσης, παρέχει δυνατότητες προσαρμογής, επιτρέποντας στους προγραμματιστές να τροποποιούν την εμφάνιση και τη συμπεριφορά των components για να ταιριάζουν με τις ανάγκες της εφαρμογής τους. Ως μέρος του οικοσυστήματος Angular, ενημερώνεται τακτικά και προσφέρει σταθερότητα και υποστήριξη για την ανάπτυξη σύγχρονων εφαρμογών.

2.4.2 RxJs

Η **RxJS** (Reactive Extensions for JavaScript) είναι μια βιβλιοθήκη για τη διαχείριση ασύγχρονων δεδομένων και συμβάντων μέσω παρατηρήσιμων (observables). Χρησιμοποιείται ευρέως στην Angular για την επεξεργασία δεδομένων, όπως αιτήματα HTTP και συμβάντα χρηστών.

Κάποια βασικά στοιχεία της RxJs:

1. **Observables**: Παρακολουθούν και εκπέμπουν δεδομένα με την πάροδο του χρόνου.
2. **Operators**: Επιτρέπουν τον μετασχηματισμό και τη διαχείριση των δεδομένων (π.χ., map, filter).
3. **Subscription**: Εγγραφή σε observables για λήψη δεδομένων.
4. **Subject**: Συνδυάζει την παρατήρηση και την εκπομπή δεδομένων.

Η RxJS κάνει τον χειρισμό ασύγχρονων λειτουργιών πιο ευέλικτο και οργανωμένο, διευκολύνοντας την ανάπτυξη σύνθετων εφαρμογών.

Σχήμα 2.5: Λογότυπο RxJs

2.4.3 Swiper

Το **Swiper** είναι μια δημοφιλής βιβλιοθήκη JavaScript για τη δημιουργία responsive sliders και carousels. Σχεδιασμένο για να παρέχει εξαιρετική εμπειρία χρήστη σε κινητές και επιτραπέζιες συσκευές, προσφέρει ευκολία στη χρήση με υποστήριξη για μετακινήσεις μέσω drag-and-drop, touch gestures, και προσαρμόσιμα εφέ. Επιπλέον, περιλαμβάνει δυνατότητες όπως αυτοματοποιημένη κύλιση, pagination, και navigation arrows, καθιστώντας το ιδανικό για την παρουσίαση περιεχομένου με εικόνες ή άλλα στοιχεία.

Σχήμα 2.6: Λογότυπο Swiper

2.4.4 PhotoSwipe

Το **PhotoSwipe** είναι μια βιβλιοθήκη JavaScript που επιτρέπει την προβολή εικόνων σε lightbox με επαγγελματική εμφάνιση και λειτουργικότητα. Προσφέρει μια responsive και ευέλικτη εμπειρία χρήστη, υποστηρίζοντας touch gestures όπως ζουμ και μετακίνηση, καθώς και την προβολή εικόνων σε πλήρη οθόνη. Το PhotoSwipe είναι εύκολο στην ενσωμάτωσή του και παρέχει μια ομαλή μετάβαση μεταξύ εικόνων, κάνοντάς το ιδανικό για εφαρμογές και ιστοσελίδες που απαιτούν κομψή παρουσίαση φωτογραφιών και οπτικού περιεχομένου.

Σχήμα 2.7: Λογότυπο PhotoSwipe

2.5 Εργαλεία

Σε αυτό το υποκεφάλαιο θα παρουσιαστούν τα εργαλεία που χρησιμοποιήθηκαν για τον σχεδιασμό, την ανάπτυξη, τη δοκιμή και τη διαχείριση του κώδικα της εφαρμογής. Περιλαμβάνει επεξεργαστές κώδικα και πλατφόρμες που διευκόλυναν την ανάπτυξη του frontend της εφαρμογής, καθώς και εργαλεία για την αποθήκευση και τον έλεγχο έκδοσης του κώδικα. Ειδικότερα, παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της διεπαφής χρήστη, τη διαχείριση του κώδικα, την αυτοματοποίηση των διαδικασιών και τη δοκιμή της εφαρμογής.

2.5.1 Visual Studio Code

Το **Visual Studio Code** (VSCode) είναι ένας δωρεάν, ανοιχτού κώδικα επεξεργαστής κώδικα που αναπτύχθηκε από τη Microsoft. Προσφέρει ένα ευρύ φάσμα λειτουργιών που το καθιστούν ιδανικό για την ανάπτυξη εφαρμογών, ειδικά για προγραμματιστές JavaScript και TypeScript. Το VSCode παρέχει υποστήριξη για σύνταξη κώδικα, ενσωματωμένο τερματικό, debugging, καθώς και πλούσιο οικοσύστημα επεκτάσεων που επιτρέπουν την προσθήκη νέων λειτουργιών και τη διαχείριση των διαφόρων εργαλείων και πλαισίων που χρησιμοποιούνται στην ανάπτυξη. Επιπλέον, η ευχρηστία και η ταχύτητά του το καθιστούν δημοφιλή επιλογή για την ανάπτυξη web εφαρμογών όπως αυτή.

Σχήμα 2.8: Λογότυπο Visual Studio Code

2.5.2 Git και Github

Το **Git** είναι ένα σύστημα ελέγχου έκδοσης ανοιχτού κώδικα που χρησιμοποιείται για την παρακολούθηση αλλαγών στον κώδικα κατά την ανάπτυξη ενός έργου. Επιτρέπει στους προγραμματιστές να διαχειρίζονται διαφορετικές εκδόσεις του κώδικα, να συνεργάζονται αποτελεσματικά σε ομάδες και να διασφαλίζουν ότι οποιεσδήποτε αλλαγές μπορούν να αναθεωρηθούν και να ανακληθούν αν χρειαστεί. Μέσω της δημιουργίας και συγχώνευσης κλάδων (branches), το Git καθιστά δυνατή την ανάπτυξη νέων χαρακτηριστικών χωρίς να επηρεάζονται οι κύριες λειτουργίες του έργου.

Το **GitHub** είναι μια διαδικτυακή πλατφόρμα που βασίζεται στο Git και παρέχει αποθήκευση και διαχείριση αποθετηρίων κώδικα. Προσφέρει πρόσθετα εργαλεία για συνεργασία, όπως pull requests και issues, επιτρέποντας στους προγραμματιστές να αναθεωρούν κώδικα, να συζητούν αλλαγές και να εντοπίζουν σφάλματα. Επιπλέον, υποστηρίζει συνεχή ενσωμάτωση και αυτοματοποίηση με τη χρήση εργαλείων όπως το GitHub Actions, καθιστώντας τη διαδικασία ανάπτυξης πιο αποδοτική.

Σχήμα 2.9: Λογότυπο GitHub

2.5.3 Postman

Το **Postman** είναι ένα εργαλείο ανάπτυξης που χρησιμοποιείται κυρίως για τη δοκιμή και τον εντοπισμό σφαλμάτων API (Application Programming Interface). Παρέχει μια εύχρηστη διεπαφή για την αποστολή αιτημάτων HTTP, την προβολή των απαντήσεων και τη διαχείριση των API endpoints. Με το Postman, οι προγραμματιστές μπορούν να δοκιμάσουν την επικοινωνία μεταξύ του frontend και του backend, να επαληθεύσουν την ορθότητα των δεδομένων και να αυτοματοποιήσουν τις δοκιμές API. Επιπλέον, προσφέρει δυνατότητες όπως η αποθήκευση και η κοινή χρήση αιτημάτων, γεγονός που το καθιστά ιδιαίτερα χρήσιμο εργαλείο για την ανάπτυξη και τη συντήρηση σύγχρονων εφαρμογών.

Σχήμα 2.10: Λογότυπο Postman

2.5.4 GitHub Actions

Το **GitHub Actions** είναι ένα εργαλείο αυτοματοποίησης που ενσωματώνεται στη πλατφόρμα του GitHub, επιτρέποντας στους προγραμματιστές να δημιουργούν ροές εργασίας (workflows) για την αυτοματοποίηση διαδικασιών ανάπτυξης λογισμικού. Μέσω του GitHub Actions, μπορείτε να αυτοματοποιήσετε εργασίες όπως η συνεχής ενσωμάτωση (CI), η συνεχής ανάπτυξη (CD), οι δοκιμές κώδικα, και η διανομή της εφαρμογής. Η ρύθμιση των workflows γίνεται με αρχεία YAML, όπου οι

προγραμματιστές καθορίζουν τις ενέργειες που θα εκτελεστούν σε κάθε στάδιο της διαδικασίας ανάπτυξης. Με την ευκολία και την ευελιξία που προσφέρει, το GitHub Actions βοηθάει στην αύξηση της αποδοτικότητας και της αξιοπιστίας της ανάπτυξης λογισμικού.

Κεφάλαιο 3ο: Σχεδίαση και Υλοποίηση της Εφαρμογής

3.1 Σχεδιασμός Αρχιτεκτονικής του Front-End

Η αρχιτεκτονική της εφαρμογής βασίστηκε στη δομή που προτείνεται από την ίδια την Angular, επιδιώκοντας να πετύχει επεκτασιμότητα, ευκολία συντήρησης και καλή απόδοση. Η βασική φιλοσοφία της αρχιτεκτονικής ήταν να οργανώσει την εφαρμογή σε ξεκάθαρα διαχωρισμένα μέρη, όπου κάθε μέρος έχει συγκεκριμένες ευθύνες και λειτουργίες. Ο διαχωρισμός αυτός διευκολύνει την ανάπτυξη, τη συντήρηση και την επέκταση της εφαρμογής στο μέλλον.

Η εφαρμογή υλοποιήθηκε σύμφωνα με την αρχή του Separation of Concerns, όπου η λογική της διεπαφής χρήστη (UI), η λογική των δεδομένων και η επιχειρησιακή λογική (Business Logic) διαχωρίστηκαν σε διαφορετικά μέρη. Παρότι σε ορισμένα components η λογική του UI και των δεδομένων συνδυάζεται για λόγους απλότητας και αποτελεσματικότητας, σε άλλες περιπτώσεις αυτά τα στοιχεία διαχωρίστηκαν πλήρως, ακολουθώντας τις αρχές του modularity και της επεκτασιμότητας.

Η διαχείριση των δεδομένων και οι κλήσεις σε APIs υλοποιήθηκαν μέσω ειδικών services, τα οποία δημιουργήθηκαν με σκοπό να διαχωρίσουν την επικοινωνία με εξωτερικά συστήματα από τη λογική του UI. Με αυτόν τον τρόπο, η εφαρμογή διατηρεί τη δυνατότητα να τροποποιεί τη διαχείριση των δεδομένων χωρίς να επηρεάζεται η εμφάνιση ή η βασική λειτουργικότητά της.

Επιπλέον, η αρχιτεκτονική της εφαρμογής ενσωματώνει μηχανισμούς για την ευκολότερη διαχείριση των εξαρτήσεων και τη βελτίωση της συντηρησιμότητας του κώδικα. Μέσω της κατάλληλης οργάνωσης σε modules και της χρήσης του Angular dependency injection, η εφαρμογή είναι ικανή να διαχειρίζεται με αποδοτικό τρόπο την επαναχρησιμοποίηση του κώδικα και να διευκολύνει τη μελλοντική επέκταση με νέα χαρακτηριστικά.

3.2 Σχεδιασμός Διεπαφής Χρήστη (UI/UX)

Η σχεδίαση της διεπαφής χρήστη (UI) της εφαρμογής επικεντρώνεται σε τρεις κύριες σελίδες, καθεμία από τις οποίες έχει σχεδιαστεί για να προσφέρει μια κατανοητή και ευχάριστη εμπειρία στον χρήστη:

1. **Αρχική Σελίδα**: Αυτή η σελίδα είναι η πρώτη που συναντούν οι χρήστες και περιλαμβάνει ένα hero slider στην κορυφή, που παρουσιάζει εντυπωσιακές φωτογραφίες από χιονοδρομικά κέντρα. Κάτω από αυτό, υπάρχει ένα slider με τα πιο πρόσφατα χιονοδρομικά κέντρα, δίνοντας στους χρήστες μια γρήγορη εικόνα των διαθέσιμων επιλογών. Επιπλέον, περιλαμβάνεται ένας διαδραστικός χάρτης για τον καιρό, που παρέχει πληροφορίες σε πραγματικό χρόνο για τις καιρικές συνθήκες στα χιονοδρομικά κέντρα, βοηθώντας τους χρήστες να προγραμματίσουν την επίσκεψή τους.

2. **Σελίδα Χιονοδρομικών Κέντρων**: Σε αυτήν τη σελίδα, οι χρήστες βλέπουν μια πλήρη λίστα με χιονοδρομικά κέντρα σε μορφή καρτών. Κάθε κάρτα παρουσιάζει βασικές πληροφορίες για το κάθε χιονοδρομικό κέντρο, επιτρέποντας στους χρήστες να περιηγηθούν εύκολα και να βρουν αυτό που τους ενδιαφέρει.

3. **Σελίδα Χιονοδρομικού Κέντρου**: Η σελίδα αυτή παρέχει λεπτομέρειες για το επιλεγμένο χιονοδρομικό κέντρο. Στην κορυφή, εμφανίζεται ο τίτλος με το όνομα και την τοποθεσία του κέντρου. Ακολουθεί ένα slider με φωτογραφίες του χιονοδρομικού κέντρου. Κάτω από αυτό, υπάρχει ένα κουμπί που ανοίγει μια εικόνα με όλες τις πίστες του χιονοδρομικού. Επίσης, η σελίδα περιλαμβάνει ένα πλαίσιο με πληροφορίες και δραστηριότητες, ένα πλαίσιο με τις πίστες όπου κάθε πίστα έχει ένα bullet με το χρώμα της δυσκολίας της, ένα πλαίσιο με τους αναβατήρες όπου κάθε αναβατήρας έχει ένα bullet με κόκκινο ή πράσινο χρώμα για να δείξει αν είναι ανοιχτός ή όχι, και τέλος, ένα πλαίσιο με την τιμή του ημερήσιου εισιτηρίου, επιλογές για τον αριθμό των επισκεπτών, και ένα κουμπί "Κράτηση".

Η εφαρμογή χρησιμοποιεί **responsive design** για να διασφαλίσει ότι η διεπαφή προσαρμόζεται σε διάφορες συσκευές και μεγέθη οθονών. Τα χρώματα και η γραμματοσειρά έχουν επιλεγεί ώστε να προσφέρουν ευχάριστη και συνεπή οπτική εμπειρία, χωρίς να είναι υπερβολικά έντονα.

3.3 Ανάπτυξη και Υλοποίηση Front-End

Η υλοποίηση του Front-End έγινε με οργάνωση σε φακέλους ανάλογα με τα συστατικά της Angular και τις κοινές πρακτικές που εφαρμόζονται με την χρήση του συγκεκριμένου framework.

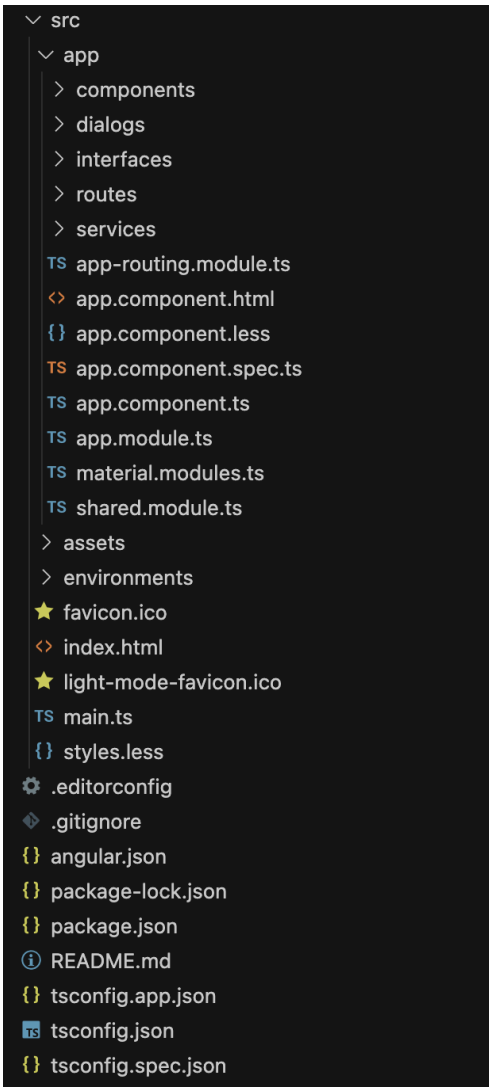
Η δομή του project περιλαμβάνει δύο βασικούς φακέλους¨

1. **Φάκελος /src**: Αυτός ο φάκελος περιέχει όλα τα αρχεία του κώδικα της εφαρμογής. Στο εσωτερικό του, υπάρχουν δύο υποφακέλοι:
    - **Φάκελος /assets**: Περιέχει όλα τα στατικά αρχεία της εφαρμογής, όπως εικονίδια και εικόνες, που φορτώνονται στον browser όταν η εφαρμογή εκτελείται.
    - **Φάκελος /app**: Περιλαμβάνει όλους τους κύριους φακέλους και αρχεία της εφαρμογής, όπως components, services, και modules. Τα components διαχειρίζονται μέσω του components.module.ts, ενώ οι διάφορες διαδρομές της εφαρμογής καθορίζονται στο app-routing.module.ts.
2. **Εξωτερικος Φάκελος (/snowhub)**:
    - **Φάκελος /node_modules**: Περιέχει όλες τις βιβλιοθήκες που χρησιμοποιούνται από την εφαρμογή, όπως η Angular και άλλες εξαρτήσεις που δηλώνονται στο package.json.
    - **Αρχεία Παραμετροποίησης**: Περιλαμβάνουν αρχεία όπως tsconfig.json που ρυθμίζουν την TypeScript, και διάφορα αρχεία παραμετροποίησης για τη διαχείριση της ανάπτυξης και της εκτέλεσης της εφαρμογής.

Επιπλέον, στον φάκελο **/src** βρίσκονται τα αρχεία:

- **index.html**: Το κύριο HTML αρχείο που φορτώνεται στον browser και φιλοξενεί το root component της εφαρμογής.
- **styles.css**: Το αρχείο που περιλαμβάνει τα βασικά στυλ της εφαρμογής.

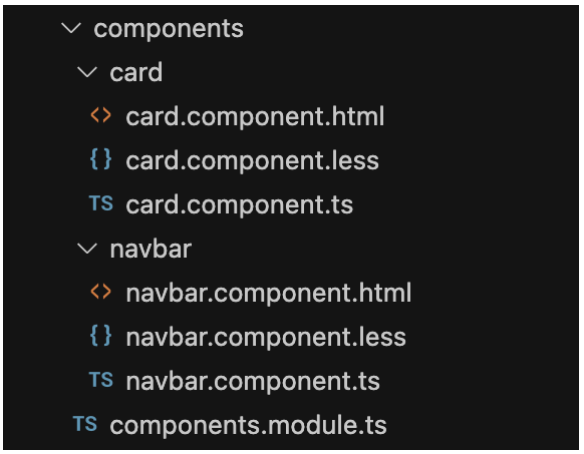
- **main.ts**: Το κύριο TypeScript αρχείο που εκκινεί την εφαρμογή, το οποίο ενσωματώνει το **AppModule** και ξεκινά την εκτέλεση της εφαρμογής στο browser.



Σχήμα 3.1: Δομή του project

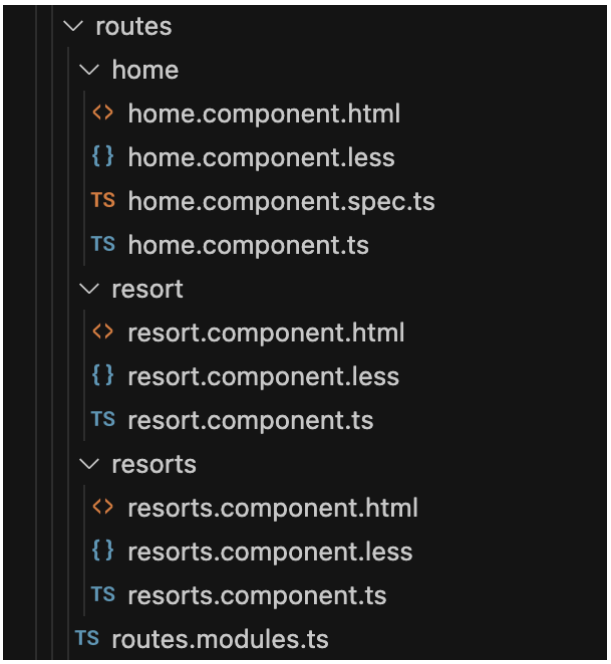
Η ομαδοποίηση των components μεσα στο app γίνεται ως εξής:

- **components:** περιλαμβάνει τους φακέλους με τα βασικά component της εφαρμογής όπως οι κάρτες των χιονοδρομικών (card) και την γραμμή πλοήγησης (navbar). Τα ονόματα των φακέλων είναι ίδια με τις κλάσεις των components που περιέχονται σε αυτούς, διατηρώντας έτσι μια οργανωμένη δομή. Κάθε φάκελος περιέχει τα τρία αρχεία που συνθέτουν ένα Angular component: το .component.ts, που περιλαμβάνει τη λογική και τις μεθόδους του component, το .component.html, που αντιπροσωπεύει το HTML template του component, και το .component.less, το οποίο περιέχει τα LESS CSS styles για τη μορφοποίηση του component. Κάθε component χαρακτηρίζεται από το annotation της Angular @Component, όπου δηλώνονται τα αρχεία του template και του styling, καθώς και ο selector του component, ο οποίος προστίθεται στο HTML για να ενσωματώσει το component μέσα σε ένα template.



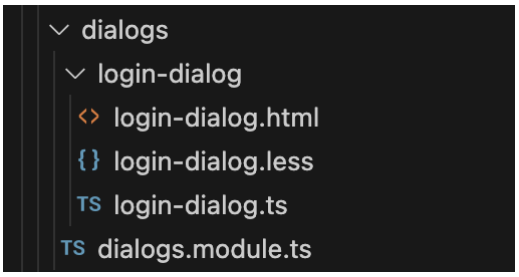
Σχήμα 3.2: Δομή φακέλου components

- **routes**: σε αυτόν τον φάκελο υπάρχουν τα components που αντιπροσωπεύουν τις ξεχωριστές σελίδες της εφαρμογής. Έτσι έχουμε τρια components ένα για κάθε σελίδα: /home.component όπου μέσα σε αυτό βρίσκεται η αρχική σελίδα που εμφανίζεται στον χρήστη, /resorts.component το οποίο είναι η σελίδα που εμφανίζει όλα τα χιονοδρομικά και το /resort.component που είναι η σελίδα με τις πληροφορίες του κάθε χιονοδρομικού.



Σχήμα 3.3: Δομή φακέλου routes

- **dialogs**: εδώ βρίσκονται τα components τα οποία είναι modals (ή αλλιώς dialogs). Τα dialogs είναι ουσιαστικά components τα οποία εμφανίζονται σαν pop-up επάνω από μία άλλη σελίδα.



Σχήμα 3.4: Δομή φακέλου dialogs

- **services**: σε αυτόν τον φάκελο βρίσκονται τα services της Angular που χρησιμοποιούνται για την διαχείριση του bussiness logic και των δεδομένων που χρειάζονται τα components. Κάθε service είναι υπεύθυνο για την αλληλεπίδραση με εξωτερικές πηγές δεδομένων, όπως APIs, καθώς και για την επεξεργασία και διανομή αυτών των δεδομένων στα components. Τα services υλοποιούνται μέσω της Angular χρησιμοποιώντας το annotation @Injectable, επιτρέποντάς τους να χρησιμοποιούνται εύκολα σε διάφορα μέρη της εφαρμογής. Τέλος κατηγοριοποιούνται με βάση το model και η ονομασία τους αποτελείται από το όνομα του model και το .service.ts ως επίθεμα (πχ. resort.service.ts)

Σχήμα 3.5: Δομή φακέλου services

Τα μονοπάτια που παρέχει η εφαρμογή για να εμφανιστούν οι σελίδες είναι χωρισμένα όπως τα routes με βάση τον ρόλο και την αντίστοιχη λειτουργία που εκτελεί το route που εμφανίζεται. Στο αρχείο app-routing.module.ts δηλώνονται τα μονοπάτια μέσα σε μία λιστα με όνομα routes με ορίσματα: το μονοπάτι ως αλφαριθμητικό και το component το οποίο θα εμφανίζεται όταν ο χρήστης επισκέπτεται το συγκεκριμένο μονοπάτι (πχ. { path: 'resorts', component: ResortsComponent }).

Τα μονοπάτια της εφαρμογής μαζί με τα components που εμφανίζουν είναι τα εξής:

- / : το αρχικό μονοπάτι της εφαρμογής, το οποίο ειναι και το μόνο που εμφανίζεται στην περίπτωση που δεν υπαρχει κανενα μονοπάτι στο url που επισκέπτεται ο χρήστης και εμφανίζει το HomeComponent.
- /resorts: η σελίδα που εμφανίζει την λίστα με τις κάρτες όλων των χιονοδρομικών και εμφανίζει το ResortsComponent.
- /resorts/:resortId: αυτή είναι η σελίδα που εμφανίζει πληροφορίες και εικόνες για το επιλεγμένο χιονοδρομικό αλλά και την δυνατότητα κράτησης και εμφανίζει το

ResortComponent. Το :resortId είναι το Id του χιονοδρομικού που επιλέχθηκε και με αυτόν τον τρόπο μπορούμε μέσα στο component να χρησιμοποιήσουμε αυτό το Id.

Στην περίπτωση που ο χρήστης προσπαθήσει να επισκεφτεί κάποιο μονοπάτι το οποίο δεν το έχουμε ορίσει τότε η εφαρμογή τον "στέλνει" στο αρχικό μονοπάτι. Αυτό το καταφέρνουμε με τα εξής ορίσματα: { path: '**', redirectTo: '' }, τα '**' ορίζουν το οτιδήποτε διαφορετικό έχει βάλει ο χρήστης από αυτά που έχουμε ήδη ορίσει και το redirectTo: '' πηγαίνει τον χρήστη στο αρχικό μονοπάτι.

```
const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'resorts', component: ResortsComponent },
  { path: 'resorts/:resortId', component: ResortComponent },
  { path: '**', redirectTo: '' }
];
```

Σχήμα 3.6: app-routing.module.ts

3.4 Ανάλυση και Χρήση των Services στην Εφαρμογή

Στην εφαρμογή, τα services διαδραματίζουν κρίσιμο ρόλο στη διαχείριση και την επεξεργασία δεδομένων, καθώς και στη διασύνδεση με εξωτερικά APIs. Η χρήση των services προσφέρει μια καθαρή και οργανωμένη προσέγγιση για την υλοποίηση της επιχειρηματικής λογικής και τη διαχείριση της επικοινωνίας μεταξύ των διαφόρων components της εφαρμογής. Στην παρακάτω ενότητα, θα αναλύσουμε τα διάφορα services που έχουν αναπτυχθεί στην εφαρμογή, εξηγώντας τη λειτουργία και τη χρησιμότητά τους.

3.4.1 Αρχιτεκτονική και Σχεδιασμός των Services

Στην εφαρμογή, τα services είναι ένας σημαντικός πυλώνας της αρχιτεκτονικής, διαχωρίζοντας τη λογική της εφαρμογής από την παρουσίαση της. Με αυτή την προσέγγιση, διασφαλίζεται ότι η επιχειρηματική λογική είναι επαναχρησιμοποιήσιμη και ανεξάρτητη από τα UI components, επιτρέποντας καλύτερη συντήρηση και επεκτασιμότητα.

Τα services ακολουθούν την αρχή της ενσωμάτωσης των δεδομένων και της επιχειρηματικής λογικής σε έναν διαχειρίσιμο χώρο, που μπορεί να προσπελαστεί από διάφορα μέρη της εφαρμογής. Αυτό επιτρέπει στα components να παραμένουν ελαφριά και συγκεντρωμένα στη διαχείριση του UI, χωρίς να χρειάζεται να περιέχουν πολύπλοκη λογική ή διαδικασίες.

Η γενική αρχιτεκτονική των services περιλαμβάνει:

- **Κεντρική Διαχείριση Δεδομένων**: Τα services λειτουργούν ως αποθηκευτικοί χώροι για τα δεδομένα της εφαρμογής, είτε αυτά προέρχονται από τοπική κατάσταση είτε από εξωτερικές πηγές (APIs).

- **Διασύνδεση με Εξωτερικά APIs**: Πολλά services είναι υπεύθυνα για την επικοινωνία με εξωτερικά APIs, λαμβάνοντας ή στέλνοντας δεδομένα και μετατρέποντας τα σε μορφή που μπορεί να χρησιμοποιηθεί από τα components.
- **Επαναχρησιμοποίηση Λογικής**: Τα services επιτρέπουν την επαναχρησιμοποίηση σύνθετων λειτουργιών και μεθόδων σε διαφορετικά μέρη της εφαρμογής, εξαλείφοντας την ανάγκη για επαναλαμβανόμενο κώδικα.
- **Αλληλεπίδραση με Components:** Τα services παρέχουν μια σταθερή διεπαφή (API) για τα components της εφαρμογής, διευκολύνοντας την πρόσβαση σε δεδομένα και λειτουργίες.

Η αρχιτεκτονική αυτή προάγει την αρχή της διαίρεσης των αρμοδιοτήτων, καθιστώντας τον κώδικα πιο καθαρό και ευκολότερο στην κατανόηση και τη συντήρηση.

3.4.2 Ανάλυση FavouriteService

Το **FavouriteService** διαχειρίζεται τις αγαπημένες επιλογές των χρηστών, επιτρέποντάς τους να προσθέτουν και να αφαιρούν χιονοδρομικά κέντρα στη λίστα των αγαπημένων τους. Αποθηκεύει αυτές τις επιλογές τοπικά στον browser, χρησιμοποιώντας το **localStorage**, ώστε να διατηρούνται ακόμη και μετά από επανεκκίνηση της εφαρμογής. Αυτό το service συνδέεται με τον τρέχοντα χρήστη μέσω του **UserService**, επιτρέποντας την αποθήκευση των αγαπημένων με βάση το μοναδικό αναγνωριστικό του χρήστη.

Ανάλυση Μεθόδων και Λειτουργικότητας

1. **addFavourite(resortId: number)**
   - **Λειτουργία**: Προσθέτει ένα χιονοδρομικό κέντρο στη λίστα των αγαπημένων.
   - **Λεπτομέρειες Υλοποίησης**:
     - Προσθέτει το resortId στη λίστα favourites.
     - Αποθηκεύει την ενημερωμένη λίστα στο localStorage με το μοναδικό αναγνωριστικό του χρήστη (user.id) ως κλειδί.
   - **Σημασία**: Επιτρέπει στον χρήστη να προσθέτει γρήγορα και εύκολα τα χιονοδρομικά κέντρα που του ενδιαφέρουν.

```
addFavourite(resortId: number) {
  this.favourites.push(resortId);
  localStorage.setItem(this.userService.user.id, JSON.stringify(this.favourites))
}
```

Σχήμα 3.7: addFavourite()

2. **removeFavourite(resortId: number)**
   - **Λειτουργία:** Αφαιρεί ένα χιονοδρομικό κέντρο από τη λίστα των αγαπημένων.
   - **Λεπτομέρειες Υλοποίησης:**
     - Φιλτράρει τη λίστα favourites για να αφαιρέσει το resortId.
     - Ενημερώνει το localStorage με τη νέα λίστα, χρησιμοποιώντας το user.id ως κλειδί**.**
   - **Σημασία:** Δίνει στον χρήστη την ελευθερία να διαχειρίζεται τις αγαπημένες του επιλογές, αφαιρώντας χιονοδρομικά κέντρα που δεν τον ενδιαφέρουν πλέον.

```
removeFavourite(resortId: number) {
  this.favourites = this.favourites.filter(id => id !== resortId);
  localStorage.setItem(this.userService.user.id, JSON.stringify(this.favourites))
}
```

Σχήμα 3.8: removeFavourite()

3. **setFavourites()**
   - **Λειτουργία:** Φορτώνει τις αγαπημένες επιλογές του χρήστη από το localStorage κατά την εκκίνηση της εφαρμογής.
   - **Λεπτομέρειες Υλοποίησης:**
     - Ανακτά τα αποθηκευμένα δεδομένα από το localStorage χρησιμοποιώντας το user.id.
     - Αν δεν υπάρχουν δεδομένα, αρχικοποιεί τη λίστα ως κενή.
   - **Σημασία:** Διασφαλίζει ότι οι αγαπημένες επιλογές του χρήστη είναι διαθέσιμες μόλις ανοίξει η εφαρμογή, διατηρώντας μια συνεπή εμπειρία χρήστη.

```
setFavourites() {
  this.favourites = JSON.parse(localStorage.getItem(this.userService.user.id) ?? '[]');
}
```

Σχήμα 3.9: setFavourites()

4. **isFavourite(resortId: number)**
   - **Λειτουργία:** Ελέγχει αν ένα συγκεκριμένο χιονοδρομικό κέντρο είναι στη λίστα των αγαπημένων του χρήστη.
   - **Λεπτομέρειες Υλοποίησης:**
     - Επιστρέφει true αν το resortId υπάρχει στη λίστα favourites, αλλιώς false.
   - **Σημασία**: Χρησιμοποιείται για να ενημερώνει το UI σχετικά με το αν ένα χιονοδρομικό κέντρο είναι ήδη αγαπημένο ή όχι, βελτιώνοντας την εμπειρία του χρήστη.

```
isFavourite(resortId: number) {
  return this.favourites.includes(resortId);
}
```

Σχήμα 3.10: isFavourite()

Ρόλος και Χρησιμότητα του Service

Το **FavouriteService** παίζει κρίσιμο ρόλο στην προσωποποίηση της εμπειρίας του χρήστη. Επιτρέπει την αποθήκευση των αγαπημένων χιονοδρομικών κέντρων με απλό και αποδοτικό τρόπο, παρέχοντας ένα μέσο για τη γρήγορη πρόσβαση σε αυτά σε μελλοντικές επισκέψεις. Η χρήση του localStorage για την αποθήκευση αυτών των δεδομένων διασφαλίζει ότι οι επιλογές του χρήστη διατηρούνται ακόμα και όταν ο χρήστης κλείσει και ανοίξει ξανά την εφαρμογή.

3.4.3 Ανάλυση του Service: RequestService

Περιγραφή του Σκοπού του Service

Το **RequestService** λειτουργεί ως ενδιάμεσος για την αποστολή και λήψη HTTP αιτημάτων από το API της εφαρμογής. Διαχειρίζεται τα αιτήματα GET και POST, συμπεριλαμβάνοντας προαιρετικά διαπιστευτήρια ελέγχου ταυτότητας για ασφαλείς κλήσεις προς τον server. Το service χρησιμοποιεί το HttpClient της Angular για να διευκολύνει την επικοινωνία με το API, ενώ παρέχει έναν απλό τρόπο διαχείρισης των αποκρίσεων και των σφαλμάτων που προκύπτουν από αυτές τις κλήσεις.

Ανάλυση Μεθόδων και Λειτουργικότητας

1. **getRequest(params: any, auth?: boolean)**
   - **Λειτουργία:** Στέλνει ένα HTTP GET αίτημα προς τον server για να λάβει δεδομένα.
   - **Λεπτομέρειες Υλοποίησης:**
     - Αρχικοποιεί τα HTTP headers, προσθέτοντας το token ελέγχου ταυτότητας αν το auth είναι αληθές.
     - Χρησιμοποιεί το HttpClient για την αποστολή του αιτήματος στον server.
     - Επιστρέφει ένα Promise που επιλύεται με τα δεδομένα που λαμβάνονται ή απορρίπτεται σε περίπτωση σφάλματος.
   - **Σημασία:** Επιτρέπει την άντληση δεδομένων από το API με ασφαλή τρόπο, διατηρώντας τη δυνατότητα ελέγχου ταυτότητας για προστατευμένες περιοχές του server.

```
getRequest(params: any, auth?: boolean) {
  const headers = auth ? new HttpHeaders({ 'Authorization': 'Bearer ' + this.token }) : new HttpHeaders();

  return new Promise((resolve, reject) => {
    this.http.get<any>(environment.apiUrl + params.url, { headers }).
      subscribe((data: any) => {
        resolve(data);
      }, error => {
        reject(error);
      })
  });
};
```

Σχήμα 3.11: getRequest()

2. **postRequest(params: any, auth?: boolean)**
   - **Λειτουργία:** Στέλνει ένα HTTP POST αίτημα προς τον server για την αποστολή δεδομένων.
   - **Λεπτομέρειες Υλοποίησης:**
     - Αρχικοποιεί τα HTTP headers με το token ελέγχου ταυτότητας αν το auth είναι αληθές.
     - Χρησιμοποιεί το HttpClient για την αποστολή δεδομένων στον server.
     - Επιστρέφει ένα Promise που επιλύεται με τα δεδομένα της απάντησης ή απορρίπτεται σε περίπτωση σφάλματος.
   - **Σημασία:** Διευκολύνει την αποστολή δεδομένων προς το API, διαχειρίζεται την ασφάλεια μέσω του token ελέγχου ταυτότητας και χειρίζεται την απάντηση.

```
postRequest(params: any, auth?: boolean) {
  const headers = auth ? new HttpHeaders({ 'Authorization': 'Bearer ' + this.token }) : new HttpHeaders();

  return new Promise((resolve, reject) => {
    this.http.post<any>(environment.apiUrl + params.url, params.req, { headers })
      .subscribe(data => {
        resolve(data);
      }, error => {
        reject(error);
      });
  });
};
```

Σχήμα 3.12: postRequest()

Ρόλος και Χρησιμότητα του Service

Το **RequestService** διαδραματίζει καθοριστικό ρόλο στη διαχείριση της επικοινωνίας μεταξύ του front-end και του back-end. Καθιστά την αποστολή και λήψη δεδομένων μέσω HTTP αιτημάτων πιο απλή και ασφαλή, διαχειρίζοντας με συνέπεια τα διαπιστευτήρια και την επεξεργασία των αποκρίσεων. Η χρήση του Promise εξασφαλίζει ότι οι μέθοδοι αυτές μπορούν να χειριστούν ασύγχρονες λειτουργίες με σαφήνεια και αξιοπιστία, παρέχοντας τα απαραίτητα δεδομένα για τη λειτουργία της εφαρμογής.

3.4.4 Ανάλυση του Service: ResortService

Περιγραφή του Σκοπού του Service

Το **ResortService** είναι υπεύθυνο για τη διαχείριση και την παροχή δεδομένων σχετικά με τα χιονοδρομικά κέντρα στην εφαρμογή. Χρησιμοποιεί το RequestService για να πραγματοποιεί αιτήματα προς το API και να λαμβάνει πληροφορίες σχετικά με όλα τα χιονοδρομικά κέντρα, τα ενεργά κέντρα, τα λιφτ, το κόστος και για να στέλνει πληροφορίες κρατήσεων.

Ανάλυση Μεθόδων και Λειτουργικότητας

1. **getAllResorts()**
   - **Λειτουργία:** Αποκτά όλες τις πληροφορίες των διαθέσιμων χιονοδρομικών κέντρων.
   - **Λεπτομέρειες Υλοποίησης:**
     - Χρησιμοποιεί το RequestService για να στείλει ένα HTTP GET αίτημα με το endpoint SnowResorts.
     - Αποθηκεύει τα δεδομένα που λαμβάνονται στο allResorts και επιστρέφει αυτά τα δεδομένα.
   - **Σημασία:** Παρέχει έναν τρόπο να φορτώνονται και να αποθηκεύονται τα στοιχεία όλων των χιονοδρομικών κέντρων, τα οποία μπορούν να χρησιμοποιηθούν σε διάφορα σημεία της εφαρμογής.

```
getAllResorts() {
  return this.request.getRequest({ url: 'SnowResorts' }).then((data: any) => {
    this.allResorts = data;
    return data;
  })
}
```

Σχήμα 3.13: getAllResorts()

2. **getActiveResort(resortId: string)**
   - **Λειτουργία:** Εντοπίζει και επιστρέφει το ενεργό χιονοδρομικό κέντρο βάσει του resortId.
   - **Λεπτομέρειες Υλοποίησης:**
     - Χρησιμοποιεί το resortId για να βρει το αντίστοιχο χιονοδρομικό κέντρο στη λίστα allResorts.
     - Αποθηκεύει και επιστρέφει το ενεργό χιονοδρομικό κέντρο.
   - **Σημασία:** Παρέχει μια μέθοδο για την εύκολη ανάκτηση και χρήση των πληροφοριών ενός συγκεκριμένου χιονοδρομικού κέντρου στην εφαρμογή.

```
getActiveResort(resortId: string) {
  this.activeResort = this.allResorts.find((item: any) => item.id == resortId)
  return this.activeResort
}
```

Σχήμα 3.14: getActiveResorts()

3. **getLifts(resortId: string)**
   - **Λειτουργία:** Λαμβάνει πληροφορίες για τα λιφτ ενός συγκεκριμένου χιονοδρομικού κέντρου.
   - **Λεπτομέρειες Υλοποίησης:**
     - Δημιουργεί τα παραμέτρους για το αίτημα με το resortId και στέλνει ένα HTTP GET αίτημα μέσω του RequestService.
     - Επιστρέφει τα δεδομένα σχετικά με τα λιφτ.
   - **Σημασία:** Δίνει τη δυνατότητα να φορτώνονται πληροφορίες για τα λιφτ, χρήσιμο για την παροχή πληροφοριών στους χρήστες για την κατάσταση των λιφτ στο χιονοδρομικό κέντρο.

```
getLifts(resortId: string) {
  let params = {
    url: 'lifts/' + resortId,
  }
  return this.request.getRequest(params).then(res => res)
}
```

Σχήμα 3.15: getLifts()

4. **getCost(resortId: string)**
    - **Λειτουργία:** Λαμβάνει το κόστος επίσκεψης για ένα συγκεκριμένο χιονοδρομικό κέντρο.
    - **Λεπτομέρειες Υλοποίησης:**
        - Στέλνει ένα HTTP GET αίτημα για να λάβει το κόστος επίσκεψης, χρησιμοποιώντας το resortId.
        - Επιστρέφει το κόστος που λαμβάνεται από το API.
    - **Σημασία:** Παρέχει την τιμολογιακή πληροφορία για τους χρήστες, δίνοντάς τους τη δυνατότητα να δουν το κόστος πριν πραγματοποιήσουν μια κράτηση.

```
getCost(resortId: string) {
  let params = {
    url: 'cost/' + resortId,
  }
  return this.request.getRequest(params).then((res: any) => res[0].cost)
}
```

Σχήμα 3.16: getCost()

5. **sendBooking(resortId: string, visitors: number, cost: number)**
    - **Λειτουργία:** Αποστέλλει μια κράτηση στο σύστημα για το συγκεκριμένο χιονοδρομικό κέντρο.
    - **Λεπτομέρειες Υλοποίησης:**
        - Δημιουργεί ένα POST αίτημα με τα απαραίτητα δεδομένα (resortId, visitors, cost) και το στέλνει μέσω του RequestService.
        - Το auth ορίζεται σε true για να επιβεβαιωθεί ότι η κράτηση γίνεται με έγκυρο session.
    - **Σημασία:** Επιτρέπει στους χρήστες να πραγματοποιήσουν κράτηση για το χιονοδρομικό κέντρο μέσω της εφαρμογής, προσθέτοντας ένα σημαντικό στοιχείο διαδραστικότητας και λειτουργικότητας.

```
sendBooking(resortId: string, visitors: number, cost: number) {
  let params = {
    url: 'booking',
    req: {
      snow_resort_id: resortId,
      number_pass: visitors,
      cost: cost
    }
  };

  this.request.postRequest(params, true)
}
```

Σχήμα 3.17: sendBooking()

Ρόλος και Χρησιμότητα του Service

Το ResortService λειτουργεί ως κεντρικό σημείο διαχείρισης όλων των δεδομένων που σχετίζονται με τα χιονοδρομικά κέντρα. Μέσω των μεθόδων του, η εφαρμογή μπορεί να παρέχει στους χρήστες ενημερωμένες πληροφορίες, όπως λίστες χιονοδρομικών κέντρων, ενεργά λιφτ, και κόστη επίσκεψης. Επίσης, ενισχύει τη διαδραστικότητα, επιτρέποντας την πραγματοποίηση κρατήσεων. Αυτή η οργάνωση βοηθά στην απλοποίηση της λογικής της εφαρμογής και την κάνει πιο ευέλικτη και εύκολη στη συντήρηση.

3.4.5 Ανάλυση του Service: UserService

Περιγραφή του Σκοπού του Service

Το **UserService** είναι ένα βασικό κομμάτι της εφαρμογής που διαχειρίζεται τις λειτουργίες που σχετίζονται με τον χρήστη, όπως η ανάκτηση των στοιχείων του, η είσοδος, η εγγραφή, και η αποσύνδεση. Παρακάτω παρουσιάζονται τα κύρια χαρακτηριστικά και οι λειτουργίες του service αυτού:

Ανάλυση Μεθόδων και Λειτουργικότητας

1. **getUser()**
    - **Λειτουργία:** Λαμβάνει τις πληροφορίες του χρήστη από το API και τις αποθηκεύει τοπικά.
    - **Λεπτομέρειες Υλοποίησης:**
        - Χρησιμοποιεί το RequestService για να στείλει ένα HTTP GET αίτημα στο endpoint user. Το auth ορίζεται σε true, πράγμα που σημαίνει ότι απαιτείται εξουσιοδότηση με το token του χρήστη.
        - Όταν λάβει την απάντηση, αποθηκεύει τις πληροφορίες του χρήστη (id, name, email) στο αντικείμενο user.
        - Σε περίπτωση επιτυχίας, επιστρέφει το αντικείμενο user.
        - Σε περίπτωση αποτυχίας του αιτήματος, επιστρέφει false.
    - **Σημασία:** Παρέχει έναν τρόπο να λαμβάνονται οι πληροφορίες του χρήστη από το API και να αποθηκεύονται τοπικά, ώστε να είναι προσβάσιμες από άλλα μέρη της εφαρμογής. Αυτό είναι σημαντικό για λειτουργίες που απαιτούν τη γνώση του τρέχοντος χρήστη, όπως η εξατομίκευση της εμπειρίας χρήστη και η διαχείριση των δεδομένων του.

```
getUser() {
  return this.requestService.getRequest({ url: 'user' }, true)
    .then((resp: any) => {
      this.user = {
        id: resp.id,
        name: resp.name,
        email: resp.email
      };
      return this.user;
    })
    .catch(() => {
      return false;
    });
}
```

Σχήμα 3.18: getUser()

2. **login(email: string, password: string)**
   - **Λειτουργία:** Πραγματοποιεί τη σύνδεση του χρήστη στην εφαρμογή.
   - **Λεπτομέρειες Υλοποίησης:**
     - Στέλνει ένα POST αίτημα στο endpoint 'login' με τα διαπιστευτήρια του χρήστη (email και password).
     - Σε περίπτωση επιτυχίας, λαμβάνει και αποθηκεύει το token αυθεντικοποίησης στο localStorage και το αναθέτει στο RequestService για μελλοντικά αιτήματα που απαιτούν εξουσιοδότηση.
     - Σε περίπτωση επιτυχίας του αιτήματος, επιστρέφει true.
     - Σε περίπτωση αποτυχίας του αιτήματος, επιστρέφει false.
   - **Σημασία:** Διαχειρίζεται τη διαδικασία αυθεντικοποίησης του χρήστη, επιτρέποντας στην εφαρμογή να γνωρίζει τον τρέχοντα χρήστη και να του παρέχει πρόσβαση σε προστατευμένες λειτουργίες.

```
login(email: string, password: string) {
  let params = {
    url: 'login',
    req: {
      email: email,
      password: password
    }
  };
  return this.requestService.postRequest(params).then((res: any) => {
    this.requestService.token = res.token;
    localStorage.setItem('token', res.token);
    return true;
  }, (error: any) => {
    console.error('An error occurred:', error);
    return false;
  });
}
```

Σχήμα 3.19: login()

3. **register(name: string, email: string, password: string)**
   - **Λειτουργία:** Εγγράφει έναν νέο χρήστη στην εφαρμογή.
   - **Λεπτομέρειες Υλοποίησης:**
     - Στέλνει ένα POST αίτημα στο endpoint 'register' με τα στοιχεία του χρήστη (name, email, password).
     - Σε περίπτωση επιτυχίας του αιτήματος, επιστρέφει true.
     - Σε περίπτωση αποτυχίας του αιτήματος, επιστρέφει false.
   - **Σημασία:** Επιτρέπει τη δημιουργία νέων λογαριασμών χρηστών, παρέχοντας στην εφαρμογή τη δυνατότητα να διαχειρίζεται την εγγραφή νέων χρηστών.

```
register(name: string, email: string, password: string) {
  let params = {
    url: 'register',
    req: {
      name: name,
      email: email,
      password: password
    }
  };
  return this.requestService.postRequest(params).then(() => {
    return true;
  }, (error: any) => {
    console.error('An error occurred:', error);
    return false;
  });
}
```

Σχήμα 3.20: register()

4. **logout()**
    - **Λειτουργία:** Αποσυνδέει τον χρήστη από την εφαρμογή.
    - **Λεπτομέρειες Υλοποίησης:**
        - Στέλνει ένα POST αίτημα στο endpoint 'logout'.
        - Αφαιρεί το token αυθεντικοποίησης από το localStorage και ανανεώνει την εφαρμογή για να διαγράψει οποιαδήποτε κατάσταση του χρήστη.
    - **Σημασία:** Διαχειρίζεται την έξοδο του χρήστη από την εφαρμογή, διασφαλίζοντας την ασφάλεια και την αποσύνδεση από τις προστατευμένες περιοχές της εφαρμογής.

```
logout() {
  this.requestService.postRequest({ url: 'logout' }, true).then(() => {
    localStorage.removeItem('token');
    location.reload();
  })
}
```

Σχήμα 3.21: logout()

Ρόλος και Χρησιμότητα του Service

Το **UserService** λειτουργεί ως το κεντρικό σημείο πρόσβασης για τη διαχείριση των δεδομένων του χρήστη και των λειτουργιών αυθεντικοποίησης μέσα στην εφαρμογή. Συγκεντρώνει τις διαδικασίες εγγραφής, σύνδεσης, λήψης πληροφοριών χρήστη, και αποσύνδεσης σε ένα σημείο, διευκολύνοντας την αλληλεπίδραση με τον χρήστη. Αυτό επιτρέπει σε άλλα μέρη της εφαρμογής να έχουν πρόσβαση

στα δεδομένα του χρήστη με ασφαλή και αποδοτικό τρόπο, μειώνοντας τις καθυστερήσεις και την πολυπλοκότητα.

3.5 Η εφαρμογή στο GitHub

Η υλοποίηση της συγκεκριμένης εφορμογής βρίσκεται σε ένα repository στο GitHub. Μέσα στο repository βρίσκονται όλα τα απαραίτητα αρχεία της εφαρμογής όπως τον φάκελο /src και τα αρχεία παραμετροποίησης (configuration files). Υπάρχει επίσης ένας φάκελος /.github/workflows οποίος έχει μέσα ένα αρχείο με κατάληξη .yml το οποίο αρχείο είναι υπεύθυνο για το GitHub Action. Σε αυτό το αρχείο υπάρχει μία σειρά βημάτων/εντολών που πρέπει να εκτελεστούν αλλά και μία εντολή ωστέ να γίνεται αυτόματα όλη η διαδικασία σε περίπτωση που γίνει κάτι που θα ορίσουμε. Στην περίπτωση μας, μόλις γίνει push στο main branch τότε τρέχει τις εντολές και έτσι η εφαρμογή ‘ανεβαίνει’ στον server. Τέλος υπάρχει και ένα αρχείο .gitignore όπου μέσα σε αυτό προσθέτουμε ονόματα από αρχεία και φακέλους τα οποία δεν θέλουμε να ανέβουν στο GitHub.

Κεφάλαιο 4ο: Παρουσίαση της Εφαρμογής

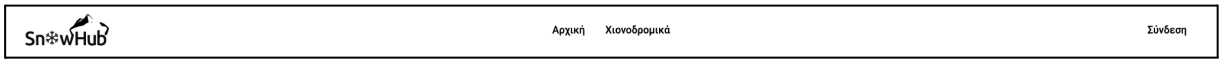
Σε αυτό το κεφάλαιο θα γίνει παρουσίαση της εφαρμογής κατηγοριοποιημένη με βάση της σελίδες που βλέπει ο χρήστης. Στο βασικό template της εφαρμογής app.component.html βρίσκεται ο selector του NavbarComponent (app-navbar) έτσι ώστε η γραμμή πλοήγησης να εμφανίζεται σε όλες τις σελίδες. Κάτω από αυτό υπάρχει το directive της Angular, <router-outlet>, που χρησιμοποιείται για τον καθορισμό του σημείου στο HTML template όπου θα εμφανίζεται το περιεχόμενο των ενεργών routes. Με άλλα λόγια, εκεί "φιλοξενεί" τα components που φορτώνονται δυναμικά καθώς ο χρήστης περιηγείται στην εφαρμογή και αλλάζει σελίδες.

```
<div class="wrapper">
  <app-navbar></app-navbar>

  <main>
    <router-outlet *ngIf="flag"></router-outlet>
  </main>
</div>
```

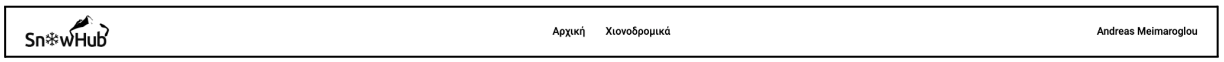
Σχήμα 4.1: app.component.ts

4.1 Γραμμή Πλοήγησης

Η γραμμή πλοήγησης βρίσκεται συνέχεια στο επάνω μέρος της οθόνης. Αποτελείται από το λογότυπο της εφαρμογής το οποίο βρίσκεται στα αριστερά επάνω της οθόνης και στην περίπτωση που ο χρήστης κάνει κλικ επάνω του τότε τον πηγαίνει στην αρχική σελίδα. Έπειτα στο μέσο της υπάρχουν δύο κουμπιά πλοήγησης ώστε να ανακατευθύνουν τον χρήστη στην συγκεκριμένη σελίδα που επέλεξε (πχ. Αρχική, Χιονοδρομικά). Τέλος στα δεξιά εμφανίζεται ένα κουμπί όπου στην περίπτωση που ο χρήστης δεν έχει πραγματοποιήσει σύνδεση τότε εμφανίζεται το κουμπί ‘Σύνδεση’ αλλιώς αν ο χρήστης έχει συνδεθεί τότε εμφανίζει το όνομα του χρήστη.



Σχήμα 4.2: Ο χρήστης δεν έχει πραγματοποιήσει σύνδεση

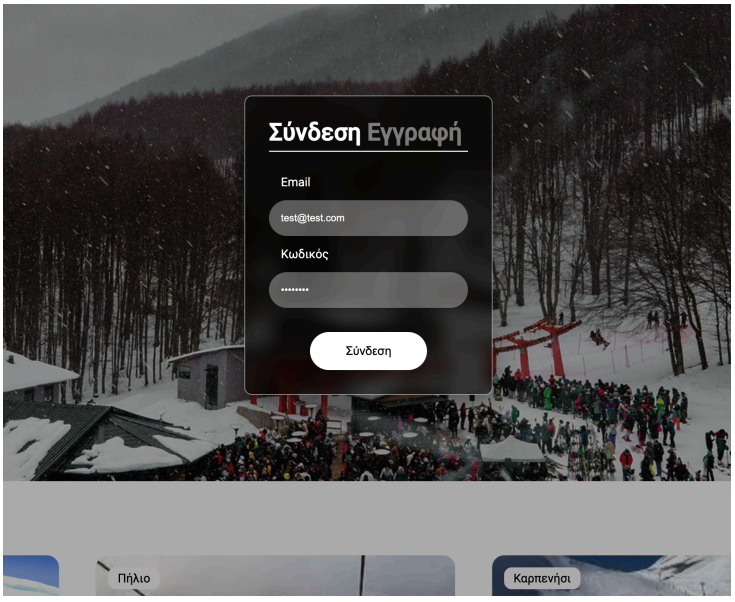


Σχήμα 4.3: Ο χρήστης έχει πραγματοποιήσει σύνδεση

Όταν ο χρήστης είναι συνδεδεμένος και κάνει hover το ποντίκι του επάνω στο όνομά του τότε εμφανίζεται κάτω από αυτό ένα κουμπί αποσύνδεσης. Στην περίπτωση πάλι που δεν έχει συνδεθεί ο χρήστης και πατήσει το κουμπί ‘Σύνδεση’ τότε εμφανίζεται στην οθόνη το LoginDialog.

4.2 Login Dialog

Το LoginDialog είναι ένα component όπου εμφανίζεται στην μέση της οθόνης. Αυτό το component αποτελείται από δύο επιλογές για σύνδεση ή εγγραφή. Στην περίπτωση που ο χρήστης διαλέξει να κάνει σύνδεση, εμφανίζεται μία φόρμα που αποτελείται από δύο πεδία εισαγωγής κειμένου, E-mail και κωδικό πρόσβασης, και το κουμπί ‘Σύνδεση’. Από την άλλη αν διαλέξει να κάνει εγγραφή τότε η φόρμα έχει τα δύο προηγούμενα πεδία και ένα ακόμα για το ονοματεπώνυμο. Αντί για το κουμπί ‘Σύνδεση’ υπάρχει το κουμπί ‘Εγγραφή’.



Σχήμα 4.4: LoginDialog

Η εναλλαγή στο κουμπί γίνεται με την βοήθεια του directive της Angular *ngIf και με μία boolean μεταβλητή isRegister εμφανίζει το κατάλληλο. Μόλις ο χρήστης κάνει κλικ στο κουμπί ‘Σύνδεση’ ή ‘Εγγραφή’ τότε γίνεται ένα POST Request στον server και αν η απάντησει από τον server είναι θετική τότε το LoginDialog κλείνει.

```
<div class="form-button">
    <button *ngIf="!isRegister" (click)="login()">Σύνδεση</button>
    <button *ngIf="isRegister" (click)="register()">Εγγραφή</button>
</div>
```

Σχήμα 4.5: Χρήση *ngIf directive στο LoginDialog

4.2.1 Ανάλυση Λειτουργικότητας του Login Dialog

Σε αυτό το υποκεφάλαιο, παρουσιάζεται η ανάλυση του κώδικα που αφορά τη λειτουργικότητα του dialog σύνδεσης και εγγραφής χρηστών στην εφαρμογή. Ο κώδικας που εμφανίζεται παρακάτω είναι υπεύθυνος για τη διαχείριση της σύνδεσης χρηστών μέσω ενός φόρμας σύνδεσης, καθώς και για την εγγραφή νέων χρηστών.

Ανάλυση του κώδικα:

- **Δημιουργία Φόρμας Σύνδεσης και Εγγραφής:** Ο constructor του component δημιουργεί μια reactive φόρμα με τη χρήση του FormBuilder. Η φόρμα περιέχει πεδία για το όνομα, το email και τον κωδικό πρόσβασης. Το πεδίο "name" είναι υποχρεωτικό μόνο όταν ο χρήστης επιλέγει την εγγραφή (μέσω της μεταβλητής isRegister).

```
constructor(public dialogRef: MatDialogRef<LoginDialog>, private fb: FormBuilder, private userService: UserService,
    @Inject(MAT_DIALOG_DATA) public data: any) {
    this.loginForm = this.fb.group({
        name: new FormControl('', [this.isRegister ? Validators.required : Validators.nullValidator]),
        email: new FormControl('', [Validators.required, Validators.email]),
        password: new FormControl('', [Validators.required, Validators.minLength(8)])
    });
}
```

Σχήμα 4.6: Login Dialog constructor

- **Διαχείριση Λειτουργίας Σύνδεσης**: Η μέθοδος login() εκτελείται όταν ο χρήστης προσπαθεί να συνδεθεί. Αρχικά, ελέγχεται η εγκυρότητα της φόρμας και στη συνέχεια καλείται η μέθοδος login() του UserService, η οποία στέλνει τα δεδομένα σύνδεσης στον server. Αν η διαδικασία σύνδεσης είναι επιτυχής, το dialog κλείνει, ενώ σε περίπτωση αποτυχίας ενεργοποιείται η μεταβλητή wrongCredentials, η οποία μπορεί να χρησιμοποιηθεί για να εμφανιστεί σχετικό μήνυμα στον χρήστη.
- **Διαχείριση Λειτουργίας Εγγραφής**: Η μέθοδος register() λειτουργεί παρόμοια με τη μέθοδο σύνδεσης. Αν η φόρμα είναι έγκυρη, καλείται η μέθοδος register() του UserService, η οποία στέλνει τα δεδομένα εγγραφής στον server. Αν η εγγραφή ολοκληρωθεί με επιτυχία, το dialog επίσης κλείνει.

```
login() {
    let email = this.loginForm.value.email;
    let password = this.loginForm.value.password;
    if (this.loginForm.valid) {
        this.userService.login(email, password).then((res: any) => {
            if (res) {
                this.dialogRef.close(true);
            } else {
                this.wrongCredentials = true;
            }
        });
    }
}

register() {
    let email = this.loginForm.value.email;
    let password = this.loginForm.value.password;
    let name = this.loginForm.value.name;
    if (this.loginForm.valid) {
        this.userService.register(name, email, password).then((res: any) => {
            if (res) {
                this.dialogRef.close(true);
            }
        });
    }
}
```

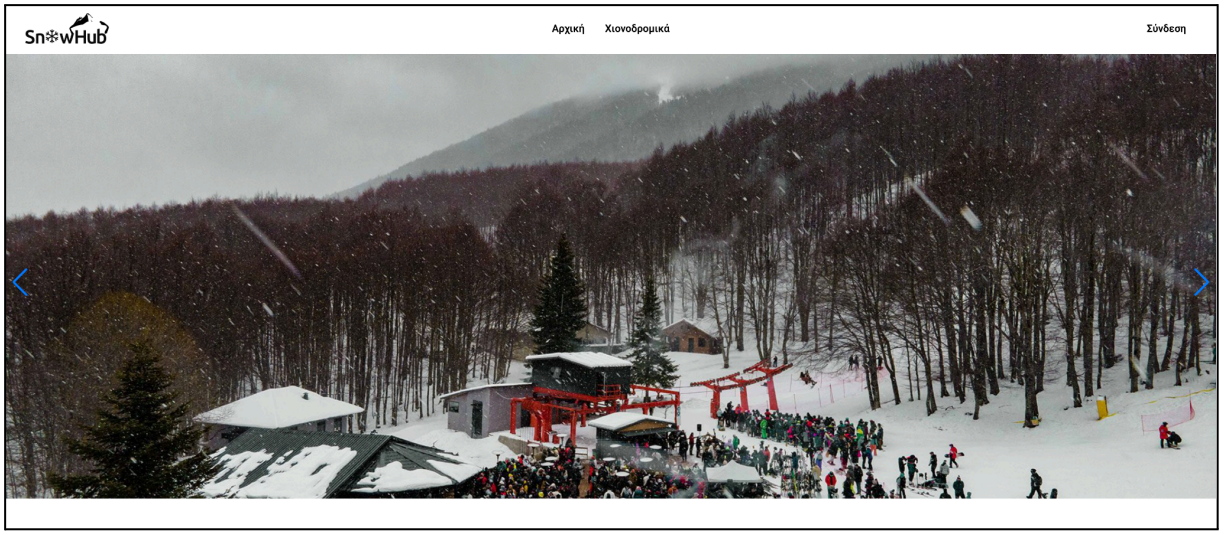
Σχήμα 4.7: Μέθοδοι σύνδεσης και εγγραφής στο Login Dialog

- **Ευελιξία Dialog**: Το dialog δίνει τη δυνατότητα στον χρήστη να επιλέξει αν θέλει να συνδεθεί ή να εγγραφεί, προσφέροντας ευέλικτες επιλογές μέσω του διακόπτη isRegister. Ο κώδικας παρέχει μια απλή και καθαρή διεπαφή για τη διαχείριση χρηστών, καλύπτοντας τις βασικές ανάγκες σύνδεσης και εγγραφής στην εφαρμογή.

Ο κώδικας αυτός επιτρέπει στους χρήστες να συνδεθούν στην εφαρμογή με ευκολία και ασφάλεια, ενώ παρέχει και τη δυνατότητα εγγραφής νέων χρηστών, ενισχύοντας τη λειτουργικότητα της εφαρμογής και βελτιώνοντας την εμπειρία χρήστη.

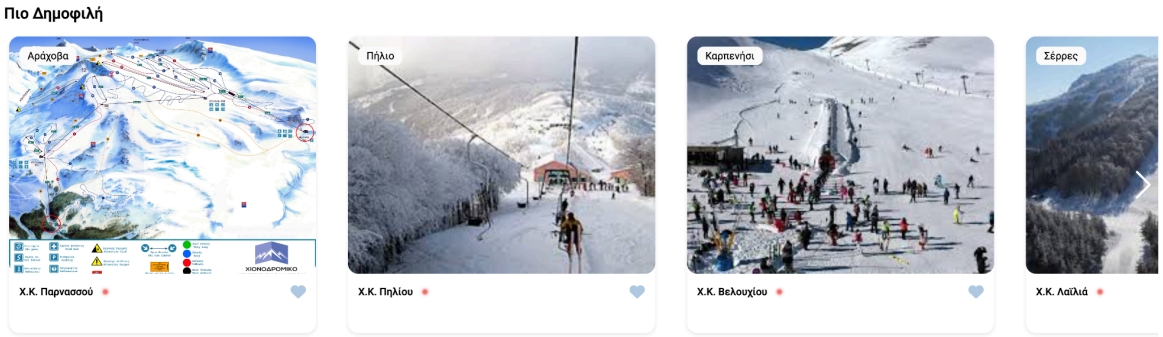
4.3 Αρχική Σελίδα

Η αρχική σελίδα αποτελείται από έναν hero slider ο οποίος πιάνει το 100% του μήκους της οθόνης και εμφανίζει κάποιες φωτογραφίες από τα χιονοδρομικά όπου αλλάζουν μόνες τους ανά κάποια δευτερόλεπτα. Έχει και ο χρήστης την δυνατότητα να εναλλάσει της φωτογραφίες είτε με κλικ στα δύο κουμπιά που βρίσκονται αριστερά και δεξιά επάνω στον slider, είτε με το να σύρει τις εικόνες με το ποντίκι.



Σχήμα 4.8: Hero Slider

Έπειτα κάτω από τον hero slider υπάρχει ένας ακόμα slider με τα πιο δημοφιλή χιονοδρομικά στην Ελλάδα. Όπως και ο hero slider έτσι και αυτός έχει την δυνατότητα πλοήγησης και με τα δύο κουμπιά αλλά και με σύρσιμο με το ποντίκι. Το κάθε χιονοδρομικό εμφανίζεται σε μορφή κάρτας με μία εικόνα, την τοποθεσία, το όνομα, μία κόκκινη ή πράσινη κουκίδα ανάλογα με το αν είναι ανοιχτο το χιονοδρομικό ή οχι και ένα κουμπί 'Μου αρέσει'. Το κουμπί 'Μου αρέσει' είναι σε μορφή καρδίας η οποία αλλάζει χρώμα ανάλογα με την επιλογή του χρήστη. Ο χρήστης έχει δυνατότητα να πατήσει το κουμπί μόνο αν είναι συνδεδεμένος, σε διαφορετική περίπτωση δεν μπορεί. Όταν ο χρήστης κάνει κλικ έπανω στην κάρτα τότε η εφαρμογή τον πηγαίνει στην σελίδα του συγκεκριμένου χιονοδρομικού.



Σχήμα 4.9: Slider με δημοφιλή χιονοδρομικά

Επίσης το Swiper, η βιβλιοθήκη που μας βοηθάει με τους slider, μας δίνει την δυνατότητα να ορίσουμε κάποιες επιλογές για το πως θέλουμε να λειτουργεί ο κάθε slider. Μπορούμε να ορίσουμε το πόσα slides θα εμφανίζει ανά μέγεθος οθόνης, το κενό που θα έχουν μεταξύ τους τα slides, την αυτόματη κύλιση κ.α.

```
config: SwiperOptions = {
  slidesPerView: 1,
  spaceBetween: 50,
  navigation: true,
  breakpoints: {
    640: {
      slidesPerView: 1.5,
      spaceBetween: 20,
    },
    768: {
      slidesPerView: 2.5,
      spaceBetween: 40,
    },
    1024: {
      slidesPerView: 3.5,
      spaceBetween: 50,
    },
  },
};
```

Σχήμα 4.10: Configuration για slider

Τέλος κάτω από τον slider με τα δημοφιλή χιονοδρομικά υπάρχει ένας διαδραστικός χάρτης ώστε να μπορεί ο χρήστης άμεσα να βλέπει τον καιρό εκεί που τον ενδιαφέρει.

4.3.1 Ανάλυση Λειτουργικότητας της Αρχικής Σελίδας

Σε αυτό το υποκεφάλαιο, παρουσιάζεται η ανάλυση του κώδικα που αφορά τη λειτουργικότητα της Αρχικής Σελίδας της εφαρμογής. Ο κώδικας που εμφανίζεται παρακάτω είναι υπεύθυνος για την εμφάνιση των χιονοδρομικών κέντρων στην κεντρική σελίδα καθώς και τη ρύθμιση των εικόνων σε έναν δυναμικό carousel (swiper).

Ανάλυση του κώδικα:

- **Προβολή Χιονοδρομικών Κέντρων**: Η μεταβλητή resorts αποθηκεύει μια λίστα με τα χιονοδρομικά κέντρα που θα εμφανίζονται στην αρχική σελίδα. Στην ngOnInit μέθοδο, φορτώνεται ένα υποσύνολο (5 κέντρα) από όλα τα χιονοδρομικά κέντρα που παρέχονται από το ResortService.
- **Διαμόρφωση του Carousel Εικόνων**: Η εφαρμογή χρησιμοποιεί την βιβλιοθήκη Swiper για τη δημιουργία ενός carousel εικόνων. Οι ρυθμίσεις configImages και config καθορίζουν πώς θα προβάλλονται οι εικόνες και οι πληροφορίες των χιονοδρομικών κέντρων σε διαφορετικά μεγέθη οθόνης. Το slidesPerView καθορίζει τον αριθμό των slides που εμφανίζονται ανάλογα με την ανάλυση της οθόνης, ενώ η δυνατότητα autoplay επιτρέπει την αυτόματη κύλιση των slides.

```
export class HomeComponent {
  resorts: any = [];

  configImages: SwiperOptions = {…
  };

  config: SwiperOptions = {…
  };

  constructor(private resortService: ResortService) { }

  ngOnInit() {
    this.resorts = this.resortService.allResorts.slice(0, 5)
  }
}
```

Σχήμα 4.11: HomeComponent

Ο κώδικας αυτός προσφέρει μία ευχάριστη και λειτουργική εμπειρία πλοήγησης, επιτρέποντας στους χρήστες να βλέπουν τις διαθέσιμες επιλογές χιονοδρομικών κέντρων με οπτική άνεση, ενώ ταυτόχρονα προσαρμόζεται δυναμικά σε διάφορες αναλύσεις συσκευών.

4.4 Όλα τα Χιονοδρομικά

Η επόμενη σελίδα είναι μία λίστα με όλα τα χιονοδρομικά. Κάθε χιονοδρομικό εμφανίζεται πάλι με μορφή κάρτας, όπως ακριβώς και στην αρχική σελίδα. Η εμφάνιση τους γίνεται με την βοήθεια του directive της Angular *ngFor το οποίο παίρνει τον πίνακα με όλα τα χιονοδρομικά και εμφανίζει μία κάρτα για το κάθε ένα.

```
<div class="wrapper">
    <div class="resorts">
        <app-card class="resort-card" [data]="resort" *ngFor="let resort of allResorts"></app-card>
    </div>
</div>
```

Σχήμα 4.12: Χρήση *ngFor για εμφάνιση χιονοδρομικών

4.4.1 Ανάλυση Λειτουργικότητας της Σελίδας “Όλα τα Χιονοδρομικά”

Σε αυτό το υποκεφάλαιο, παρουσιάζεται η ανάλυση του κώδικα που αφορά τη λειτουργικότητα της σελίδας "Όλα τα Χιονοδρομικά" της εφαρμογής. Ο κώδικας παρακάτω είναι υπεύθυνος για την εμφάνιση όλων των διαθέσιμων χιονοδρομικών κέντρων που προσφέρονται στους χρήστες μέσω της εφαρμογής.

Ανάλυση του κώδικα:

- **Προβολή Όλων των Χιονοδρομικών Κέντρων:** Η μεταβλητή allResorts αποθηκεύει όλα τα χιονοδρομικά κέντρα που είναι διαθέσιμα στην εφαρμογή. Η μέθοδος ngOnInit είναι

υπεύθυνη για τη φόρτωση αυτών των δεδομένων κατά την αρχική φόρτωση της σελίδας, χρησιμοποιώντας το ResortService για την ανάκτηση των χιονοδρομικών.

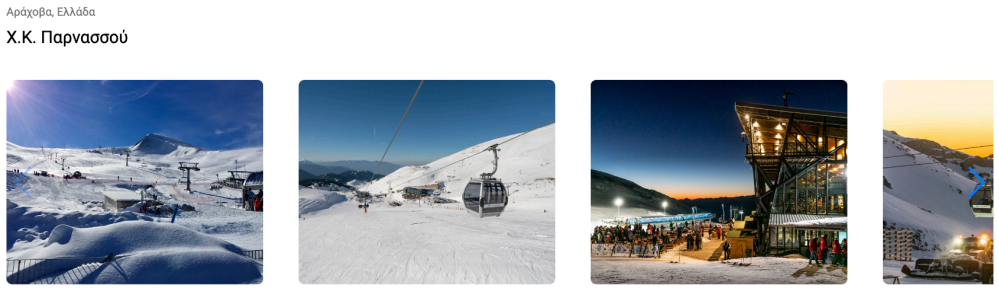
```
export class ResortsComponent {
  allResorts: any;

  constructor(private resort: ResortService) { }

  ngOnInit() {
    this.allResorts = this.resort.allResorts;
  }
}
```

Σχήμα 4.13: ResortsComponent

Ο κώδικας αυτός επιτρέπει την άμεση πρόσβαση των χρηστών σε όλα τα διαθέσιμα χιονοδρομικά κέντρα και εξασφαλίζει ότι τα δεδομένα είναι διαθέσιμα από την αρχή της πλοήγησης, παρέχοντας μια απρόσκοπτη εμπειρία χρήστη.

4.5 Σελίδα Χιονοδρομικού

Τέλος, έχουμε την σελίδα του χιονοδρομικού με περιεχόμενο που αφορά μόνο το συγκεκριμένο χιονοδρομικό που έκανε κλικ ο χρήστης. Αρχικά η σελίδα αποτελείται από την τοποθεσία και το όνομα του χιονοδρομικού. Κάτω από αυτά ύπαρχει ένας slider με φωτογραφίες του συγκεκριμένου χιονοδρομικού. Όπως και οι υπόλοιποι sliders που παρουσιάστηκαν έτσι και αυτός έχει την δυνατότητα πλοήγησης και με τα δύο κουμπιά αλλά και με σύρσιμο με το ποντίκι.

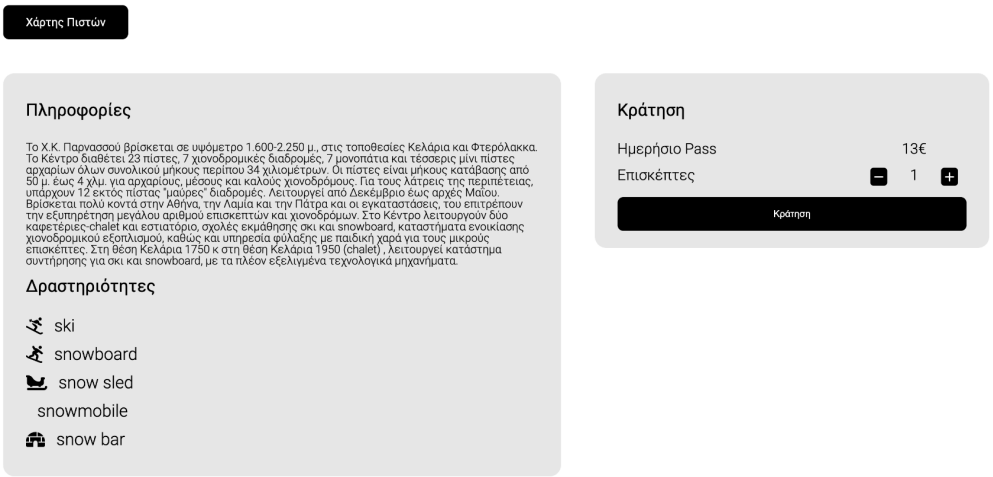


Σχήμα 4.14: Τίτλος, τοποθεσία και slider φωτογραφιών

Ταυτόχρονα εάν ο χρήστης κάνει κλικ επάνω σε μία από τις φωτογραφίες τότε ανοίγει την συγκεκριμένη φωτογραφία σε ένα lightbox gallery όπου ο χρήστης μπορεί να περιηγηθεί ταυτόχρονα και στις υπόλοιπες εικόνες είτε πάλι με τα δύο κουμπιά είτε με σύρσιμο του ποντικιού. Μόλις ο χρήστης κάνει κλικ έξω από την εικόνα ή στο κουμπί “X” τότε το lightbox gallery κλείνει και επιστρέφει στη σελίδα του χιονοδρομικού.

Σχήμα 4.15: Lightbox Gallery

Κάτω από τον slider με τις εικόνες βρίσκεται ένα κουμπί “Χάρτης Πιστών” το οποίο μόλις ο χρήστης το πατήσει του ανοίγει μία είκονα με έναν χάρτη του χιονοδρομικού και όλες τις πίστες. Αμέσως μετά βρίσκεται ένα πλαίσιο με τις πληροφορίες αλλά και τις διαθέσιμες δραστηριότητες που είναι διαθέσιμες στο χιονοδρομικό. Δίπλα σε αυτό υπάρχει ένα πλαίσιο για να μπορεί ο χρήστης να κάνει κράτηση. Σε αυτό το πλαίσιο εμφανίζεται η τιμή του ημερήσιου εισιτηρίου και ακριβώς από κάτω τον αριθμό των επισκεπτών που θα ήθελε ο χρήστης να κάνει κράτηση με δυνατότητα αυξομείωσης αυτής της τιμής.



Σχήμα 4.16: Χάρτης πιστών, πληροφορίες και δυνατότητα κράτησης

Μετά υπάρχει ένα πλαίσιο με όλες τις πίστες του χιονοδρομικού και ένα bullet χρωματισμένο ανάλογα με την δυσκολία της κάθε πίστας. Με πράσινο χρώμα είναι η πολύ εύκολες πίστες, με μπλε οι εύκολες, κόκκινο οι δύσκολες και με μαύρο οι πολύ δύσκολες. Ακριβώς μετά από αυτό υπάρχει ένα τελευταίο πλαίσιο με όλους τους αναβατήρες του χιονοδρομικού και ένα bullet χρωματισμένο ανάλογα με την διαθεσιμότητα του. Με κόκκινο χρώμα αποικονίζονται οι εκτός λειτουργίας αναβατήρες και με πράσινο οι ανοιχτοί στο κοινό αναβατήρες.

Σχήμα 4.17: Πίστες και αναβατήρες χιονοδρομικού

4.5.1 Ανάλυση Λειτουργικότητας της Σελίδας Χιονοδρομικού

Σε αυτό το υποκεφάλαιο, παρουσιάζεται η ανάλυση του κώδικα που αφορά τη λειτουργικότητα της σελίδας "Χιονοδρομικού". Ο κώδικας που εμφανίζεται παρακάτω είναι υπεύθυνος για την εμφάνιση των λεπτομερειών ενός συγκεκριμένου χιονοδρομικού κέντρου, συμπεριλαμβανομένων των εικόνων, του κόστους, των διαθέσιμων αναβατήρων, και τη δυνατότητα κράτησης.

Ανάλυση του κώδικα:

- **Ανάκτηση Πληροφοριών Χιονοδρομικού Κέντρου:** Η μεταβλητή resortId αποθηκεύει το αναγνωριστικό του χιονοδρομικού που λαμβάνεται από τη διεύθυνση URL. Στη μέθοδο ngOnInit, γίνεται χρήση του ResortService για την ανάκτηση των πληροφοριών του χιονοδρομικού, του κόστους εισόδου και των διαθέσιμων αναβατήρων. Οι εικόνες του χιονοδρομικού κέντρου φιλτράρονται, με έμφαση στην εμφάνιση των εικόνων που έχουν την ετικέτα "view", ενώ ο χάρτης των πιστών ανακτάται μέσω της εικόνας με την ετικέτα "map".

```
ngOnInit() {
  this.resort = this.resortService.getActiveResort(this.resortId)
  this.cost = this.resortService.getCost(this.resortId);
  this.lifts = this.resortService.getLifts(this.resortId);
  this.images = this.resort.images.filter((image: any) => image.caption === 'view');
  this.slopesMap = this.resort.images.find((image: any) => image.caption === 'map').image_url;
}
```

Σχήμα 4.18: Ανάκτηση Πληροφοριών Χιονοδρομικού Κέντρου

- **Δυναμική Διαχείριση Εικόνων**: Η βιβλιοθήκη Swiper χρησιμοποιείται για την προβολή των εικόνων του χιονοδρομικού με δυνατότητα διαχείρισης διαφορετικών διαστάσεων οθόνης. Επιπλέον, η μέθοδος ngAfterViewInit υλοποιεί το PhotoSwipeLightbox, επιτρέποντας την εμφάνιση των εικόνων σε πλήρη οθόνη με ευέλικτη πλοήγηση.

```
ngAfterViewInit() {
  // Get the images data before initializing PhotoSwipe
  const items = this.getImagesData();

  // Initialize PhotoSwipeLightbox with the items
  const lightbox = new PhotoSwipeLightbox({
    gallery: '#gallery--responsive-images',
    children: 'a',
    pswpModule: PhotoSwipe,
    dataSource: items // Pass the image data directly here
  });

  lightbox.init();
}
```

Σχήμα 4.19: PhotoSwipe

- **Διαχείριση Κράτησης Επισκεπτών**: Η μέθοδος changeVisitors επιτρέπει την προσθήκη ή την αφαίρεση του αριθμού επισκεπτών, ενώ η sendBooking καλεί το ResortService για να καταχωρήσει την κράτηση για το συγκεκριμένο χιονοδρομικό κέντρο.

```
changeVisitors(type: string) {
  if (type === 'add' && this.visitors < 8) {
    this.visitors += 1;
  } else if (type === 'remove' && this.visitors > 1) {
    this.visitors -= 1;
  }
}

sendBooking() {
  this.resortService.sendBooking(this.resortId, this.visitors, this.cost);
}
```

Σχήμα 4.20: Διαχείριση Κράτησης Επισκεπτών

Ο κώδικας αυτός επιτρέπει στους χρήστες να δουν αναλυτικές πληροφορίες για κάθε χιονοδρομικό κέντρο, να πλοηγηθούν μέσω εικόνων και να πραγματοποιήσουν κράτηση ανάλογα με τον αριθμό των επισκεπτών, προσφέροντας μια ολοκληρωμένη και διαδραστική εμπειρία χρήστη.

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Εξέλιξη

5.1 Συμπεράσματα της Εργασίας

Η εφαρμογή για τα χιονοδρομικά κέντρα, όπως υλοποιήθηκε, αποτελεί μια ολοκληρωμένη λύση για την παροχή πληροφοριών και την ενίσχυση της εμπειρίας των χρηστών κατά την επίσκεψή τους στα χιονοδρομικά κέντρα στην Ελλάδα. Μέσω της εφαρμογής, οι χρήστες μπορούν να βλέπουν

ενημερωμένα δεδομένα για τα χιονοδρομικά κέντρα, όπως πληροφορίες για τις πίστες, τις καιρικές συνθήκες, και τη διαθεσιμότητα των αναβατήρων, καθώς και να κάνουν κρατήσεις εισιτηρίων.

Η χρήση του Angular framework επέτρεψε τη δημιουργία μιας δυναμικής και διαδραστικής web εφαρμογής, με αρχιτεκτονική MVVM που διασφαλίζει τη διατήρηση της απόδοσης και τη δυνατότητα μελλοντικών επεκτάσεων. Η εφαρμογή είναι σχεδιασμένη για να προσφέρει ευχρηστία και λειτουργικότητα, με μοντέρνο responsive design που προσαρμόζεται στις ανάγκες του χρήστη.

Η ανάπτυξη της εφαρμογής ήταν μια πολύτιμη διαδικασία που μου πρόσφερε σημαντική εμπειρία στον προγραμματισμό και την οργάνωση του κώδικα. Μέσα από αυτήν τη διαδικασία, κατάφερα να βελτιώσω τις δεξιότητές μου στη διαχείριση έργων λογισμικού, στον καθορισμό προδιαγραφών και στη διαχείριση του χρόνου, μαθήματα που θα με συνοδεύουν σε μελλοντικά έργα.

5.2 Δυνατότητες Μελλοντικών Επεκτάσεων

Για τις μελλοντικές επεκτάσεις της εφαρμογής, υπάρχουν αρκετές δυνατότητες που θα μπορούσαν να βελτιώσουν περαιτέρω την εμπειρία των χρηστών και να προσφέρουν νέες λειτουργίες. Αρχικά, θα μπορούσε να προστεθεί ένα σύστημα ειδοποιήσεων (notifications), το οποίο θα ενημερώνει τους χρήστες για σημαντικές αλλαγές στις συνθήκες των χιονοδρομικών κέντρων, όπως ο καιρός ή η διαθεσιμότητα των αναβατήρων.

Επιπλέον, θα μπορούσε να ενσωματωθεί ένα σύστημα αξιολόγησης και σχολίων, όπου οι χρήστες θα μπορούν να αξιολογούν τα χιονοδρομικά κέντρα και να μοιράζονται την εμπειρία τους με άλλους επισκέπτες. Αυτό θα προσέθετε έναν διαδραστικό χαρακτήρα στην εφαρμογή και θα ενίσχυε την αίσθηση της κοινότητας. Τέλος, η ενσωμάτωση ενός συστήματος πληρωμών μέσα από την εφαρμογή θα μπορούσε να βελτιώσει σημαντικά τη διαδικασία κρατήσεων και την εξυπηρέτηση των χρηστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Angular Documentation. (n.d.). [Online]. Available: https://angular.dev

[2] TypeScript Documentation. (n.d.). *The TypeScript Handbook*. [Online]. Available: https://www.typescriptlang.org/docs

[3] Less.js. (n.d.). *The LESS Documentation*. [Online]. Available: http://lesscss.org

[4] RxJS Documentation. (n.d.). *Reactive Extensions Library for JavaScript*. [Online]. Available: https://rxjs.dev

[5] Angular Material. (n.d.). *Material Design Components for Angular*. [Online]. Available: https://material.angular.io

[6] Swiper. (n.d.). *Swiper - The Most Modern Mobile Touch Slider*. [Online]. Available: https://swiperjs.com

[7] PhotoSwipe. (n.d.). *JavaScript image gallery and lightbox*. [Online]. Available: https://photoswipe.com

[8] Microsoft. (n.d.). *Visual Studio Code Documentation*. [Online]. Available: https://code.visualstudio.com/docs

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

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