ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Πληροφοριακό σύστημα για επιβάτες αεροδρομίων σε πραγματικό χρόνο»

Φοιτητής

Όνομα: Σοφία Σπάρταλη (ele518137)

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2025

Πληροφοριακό σύστημα για επιβάτες αεροδρομίων σε πραγματικό χρόνο

Κωδικός: 25147

Φοιτητής: Σοφία Σπάρταλη (ele518137)

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 06-03-2025

Ημερομηνία περάτωσης Π.Ε. 05-08-2025

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

Περίληψη

Η παρούσα εργασία πραγματεύεται την ανάπτυξη μιας ολοκληρωμένης διαδικτυακής εφαρμογής για το αεροδρόμιο «Μακεδονία» της Θεσσαλονίκης· στόχος είναι η παροχή ενός σύγχρονου περιβάλλοντος πληροφόρησης και αλληλεπίδρασης προς επισκέπτες, επιβάτες και συνεργαζόμενους φορείς. Η υλοποίηση στηρίζεται σε αρχιτεκτονική client–server και συνδυάζει backend σε Python με χρήση του πλαισίου Flask, καθώς και frontend σε vanilla JavaScript με αρθρωτή σχεδίαση και δυναμικά στοιχεία διεπαφής. Η επικοινωνία μεταξύ των δύο τμημάτων πραγματοποιείται μέσω HTTP(S), ακολουθώντας το μοντέλο αιτήματος/απάντησης· επιπλέον, η πρόσβαση ρυθμίζεται με βάση τον ρόλο χρήστη (διαχειριστής, κατάστημα, απλός χρήστης) μέσω συστήματος συνεδριών με cookies.

Η εφαρμογή παρέχει αναλυτικές πληροφορίες για υποδομές και υπηρεσίες του αεροδρομίου, όπως νοσοκομεία, πρεσβείες, συγκοινωνίες, αεροπορικές εταιρείες και πτήσεις· περιλαμβάνει ακόμη παρουσίαση των καταστημάτων και των προϊόντων τους, με δυνατότητα εφαρμογής συστήματος εκπτώσεων. Παράλληλα, οι απλοί χρήστες έχουν τη δυνατότητα να λαμβάνουν ζωντανές ειδοποιήσεις για επιλεγμένες πτήσεις. Οι επιχειρήσεις του αεροδρομίου μπορούν να ενημερώνουν και να διαχειρίζονται σε πραγματικό χρόνο τα εμπορεύματά τους. Η διαχείριση δεδομένων υποστηρίζεται από CRUD API και αποθηκεύεται σε βάση SQLite, γεγονός που εξασφαλίζει ευελιξία και επεκτασιμότητα.

Ιδιαίτερη βαρύτητα δίνεται επίσης στη συλλογή και ενημέρωση δεδομένων: η βιβλιοθήκη Selenium αξιοποιείται για την αυτόματη άντληση πληροφοριών σχετικά με αφίξεις, αναχωρήσεις και καιρικές συνθήκες· η ταυτόχρονη λειτουργία του διακομιστή και της διαδικασίας συλλογής δεδομένων καθίσταται εφικτή μέσω παράλληλου προγραμματισμού με χρήση της βιβλιοθήκης threading. Με αυτόν τον τρόπο, η εφαρμογή εξασφαλίζει διαρκή ενημέρωση, αδιάλειπτη λειτουργία και υψηλό επίπεδο αξιοπιστίας.

Συνολικά, το έργο ενσωματώνει τεχνολογίες αιχμής, θέτοντας στο επίκεντρο την ασφάλεια, τη φιλικότητα προς τον τελικό χρήστη και την ομαλή συνεργασία όλων των εμπλεκομένων μερών. Η συμβολή του έγκειται τόσο στην πρακτική βελτίωση της εμπειρίας ταξιδιού και κατανάλωσης υπηρεσιών όσο και στην ανάδειξη ενός ευέλικτου και προσαρμόσιμου μοντέλου ανάπτυξης διαδικτυακών εφαρμογών για αντίστοιχες αεροπορικές ή συγκοινωνιακές υποδομές.

Λέξεις-Κλειδιά: Διαδικτυακή εφαρμογή, Αεροδρόμιο Μακεδονία, σύστημα διαχείρισης συνεδρίας, αρθρωτή σχεδίαση, CRUD API, Python, JavaScript, modular design, σύστημα ειδοποιήσεων, backend, frontend, διακομιστής, Flask, Selenium, SQLite, threading, συλλογή δεδομένων.

«Web Application for Makedonia Airport»

Abstract

This thesis addresses the development of a comprehensive web application for Thessaloniki International Airport “Makedonia”; its aim is to provide a modern information and interaction environment for visitors, passengers, and affiliated stakeholders. The implementation is based on a client–server architecture and combines a Python backend developed with the Flask framework with a modular frontend in vanilla JavaScript, enriched with dynamic interface components. Communication between the two parts takes place through HTTP(S), following the request/response model; furthermore, access control is role-based (administrator, store, user) and is managed via a session cookie system.

The application delivers detailed information on airport-related services and facilities, such as hospitals, embassies, transportation networks, airlines, and flights; it also includes a structured presentation of the airport’s shops and their products, complemented by a discount mechanism. At the same time, ordinary users can receive live notifications regarding selected flights, whereas airport businesses are able to manage and update their products in real time. Data management is supported through a CRUD API and stored in an SQLite database, thus ensuring flexibility and scalability.

Particular emphasis is also placed on data retrieval and updating: the Selenium library is employed to automatically collect information concerning arrivals, departures, and weather conditions; the concurrent operation of the server and the data collection process is enabled by parallel programming with the threading library. In this manner, the application guarantees continuous updates, uninterrupted operation, and a high level of reliability.

Overall, the project integrates state-of-the-art technologies, focusing on security, user-friendliness, and smooth interaction among all stakeholders. Its contribution lies not only in the practical improvement of travel and service consumption experiences but also in the demonstration of a flexible and adaptable model for the development of web applications targeting airport and transportation infrastructures.

Keywords: Web application, Macedonia Airport, session management system, modular design, CRUD API, Python, JavaScript, modular design, notification system, backend, frontend, server, Flask, Selenium, SQLite, threading, scrapping data.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στους γονείς μου για την αμέριστη υποστήριξή τους καθ' όλη τη διάρκεια των σπουδών μου και στον κ. Κυριάκο Τσιακμάκη για την πολύτιμη βοήθεια, καθοδήγηση και απεριόριστη κατανόηση που επέδειξε κατά τη δημιουργία της παρούσας εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στα πλαίσια της παρούσας εργασίας αναπτύχθηκε μια διαδικτυακή εφαρμογή (web application) που αφορά το αεροδρόμιο «Μακεδονία» της Θεσσαλονίκης. Στόχος της εφαρμογής είναι να παρέχει ένα ολοκληρωμένο περιβάλλον πληροφόρησης και αλληλεπίδρασης για τους επισκέπτες, συνδυάζοντας τόσο το κομμάτι του backend (διαχείριση δεδομένων, βάση δεδομένων, επεξεργασία αιτημάτων) όσο και το modular frontend (παρουσίαση πληροφοριών, αποστολή πληροφοριών σε φιλικό και εύχρηστο γραφικό περιβάλλον).

Το αεροδρόμιο «Μακεδονία» αποτελεί το δεύτερο μεγαλύτερο αεροδρόμιο της Ελλάδας, εξυπηρετώντας χιλιάδες πτήσεις ετησίως, τόσο εσωτερικού όσο και εξωτερικού. Σκοπός της εφαρμογής είναι να παρουσιαστούν βασικές πληροφορίες για την πόλη της Θεσσαλονίκης και το αεροδρόμιο, όπως τα νοσοκομεία, οι πρεσβείες, οι συγκοινωνίες που εξυπηρετούν το αεροδρόμιο, οι αφίξεις και οι αναχωρήσεις με πιθανές καθυστερήσεις, οι συνεργαζόμενες αεροπορικές εταιρείες και οι καιρικές συνθήκες. Όλες αυτές οι πληροφορίες παρέχονται με τρόπο διαδραστικό και εύκολα προσβάσιμο, ώστε να διευκολύνεται η άμεση ενημέρωση των ταξιδιωτών και των επισκεπτών.

Στην παρούσα εργασία μελετώνται οι τεχνολογίες που είναι απαραίτητες για την υλοποίηση ενός ολοκληρωμένου συστήματος διαχείρισης και παρουσίασης πληροφοριών. Εξετάζεται ο παράλληλος προγραμματισμός όλων των επεξεργαστών οποιουδήποτε υπολογιστικού συστήματος (Windows, Linux, macOS, Android, iOS, Raspbian κ.ά.) που είναι σε θέση να εκτελέσει τη γλώσσα προγραμματισμού Python, με σκοπό την ανάπτυξη και λειτουργία διακομιστή ιστού (web server). Αντίστοιχα, αναλύεται η αξιοποίηση των επεξεργαστικών πόρων σε οποιοδήποτε μηχάνημα διαθέτει φυλλομετρητή ιστού (web browser), ώστε να είναι εφικτή η αλληλεπίδραση του χρήστη με τον διακομιστή. Η επικοινωνία μεταξύ των δύο μερών υλοποιείται μέσω του πρωτοκόλλου HTTP(S), ακολουθώντας το μοντέλο αιτήματος/απάντησης (request/response).

Η εφαρμογή περιλαμβάνει σύστημα σύνδεσης και προστασίας δεδομένων (login system) που διαχειρίζεται διαφορετικούς ρόλους χρηστών – απλούς επισκέπτες, καταστήματα και διαχειριστές – προσφέροντας προσωποποιημένη και ασφαλή εμπειρία. Το navigation bar προσαρμόζεται δυναμικά ανάλογα με τον ρόλο του χρήστη: εμφανίζει διαφορετικές επιλογές για τους επισκέπτες, τους συνδεδεμένους χρήστες, τα καταστήματα και τους διαχειριστές (π.χ. admin dashboard ή store dashboard), επιτρέποντας σε κάθε χρήστη να έχει πρόσβαση μόνο στις λειτουργίες και τα δεδομένα που του αντιστοιχούν. Το σύστημα διατηρεί την ταυτότητα του χρήστη ενεργή καθ’ όλη τη διάρκεια της σύνδεσης.

Η εργασία αυτή δεν περιορίζεται μόνο στην παρουσίαση πληροφοριών για το αεροδρόμιο «Μακεδονία», αλλά επιδιώκει να αναδείξει την ολοκληρωμένη διαδικασία ανάπτυξης ενός σύγχρονου διαδικτυακού συστήματος. Η υλοποίηση ενός Web Application με αρχιτεκτονική που συνδυάζει modular frontend και backend μπορεί εύκολα να επεκταθεί σε διάφορους προγραμματιστικούς τομείς πχ. στον χώρο του Internet of Things (IoT). Συγκεκριμένα, στο frontend, αντί τα κουμπιά στη γραμμή πλοήγησης (navigation bar) να εμφανίζουν πληροφορίες για το αεροδρόμιο, θα μπορούσαν να στέλνουν αιτήματα (requests) για ενέργειες σε IoT συσκευές, όπως το άναμμα ενός θερμοσίφωνα, η εκκίνηση μιας αντλίας ή η λήψη της τρέχουσας τιμής ενός αισθητήρα. Στο backend, τα αντίστοιχα endpoints (σημεία πρόσβασης) θα μπορούσαν να υλοποιούν την επικοινωνία με το υλικό (hardware) και να εκτελούν εντολές απευθείας στις ακίδες (pins) ενός μικροϋπολογιστή, όπως το Raspberry Pi. Με αυτόν τον τρόπο, η ίδια αρχιτεκτονική παρέχει μια ισχυρή και επεκτάσιμη βάση, ικανή να υποστηρίξει όχι μόνο την πληροφόρηση επιβατών, αλλά και την ανάπτυξη εφαρμογών για τον έξυπνο αυτοματισμό.

1.2 Δομή της εργασίας

Στην παρούσα εργασία αναπτύχθηκαν πέντε κεφάλαια, τα οποία συνδυάζουν τη μελέτη του θεωρητικού υπόβαθρου με τη δημιουργία και υλοποίηση της εφαρμογής. Κάθε κεφάλαιο πραγματεύεται διαφορετικές πτυχές του αντικειμένου, καλύπτοντας τόσο την ανάλυση και τον σχεδιασμό όσο και την πρακτική ανάπτυξη των επιμέρους λειτουργιών.

Κεφάλαιο 1 – Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται οι λόγοι για τους οποίους επιλέχθηκε το συγκεκριμένο θέμα και η σημασία των διαδικτυακών εφαρμογών στη σύγχρονη εποχή. Εξηγείται σε ποια υπολογιστικά συστήματα μπορεί να λειτουργήσει η εφαρμογή και αναφέρονται ορισμένα βασικά χαρακτηριστικά της. Τέλος, περιγράφονται συνοπτικά ο σκοπός και οι στόχοι της παρούσας εργασίας.

Κεφάλαιο 2 – Ανάλυση Τρέχοντος Επιπέδου

Στο κεφάλαιο αυτό περιγράφονται αναλυτικά οι τεχνολογίες και οι έννοιες που είναι απαραίτητες για τον σχεδιασμό, την υλοποίηση και την κατανόηση της παρούσας εργασίας. Παρουσιάζεται το θεωρητικό υπόβαθρο πάνω στο οποίο βασίζεται η ανάπτυξη της εφαρμογής· ταυτόχρονα αναλύονται οι βασικές μέθοδοι, τα εργαλεία και τα λογισμικά που αξιοποιήθηκαν.

Κεφάλαιο 3 – Προγράμματα Υποστήριξης και Τεχνολογίες Υλοποίησης

Στο κεφάλαιο αυτό περιγράφονται τα προγράμματα που χρησιμοποιήθηκαν για την ανάπτυξη του κώδικα και τον έλεγχο των δεδομένων και οι γλώσσες προγραμματισμού που επιλέχθηκαν.

Αναλύεται ο λόγος της επιλογής τους και ο τρόπος επικοινωνίας μεταξύ τους, ώστε να υποστηρίζεται η σωστή λειτουργία της εφαρμογής και η αποτελεσματική διαχείριση των δεδομένων.

Κεφάλαιο 4 – Το Σύστημα

Στο κεφάλαιο αυτό περιγράφονται αναλυτικά όλες οι δραστηριότητες που ακολουθήθηκαν για την υλοποίηση της διαδικτυακής εφαρμογής. Στην παρούσα εργασία εκτελέστηκαν σαράντα δύο δραστηριότητες, οι οποίες οργανώθηκαν σε δεκαοκτώ κατηγορίες για μεγαλύτερη σαφήνεια και ευκολία παρακολούθησης.

Επιπλέον, στο κεφάλαιο αυτό παρατίθεται και το εγχειρίδιο χρήσης της εφαρμογής, το οποίο περιγράφει τη λειτουργικότητα και τις διαδικασίες για κάθε ρόλο του συστήματος, διασφαλίζοντας την ορθή χρήση και την προστασία των δεδομένων.

Κεφάλαιο 5 – Συμπεράσματα και Προοπτικές

Στο τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα που προκύπτουν από την παρούσα εργασία, αναφορικά με τα εργαλεία που χρησιμοποιήθηκαν, τη μεθοδολογία ανάπτυξης του κώδικα και τον παράλληλο προγραμματισμό των επεξεργαστών των μηχανημάτων που αξιοποιήθηκαν. Επιπλέον, εξετάζεται η δυνατότητα χρήσης της εφαρμογής σε άλλα συστήματα, επιπροσθέτως και οι προοπτικές για περαιτέρω βελτιώσεις. Εν τέλει, παρατίθενται τα επόμενα βήματα που μπορούν να ακολουθηθούν για την αναβάθμιση και επέκταση της διαδικτυακής εφαρμογής, με στόχο την αύξηση της αποδοτικότητας και της λειτουργικότητάς της.

Βιβλιογραφία

Παράρτημα Α

Παρατίθεται το παράρτημα με κάποιους από τους κώδικες που χρησιμοποιήθηκαν στην εργασία.

Κεφάλαιο 2ο: Ανάλυση Τρέχοντος Επιπέδου

2.1 Εισαγωγή

Διαδικτυακές εφαρμογές έχουν αναπτυχθεί σε διάφορους τομείς, όπως οι πλατφόρμες διαχείρισης αεροδρομίων, τα συστήματα κρατήσεων ξενοδοχείων ή οι εφαρμογές πληροφόρησης για δημόσιες συγκοινωνίες, IoT συστήματα κ.α. Αυτά τα συστήματα συνήθως συνδυάζουν frontend και backend, παρέχοντας διαδραστική πλοήγηση, δυναμική φόρτωση περιεχομένου μέσω API και εξατομικευμένη εμπειρία ανάλογα με τον χρήστη. Επιπλέον, πολλά από αυτά χρησιμοποιούν αρθωτή (modular) αρχιτεκτονική με components, ώστε να είναι επεκτάσιμα, ευέλικτα και εύκολα στη συντήρηση, υποστηρίζοντας διαφορετικούς ρόλους χρηστών και συχνά ενσωματώνουν ασφαλή συστήματα login για την προστασία των δεδομένων.

Για τη δημιουργία και για την κατανόηση της παρούσας εργασίας, απαιτείται γνώση της γλώσσας προγραμματισμού Python, της γλώσσας προγραμματισμού JavaScript, της δηλωτικής γλώσσας SQL, της γλώσσας σήμανσης HTML, όπως επίσης και της γλώσσας μορφοποίησης **CSS**. Επιπλέον, απαιτείται η κατανόηση των εννοιών: διακομιστής ιστού (web server), διεπαφή προγραμματισμού εφαρμογών (API - Application Programming Interface), API για Δημιουργία, Ανάγνωση, Ενημέρωση και Διαγραφή δεδομένων (CRUD API), παράλληλος προγραμματισμός (threading), συλλογή δεδομένων με αυτοματοποίηση (data scraping), βάση δεδομένων (database), φυλλομετρητής ιστού (web browser), διαδικτυακή εφαρμογή (web application), σύστημα προστασίας δεδομένων (login system), μηχανισμός ειδοποιήσεων (notification system), υβριδικό HTML (Jinja), μενού πλοήγησης (navigation bar), αρθρωτό στοιχείο (modular component) και ενσωματωμένη εφαρμογή/πλαίσιο (widget/iframe).

2.2 Γλώσσα προγραμματισμού Python

Η Python είναι μια υψηλού επιπέδου, ερμηνευόμενη γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως για τη δημιουργία διαφόρων ειδών λογισμικού, από απλά σενάρια αυτοματισμού μέχρι πολύπλοκες web εφαρμογές και επιστημονική ανάλυση δεδομένων. Αναπτύχθηκε από τον Guido van Rossum και η πρώτη της έκδοση κυκλοφόρησε το 1991 [1]. Είναι γνωστή για την απλότητά της και την ευανάγνωστη σύνταξή της, γεγονός που την καθιστά ιδανική για αρχάριους, αλλά και για έμπειρους προγραμματιστές. Η Python αναπτύσσεται ως ανοιχτό λογισμικό και η διαχείρισή της γίνεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Το όνομα της γλώσσας προέρχεται από την ομάδα των Άγγλων κωμικών Monty Python [2], παρότι το λογότυπο της εμφανίζει έναν πύθωνα.

Βασικά χαρακτηριστικά της Python είναι η απλότητα και καθαρή σύνταξη που ενθαρρύνει τη χρήση αναγνωρίσιμου και καθαρού κώδικα, διευκολύνοντας την εκμάθησή της. Είναι διαδραστική και

ερμηνευόμενη, με τον κώδικα να εκτελείται γραμμή προς γραμμή από τον διερμηνέα, διευκολύνοντας την ανίχνευση σφαλμάτων και τον πειραματισμό. Η Python υποστηρίζει δυναμική πληκτρολόγηση, με τις μεταβλητές να μην απαιτούν δήλωση του τύπου τους.

Διαθέτει μια μεγάλη ποικιλία από βιβλιοθήκες και πλαίσια που καλύπτουν ένα ευρύ φάσμα εφαρμογών, όπως NumPy και Pandas για επιστημονικούς υπολογισμούς, Django και Flask για web ανάπτυξη, TensorFlow και PyTorch για μηχανική μάθηση. Η Python είναι πολυπλατφορμική και διατίθεται για πολλά λειτουργικά συστήματα, επιτρέποντας την ανάπτυξη εφαρμογών σε διαφορετικές πλατφόρμες.

Έχει μια από τις μεγαλύτερες και πιο ενεργές κοινότητες προγραμματιστών, προσφέροντας άφθονο υλικό εκμάθησης, τεκμηρίωση και υποστήριξη μέσω φόρουμ και άλλων πόρων. Η Python επιλέχθηκε γιατί είναι διαλειτουργική και πολυπλατφορμική, δηλαδή μπορεί να εκτελεστεί σε διάφορα λειτουργικά συστήματα, όπως Windows, Linux, macOS, αλλά και σε μικροϋπολογιστές όπως το Raspberry Pi. Η υποστήριξη για διασύνδεση με web servers και βάσεις δεδομένων την καθιστά ιδανική επιλογή για την ανάπτυξη διαδικτυακών εφαρμογών, όπως η παρούσα εργασία.

2.3 Γλώσσα προγραμματισμού Javascript

Η JavaScript είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου που χρησιμοποιείται κυρίως για τη δημιουργία δυναμικών και διαδραστικών ιστοσελίδων. Σε αντίθεση με γλώσσες όπως η Python ή η C++, η JavaScript εκτελείται κυρίως στο φυλλομετρητή (web browser) του χρήστη, επιτρέποντας την άμεση αλληλεπίδραση με το περιβάλλον μιας ιστοσελίδας χωρίς να απαιτείται ανανέωση της σελίδας. Αυτό καθιστά δυνατή τη δημιουργία λειτουργιών όπως κινούμενα μενού, φόρμες επικοινωνίας, δυναμική ενημέρωση περιεχομένου και αλληλεπίδραση με APIs [3].

Η JavaScript είναι μια δραστηριότητα αντικειμενοστραφής γλώσσα, με δυνατότητες λειτουργικού προγραμματισμού, που σημαίνει ότι επιτρέπει την οργάνωση του κώδικα σε αντικείμενα, συναρτήσεις και μεθόδους. Αυτό διευκολύνει την ανάπτυξη σύνθετων εφαρμογών και την επαναχρησιμοποίηση κώδικα. Η υποστήριξη για modular components επιτρέπει τη δημιουργία ανεξάρτητων τμημάτων εφαρμογής που μπορούν να ενσωματωθούν σε μεγαλύτερα συστήματα.

Με την εμφάνιση του Node.js, η JavaScript πέρασε και στο backend, επιτρέποντας την εκτέλεση κώδικα εκτός φυλλομετρητή. Αυτό σημαίνει ότι η ίδια γλώσσα μπορεί να χρησιμοποιηθεί για την ανάπτυξη πλήρους εφαρμογής, από τον διακομιστή (server-side) μέχρι το περιβάλλον του χρήστη (client-side), καθιστώντας την ιδανική επιλογή για σύγχρονες διαδικτυακές εφαρμογές (web apps) που χρειάζονται ομοιογενή τεχνολογικό stack.

Η JavaScript διαθέτει ένα πλούσιο οικοσύστημα βιβλιοθηκών και frameworks, όπως React, Vue και Angular, τα οποία επιτρέπουν την ταχύτερη ανάπτυξη και συντήρηση πολύπλοκων εφαρμογών.

Παράλληλα, η συνεχής ανάπτυξη και οι πρότυπες ενημερώσεις της γλώσσας εξασφαλίζουν ότι παραμένει επίκαιρη και κατάλληλη για μοντέρνα web projects.

2.4 Δηλωτική Γλώσσα SQL (Structured Query Language)

Η SQL επιτρέπει τη δημιουργία και οργάνωση βάσεων δεδομένων, τον ορισμό πινάκων και σχέσεων μεταξύ τους και την εκτέλεση σύνθετων ερωτημάτων (queries) για την εξαγωγή συγκεκριμένων πληροφοριών [4]. Μέσω εντολών όπως SELECT, INSERT, UPDATE και DELETE, οι χρήστες μπορούν να αλληλεπιδρούν με τα δεδομένα με τρόπο αποτελεσματικό και τυποποιημένο, ανεξάρτητα από το μέγεθος ή την πολυπλοκότητα της βάσης δεδομένων.

Επιπλέον, η SQL υποστηρίζει μηχανισμούς ελέγχου πρόσβασης και ασφάλειας, επιτρέποντας τον καθορισμό ποιος χρήστης μπορεί να διαβάζει ή να τροποποιεί δεδομένα. Συνδυάζοντας τη με άλλες τεχνολογίες, όπως το Python ή το JavaScript, η SQL γίνεται ένα ισχυρό εργαλείο για την ανάπτυξη διαδικτυακών εφαρμογών, όπου η διαχείριση δεδομένων είναι κρίσιμη.

Η SQL είναι ευρέως χρησιμοποιούμενη στη βιομηχανία, με δημοφιλείς υλοποιήσεις όπως MySQL, PostgreSQL και SQLite. Η γνώση της SQL αποτελεί βασικό εφόδιο για κάθε προγραμματιστή που ασχολείται με βάσεις δεδομένων και εφαρμογές που απαιτούν αποτελεσματική αποθήκευση και ανάκτηση δεδομένων.

2.5 Γλώσσα Σήμανσης HTML

Η HTML (HyperText Markup Language) είναι η γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία και δομή ιστοσελίδων στο Διαδίκτυο. Σε αντίθεση με τις γλώσσες προγραμματισμού, η HTML δεν περιγράφει λογική ή αλγοριθμικές διαδικασίες, αλλά καθορίζει πώς οργανώνονται και εμφανίζονται τα στοιχεία μιας ιστοσελίδας [5]. Με την HTML, ο προγραμματιστής ορίζει επικεφαλίδες, παραγράφους, συνδέσμους, εικόνες, πίνακες, λίστες και άλλα στοιχεία περιεχομένου.

Η HTML λειτουργεί μέσω tags (ετικετών), οι οποίες περιβάλλουν το περιεχόμενο και του αποδίδουν νόημα και δομή. Για παράδειγμα, το <h1> χρησιμοποιείται για κύριους τίτλους, το <p> για παραγράφους και το <a> για υπερσυνδέσμους. Με αυτόν τον τρόπο, η HTML επιτρέπει την αναπαράσταση των πληροφοριών σε μια οργανωμένη και κατανοητή μορφή, τόσο για τους χρήστες όσο και για τους φυλλομετρητές ιστού (web browsers).

Επιπλέον, η HTML συνεργάζεται με άλλες τεχνολογίες, όπως η CSS για τη μορφοποίηση και το στυλ των στοιχείων και η JavaScript για την προσθήκη δυναμικότητας και αλληλεπίδρασης. Σε συνδυασμό, αυτές οι τεχνολογίες αποτελούν τη βάση για τη δημιουργία διαδικτυακών εφαρμογών και ιστοσελίδων με σύγχρονη εμφάνιση και λειτουργικότητα.

Η HTML εξελίσσεται συνεχώς με νέες εκδόσεις, όπως η HTML5, η οποία εισάγει νέες δυνατότητες, όπως πολυμέσα (audio, video), ενσωματωμένα γραφικά (canvas) και βελτιωμένη υποστήριξη για εφαρμογές web, καθιστώντας την πιο ισχυρή και ευέλικτη για σύγχρονες διαδικτυακές ανάγκες.

2.6 Γλώσσα Μορφοποίησης CSS

Η CSS (Cascading Style Sheets) είναι η γλώσσα μορφοποίησης που χρησιμοποιείται για τον καθορισμό της εμφάνισης και της διάταξης των στοιχείων σε μια ιστοσελίδα. Σε αντίθεση με την HTML, που καθορίζει τη δομή και το περιεχόμενο, η CSS εστιάζει στο πώς αυτά τα στοιχεία εμφανίζονται στους χρήστες. Μέσω της CSS, μπορούμε να ορίσουμε χρώματα, γραμματοσειρές, περιθώρια, σκιές, θέσεις, διατάξεις και πολλές άλλες οπτικές ιδιότητες [6].

Η CSS λειτουργεί με κανόνες (rules), οι οποίοι συνδέουν selectors (επιλογείς) με declarations (δηλώσεις ιδιοτήτων). Για παράδειγμα, ένας κανόνας μπορεί να καθορίζει ότι όλα τα <h1> στοιχεία μιας σελίδας θα εμφανίζονται με μπλε χρώμα και γραμματοσειρά Arial. Αυτό επιτρέπει τον πλήρη έλεγχο της αισθητικής της ιστοσελίδας χωρίς να επηρεάζεται η δομή της που καθορίζεται από την HTML.

Επιπλέον, η CSS υποστηρίζει τεχνικές όπως responsive design, που προσαρμόζει την εμφάνιση της ιστοσελίδας ανάλογα με τη συσκευή του χρήστη (υπολογιστής, tablet, κινητό). Η χρήση CSS σε συνδυασμό με HTML και JavaScript επιτρέπει τη δημιουργία σύγχρονων, δυναμικών και ευχάριστων στο μάτι διαδικτυακών εφαρμογών.

Η CSS εξελίσσεται συνεχώς, με νέες δυνατότητες όπως flexbox, grid layout και animations, οι οποίες επιτρέπουν πιο σύνθετες και ευέλικτες διατάξεις και εφέ, καθιστώντας την απαραίτητη για οποιονδήποτε ασχολείται με τον σχεδιασμό ιστοσελίδων και web εφαρμογών.

2.7 Διακομιστής Ιστού (Web Server)

Ένας web server είναι ένα λογισμικό ή/και υλικό σύστημα που δέχεται αιτήματα (requests) από πελάτες (clients), συνήθως φυλλομετρητές ιστού (web browsers) και παρέχει τις αντίστοιχες απαντήσεις (responses). Οι απαντήσεις αυτές μπορεί να περιλαμβάνουν στατικά αρχεία, όπως HTML, CSS, εικόνες ή JavaScript, αλλά και δυναμικά παραγόμενο περιεχόμενο που προέρχεται από βάσεις δεδομένων ή άλλες πηγές. Ο web server υλοποιεί το πρωτόκολλο HTTP/HTTPS, που καθορίζει πώς ανταλλάσσονται τα δεδομένα στο διαδίκτυο [7].

Στο πλαίσιο μιας διαδικτυακής εφαρμογής (web app), ο web server λειτουργεί ως ενδιάμεσος μεταξύ του χρήστη και του backend, διαχειρίζοντας αιτήματα εισαγωγής δεδομένων, αναζητήσεις σε βάσεις δεδομένων ή κλήσεις σε API. Επιπλέον, οι σύγχρονοι web servers μπορούν να υποστηρίζουν ασφάλεια, διαχείριση συνεδριών (sessions) και φορτώσεις πολλαπλών χρηστών ταυτόχρονα, καθιστώντας τους κρίσιμο κομμάτι για την αξιόπιστη λειτουργία μιας εφαρμογής.

Παράλληλα, ένα web server μπορεί να σερβίρει τόσο στατικό περιεχόμενο, όπως αρχεία HTML και εικόνες, όσο και δυναμικό περιεχόμενο, το οποίο παράγεται κατά παραγγελία ανάλογα με τα δεδομένα ή τις ενέργειες του χρήστη.

Η σωστή ρύθμιση και διαχείριση ενός web server είναι ουσιώδης για τη διαθεσιμότητα, την ασφάλεια και την απόδοση της διαδικτυακής εφαρμογής, όπως και κάθε αίτημα των χρηστών περνά μέσα από αυτόν πριν φτάσει στο backend ή στον χρήστη

2.8 Διεπαφή προγραμματισμού εφαρμογών (API)

Ένα API (Application Programming Interface) είναι ένα σύνολο κανόνων και προδιαγραφών που επιτρέπει σε διαφορετικά λογισμικά να επικοινωνούν μεταξύ τους. Μέσω ενός API, μια εφαρμογή μπορεί να στέλνει αιτήματα σε ένα άλλο λογισμικό ή σύστημα και να λαμβάνει απαντήσεις, χωρίς να χρειάζεται να γνωρίζει τις εσωτερικές λεπτομέρειες της λειτουργίας του άλλου συστήματος. Τα API αποτελούν βασικό στοιχείο για την υλοποίηση σύγχρονων διαδικτυακών εφαρμογών, επειδή διευκολύνουν την επικοινωνία μεταξύ frontend και backend ή μεταξύ διαφορετικών υπηρεσιών [8].

Στην πράξη, τα API μπορούν να παρέχουν δεδομένα από βάσεις δεδομένων. Η επικοινωνία γίνεται συνήθως μέσω του πρωτοκόλλου HTTP/HTTPS και ακολουθεί το μοντέλο αιτήματος/απάντησης (request/response), καθιστώντας τη διαχείριση και την ανταλλαγή δεδομένων δομημένη και ασφαλή.

Τα API επιτρέπουν επίσης την επεκτασιμότητα της εφαρμογής, διότι νέα λειτουργικότητα μπορεί να προστεθεί χωρίς να τροποποιείται ολόκληρο το σύστημα. Η ύπαρξη καλά οργανωμένων API διευκολύνει τη συντήρηση, την αναβάθμιση και την ενσωμάτωση τρίτων υπηρεσιών ή εφαρμογών σε ένα ολοκληρωμένο διαδικτυακό σύστημα.

2.9 CRUD API

Ένα CRUD API είναι ένα είδος API που παρέχει τις βασικές λειτουργίες για τη διαχείριση δεδομένων: Δημιουργία (Create), Ανάγνωση (Read), Ενημέρωση (Update) και Διαγραφή (Delete). Μέσω αυτών των λειτουργιών, η εφαρμογή μπορεί να αλληλεπιδρά με μια βάση δεδομένων με δομημένο και ελεγχόμενο τρόπο, επιτρέποντας την καταχώρηση νέων δεδομένων, την ανάκτηση υπαρχόντων, την τροποποίηση πληροφοριών και τη διαγραφή περιττών ή λανθασμένων εγγραφών [9].

Τα CRUD API αποτελούν τη βάση για τη λειτουργία πολλών διαδικτυακών εφαρμογών, επιτρέποντας την ασφαλή και συνεπή διαχείριση των δεδομένων χωρίς να εκτίθενται οι εσωτερικές λεπτομέρειες της βάσης δεδομένων στο χρήστη. Η επικοινωνία με το CRUD API γίνεται συνήθως μέσω του πρωτοκόλλου HTTP/HTTPS, χρησιμοποιώντας μεθόδους όπως GET, POST, PATCH και DELETE, οι οποίες αντιστοιχούν στις βασικές CRUD ενέργειες.

2.10 Παράλληλος προγραμματισμός (Threading)

Ο παράλληλος προγραμματισμός (threading) είναι μια τεχνική που επιτρέπει την εκτέλεση πολλαπλών κομματιών ενός προγράμματος ταυτόχρονα, αξιοποιώντας καλύτερα τους διαθέσιμους πόρους του υπολογιστικού συστήματος. Κάθε νήμα (thread) εκτελεί ανεξάρτητες εργασίες, οι οποίες μπορούν να τρέχουν παράλληλα με άλλα νήματα, αυξάνοντας την αποδοτικότητα και τη διαχείριση χρόνου εκτέλεσης, ειδικά σε εφαρμογές που χρειάζονται ταυτόχρονη επεξεργασία πολλών αιτημάτων ή δεδομένων [10].

Ο παράλληλος προγραμματισμός είναι ιδιαίτερα χρήσιμος σε διαδικτυακές εφαρμογές, εφόσον επιτρέπει την ταυτόχρονη διαχείριση πολλαπλών χρηστών ή αιτημάτων χωρίς να μπλοκάρεται η κύρια ροή της εφαρμογής. Μέσω των νημάτων, είναι δυνατή η εκτέλεση εργασιών, όπως επί παραδείγματι η επικοινωνία με βάσεις δεδομένων, η συλλογή δεδομένων από εξωτερικές πηγές και η επεξεργασία δεδομένων στο παρασκήνιο, διατηρώντας τη διεπαφή χρήστη αξιόπιστη (responsive) και λειτουργική.

Η χρήση threading προσφέρει σημαντικά πλεονεκτήματα σε ό,τι αφορά την ταχύτητα και την αποδοτικότητα, αλλά απαιτεί προσεκτικό σχεδιασμό για την αποφυγή συγκρούσεων δεδομένων (data races) και την εξασφάλιση της ασφάλειας και της ακεραιότητας των πληροφοριών.

2.11 Συλλογή δεδομένων (Data Scraping)

Η συλλογή δεδομένων με αυτοματισμό (data scraping) είναι η διαδικασία εξαγωγής πληροφοριών από ιστοσελίδες ή άλλες πηγές δεδομένων με τη βοήθεια προγραμμάτων και scripts, χωρίς την ανάγκη χειροκίνητης συλλογής (συνήθως με αυτοματοποιημένη χρήση web browser). Αυτή η τεχνική επιτρέπει την αποδοτική συγκέντρωση μεγάλου όγκου δεδομένων που μπορούν να χρησιμοποιηθούν για ανάλυση, παρουσίαση ή ενημέρωση σε διαδικτυακές εφαρμογές [11].

Η αυτοματοποιημένη συλλογή δεδομένων είναι ιδιαίτερα χρήσιμη για εφαρμογές που απαιτούν συνεχόμενη ενημέρωση πληροφοριών, όπως τα προγράμματα αφίξεων και αναχωρήσεων, οι καιρικές συνθήκες ή οποιαδήποτε άλλη πληροφορία μεταβάλλεται συχνά. Η χρήση scripts για data scrapping εξασφαλίζει ταχύτητα, ακρίβεια και επαναληψιμότητα της διαδικασίας, ενώ συνδέεται άμεσα με την ομαλή λειτουργία του backend, όπου τα δεδομένα επεξεργάζονται και αποθηκεύονται για χρήση από το frontend.

Παράλληλα, η υλοποίηση αυτής της διαδικασίας απαιτεί σεβασμό στους κανόνες πρόσβασης των ιστοσελίδων και τη νομοθεσία περί πνευματικών δικαιωμάτων, ώστε η συλλογή των δεδομένων να είναι νόμιμη και ηθικά σωστή.

2.12 Βάση Δεδομένων (Database)

Μια βάση δεδομένων (database) είναι ένα οργανωμένο σύστημα αποθήκευσης και διαχείρισης δεδομένων, το οποίο επιτρέπει την αποδοτική εισαγωγή, ανάκτηση, ενημέρωση και διαγραφή πληροφοριών. Οι βάσεις δεδομένων χρησιμοποιούνται για τη συγκέντρωση και διαχείριση μεγάλου όγκου δεδομένων με τρόπο δομημένο και ασφαλή, έτσι ώστε να είναι εύκολα προσβάσιμα από τις εφαρμογές [12].

Στο πλαίσιο των διαδικτυακών εφαρμογών, η βάση δεδομένων παίζει κεντρικό ρόλο στην αποθήκευση πληροφοριών σχετικά με χρήστες, προϊόντα, πτήσεις και οποιοδήποτε άλλο περιεχόμενο χρειάζεται η εφαρμογή. Η επικοινωνία με τη βάση δεδομένων πραγματοποιείται μέσω ειδικών εντολών (π.χ. SQL queries) ή API, που επιτρέπουν στο backend να διαχειρίζεται τα δεδομένα και να παρέχει στον χρήστη τις κατάλληλες πληροφορίες σε πραγματικό χρόνο.

Η σωστή σχεδίαση και διαχείριση της βάσης δεδομένων εξασφαλίζει την αξιοπιστία, την ταχύτητα και την ασφάλεια της εφαρμογής, καθιστώντας τη λειτουργία της ομαλή και επεκτάσιμη σε μελλοντικές ανάγκες.

2.13 Φυλλομετρητής ιστού (Web Browser)

Ο φυλλομετρητής ιστού (web browser) είναι ένα πρόγραμμα που επιτρέπει την πρόσβαση, προβολή και αλληλεπίδραση με περιεχόμενο στο Διαδίκτυο. Μέσω του φυλλομετρητή, ο χρήστης μπορεί να περιηγηθεί σε ιστοσελίδες, να υποβάλει δεδομένα μέσω φορμών, να εκτελεί εφαρμογές web και να λαμβάνει δυναμικό περιεχόμενο από έναν διακομιστή [13].

Ο φυλλομετρητής ερμηνεύει γλώσσες σήμανσης όπως το HTML και στυλ που ορίζονται με CSS, εκτελώντας κώδικα (scripts) γραμμένο σε γλώσσα προγραμματισμού JavaScript για την υποστήριξη διαδραστικών λειτουργιών. Παράλληλα, χειρίζεται την αποστολή και λήψη αιτημάτων HTTP(S), επιτρέποντας την επικοινωνία με τον διακομιστή, ώστε τα δεδομένα να εμφανίζονται στον χρήστη με φιλικό και εύχρηστο τρόπο.

Η σωστή χρήση του φυλλομετρητή εξασφαλίζει τη λειτουργικότητα της διαδικτυακής εφαρμογής, την αλληλεπίδραση με τον χρήστη και την παρουσίαση των πληροφοριών με τρόπο κατανοητό και ευχάριστο.

2.14 Διαδικτυακή Εφαρμογή (web application)

Η διαδικτυακή εφαρμογή (web application) είναι ένα λογισμικό που εκτελείται μέσω ενός φυλλομετρητή ιστού και παρέχει υπηρεσίες ή λειτουργίες στους χρήστες χωρίς να απαιτείται εγκατάσταση τοπικά στον υπολογιστή τους. Οι web εφαρμογές συνδυάζουν το frontend, που αναλαμβάνει την παρουσίαση των πληροφοριών και την αλληλεπίδραση με τον χρήστη, με το backend, το οποίο χειρίζεται τα δεδομένα, τις βάσεις δεδομένων και τις επιχειρησιακές λογικές [14].

Η λειτουργία μιας διαδικτυακής εφαρμογής χαρακτηρίζεται από παραλληλία, καθότι ο διακομιστής είναι σε θέση να εξυπηρετεί πολλαπλά αιτήματα ταυτόχρονα, χρησιμοποιώντας τεχνικές όπως multi-threading ή asynchronous processing στο backend. Παράλληλα και το frontend αξιοποιεί την παραλληλία μέσω ασύγχρονων αιτημάτων (asynchronous fetch requests), επιτρέποντας την ταυτόχρονη φόρτωση δεδομένων, την ενημέρωση περιεχομένου και την αλληλεπίδραση με τον χρήστη χωρίς να «παγώνει» η διεπαφή. Αυτό εξασφαλίζει ότι πολλοί χρήστες μπορούν να αλληλεπιδρούν με την εφαρμογή χωρίς καθυστερήσεις, με τις διάφορες λειτουργίες να εκτελούνται παράλληλα και ανεξάρτητα η μία από την άλλη.

Μέσα από την επικοινωνία με τον διακομιστή μέσω πρωτοκόλλων HTTP(S), οι διαδικτυακές εφαρμογές μπορούν να εμφανίζουν δυναμικό περιεχόμενο, να διαχειρίζονται αιτήματα χρηστών και να προσφέρουν εξατομικευμένες λειτουργίες ανάλογα με το ρόλο και τα δικαιώματα κάθε χρήστη. Η αξιοποίηση σύγχρονων τεχνολογιών, όπως JavaScript, HTML, CSS και API, επιτρέπει τη δημιουργία φιλικών, διαδραστικών και επεκτάσιμων συστημάτων που ικανοποιούν τις ανάγκες της σύγχρονης ψηφιακής εμπειρίας.

2.15 Σύστημα προστασίας δεδομένων (login system)

Το σύστημα προστασίας δεδομένων (login system) αποτελεί βασικό μηχανισμό ασφάλειας σε μια διαδικτυακή εφαρμογή, δεδομένου ότι εξασφαλίζει ότι μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση σε συγκεκριμένες λειτουργίες ή πληροφορίες. Μέσω της διαδικασίας ταυτοποίησης (authentication), ο χρήστης εισάγει διαπιστευτήρια, όπως όνομα χρήστη και κωδικό πρόσβασης, τα οποία ελέγχονται σε σύγκριση με αποθηκευμένα, κρυπτογραφημένα δεδομένα στη βάση. Σε πιο εξελιγμένες μορφές, το σύστημα υποστηρίζει και εξουσιοδότηση (authorization), ώστε διαφορετικοί ρόλοι χρηστών (π.χ. διαχειριστής, κατάστημα, απλός επισκέπτης) να έχουν πρόσβαση σε διαφορετικό περιεχόμενο και δυνατότητες. Ένα χαρακτηριστικό παράδειγμα αποτελεί η δυναμική αλλαγή του μενού πλοήγησης (navigation bar), το οποίο προσαρμόζεται ανάλογα με τον ρόλο και την κατάσταση σύνδεσης του χρήστη, παρέχοντας πρόσβαση είτε σε διαχειριστικό πίνακα (admin dashboard), είτε σε πίνακα καταστήματος (store dashboard), είτε σε γενικό περιβάλλον χρήστη [15].

Η ασφάλεια του συστήματος ενισχύεται με τη χρήση πρωτοκόλλου HTTPS για την προστασία των δεδομένων κατά τη μεταφορά, την κρυπτογράφηση κωδικών πρόσβασης με ασφαλείς αλγορίθμους κατακερματισμού (όπως bcrypt ή SHA-256) και τη διαχείριση session tokens ή cookies για τη διατήρηση της κατάστασης σύνδεσης. Επιπλέον, μπορούν να χρησιμοποιηθούν μηχανισμοί όπως πολλαπλή επαλήθευση ταυτότητας (multi-factor authentication) ή περιορισμός προσπαθειών εισόδου, ώστε να μειωθεί ο κίνδυνος μη εξουσιοδοτημένης πρόσβασης. Με αυτόν τον τρόπο διασφαλίζεται τόσο η προστασία των δεδομένων των χρηστών όσο και η ομαλή και προσωποποιημένη εμπειρία πλοήγησης στην εφαρμογή.

2.16 Μηχανισμός ειδοποιήσεων (notification system)

Ο μηχανισμός ειδοποιήσεων (notification system) αποτελεί ένα σημαντικό στοιχείο σε μια διαδικτυακή εφαρμογή, λαμβάνοντας υπόψιν ότι επιτρέπει την άμεση και αποτελεσματική ενημέρωση του χρήστη σχετικά με γεγονότα, αλλαγές ή ενέργειες που απαιτούν προσοχή. Οι ειδοποιήσεις μπορούν να αφορούν, για παράδειγμα, την ολοκλήρωση μιας ενέργειας, την ύπαρξη νέων δεδομένων, την εμφάνιση σφαλμάτων ή ακόμα και την παροχή οδηγιών χρήσης. Η υλοποίησή τους στο frontend γίνεται συνήθως με δυναμικά στοιχεία διεπαφής (π.χ. αναδυόμενα παράθυρα/modals), τα οποία δημιουργούνται μέσω JavaScript και CSS· εν αντιθέσει, στο backend μπορούν να βασίζονται σε γεγονότα (events) που ενεργοποιούν την αποστολή των κατάλληλων μηνυμάτων. Επιπλέον, σε πολλές περιπτώσεις, οι ειδοποιήσεις συνοδεύονται από ηχητικά σήματα, ώστε να προσελκύεται άμεσα η προσοχή του χρήστη [16].

Στα σύγχρονα συστήματα, οι ειδοποιήσεις συχνά λειτουργούν με ασύγχρονα αιτήματα (asynchronous requests) ή με μηχανισμούς όπως το WebSocket και το Server-Sent Events (SSE), ώστε να παρέχεται ενημέρωση σε πραγματικό χρόνο χωρίς να απαιτείται ανανέωση της σελίδας. Έτσι, ο χρήστης απολαμβάνει μια πιο άμεση και διαδραστική εμπειρία, μια και η εφαρμογή μπορεί να του παρουσιάζει κρίσιμες πληροφορίες τη στιγμή που συμβαίνουν. Παράλληλα, η σωστή σχεδίαση του μηχανισμού ειδοποιήσεων συμβάλλει στην καλύτερη χρηστικότητα, αποφεύγοντας την υπερφόρτωση με περιττά μηνύματα και διατηρώντας την πλοήγηση ομαλή και φιλική.

2.17 Υβριδικό HTML (Jinja)

Το Jinja αποτελεί μια μηχανή προτύπων (template engine) που χρησιμοποιείται κυρίως σε εφαρμογές βασισμένες στη γλώσσα προγραμματισμού Python. Πρόκειται για μια μορφή υβριδικού HTML, η οποία συνδυάζει την κλασική γλώσσα σήμανσης HTML με ειδικές εντολές και μεταβλητές της Python. Με αυτόν τον τρόπο δίνεται η δυνατότητα να δημιουργούνται δυναμικές ιστοσελίδες, όπου το περιεχόμενο δεν είναι στατικό αλλά προσαρμόζεται ανάλογα με τα δεδομένα που παρέχει ο διακομιστής (backend). Για παράδειγμα, μέσω του Jinja μπορούν να εμφανίζονται λίστες δεδομένων από μια βάση, να δημιουργούνται δυναμικά μενού ή να προσαρμόζεται η εμφάνιση μιας σελίδας ανάλογα με τον ρόλο του χρήστη [17].

Ένα από τα βασικά πλεονεκτήματα του Jinja είναι η παραμετροποίηση και επαναχρησιμοποίηση του κώδικα. Ο προγραμματιστής μπορεί να ορίσει πρότυπα (templates) και να τα εμπλουτίσει με δυναμικές μεταβλητές, συνθήκες ή βρόχους, ώστε η ίδια σελίδα να μπορεί να προβάλλει διαφορετικό περιεχόμενο χωρίς την ανάγκη επανάληψης κώδικα. Αυτό το χαρακτηριστικό το καθιστά ιδιαίτερα αποδοτικό σε σύνθετα έργα, όπου απαιτείται η συνδυασμένη λειτουργία πολλών σελίδων με κοινή δομή.

Στο πλαίσιο μιας διαδικτυακής εφαρμογής, το Jinja λειτουργεί ως γέφυρα μεταξύ frontend και backend, επιτρέποντας στο διακομιστή να εισάγει δεδομένα κατευθείαν μέσα στον HTML κώδικα που παραδίδεται στον φυλλομετρητή του χρήστη. Με αυτόν τον τρόπο υποστηρίζεται η παράλληλη λειτουργία με άλλες τεχνολογίες· το frontend μπορεί να συνεχίσει να χρησιμοποιεί JavaScript για ασύγχρονα αιτήματα (asynchronous requests), ταυτόχρονα το backend διαχειρίζεται και σερβίρει τα δεδομένα μέσω Jinja templates.

2.18 Μενού πλοήγησης (Navigation Bar)

Το μενού πλοήγησης (navigation bar) αποτελεί βασικό στοιχείο μιας διαδικτυακής εφαρμογής, παρέχοντας στον χρήστη τη δυνατότητα εύκολης και άμεσης μετακίνησης ανάμεσα σε διαφορετικές λειτουργίες ή ενότητες του συστήματος. Συνήθως βρίσκεται στο πάνω μέρος της σελίδας και περιλαμβάνει συνδέσμους (links) ή κουμπιά που ενεργοποιούν συγκεκριμένες ενέργειες, όπως η μετάβαση σε σελίδες πληροφοριών, η εμφάνιση καταλόγων δεδομένων ή η πρόσβαση σε προσωπικό λογαριασμό. Η σωστή σχεδίαση ενός navigation bar είναι κρίσιμη για την καλή εμπειρία χρήστη (user experience), αφού διευκολύνει τον προσανατολισμό και τη γρήγορη εκτέλεση ενεργειών [18].

Σε πιο εξελιγμένα συστήματα, το μενού πλοήγησης προσαρμόζεται δυναμικά ανάλογα με τον ρόλο ή την κατάσταση του χρήστη. Για παράδειγμα, σε μια εφαρμογή με σύστημα εισόδου (login system), ένας μη συνδεδεμένος επισκέπτης μπορεί να βλέπει περιορισμένες επιλογές, ενώ μετά τη σύνδεση εμφανίζονται πρόσθετες λειτουργίες. Αντίστοιχα, διαφορετικοί ρόλοι, όπως διαχειριστής (admin), ιδιοκτήτης καταστήματος (store) ή απλός χρήστης (user), ενδέχεται να έχουν ξεχωριστά nav bars με επιλογές που αντιστοιχούν στα δικαιώματα και τις αρμοδιότητές τους.

Η υλοποίηση ενός navigation bar γίνεται συνήθως με HTML και CSS για τη δομή και την εμφάνιση. Η JavaScript αναλαμβάνει τη δυναμική φόρτωση και την αλλαγή των στοιχείων ανάλογα με τις ενέργειες ή τα δεδομένα που παρέχονται από το backend. Σε πιο σύγχρονες διαδικτυακές εφαρμογές, το μενού πλοήγησης συνεργάζεται με τεχνικές όπως τα asynchronous requests ή η χρήση modular components, ώστε να ενσωματώνει λειτουργικότητα χωρίς την ανάγκη ανανέωσης της σελίδας, εξασφαλίζοντας μια πιο ομαλή και διαδραστική εμπειρία.

2.19 Αρθρωτό στοιχείο (modular component)

Η αρθρωτή σχεδίαση (modular design) είναι μια τεχνική σχεδίασης λογισμικού που βασίζεται στη διάσπαση της εφαρμογής σε μικρότερα, αυτόνομα και επαναχρησιμοποιήσιμα κομμάτια κώδικα (modular components). Κάθε modular component έχει συγκεκριμένη λειτουργία, όπως η εμφάνιση ενός πίνακα δεδομένων, ενός χάρτη ή ενός στατικού τμήματος πληροφοριών (π.χ. σελίδα "About") και μπορεί να ενσωματώνεται εύκολα σε διαφορετικά σημεία της εφαρμογής. Με τον τρόπο αυτόν, η ανάπτυξη και η συντήρηση του συστήματος γίνονται πιο ευέλικτες, επειδή οι αλλαγές σε ένα component δεν επηρεάζουν άμεσα τα υπόλοιπα [19].

Στις διαδικτυακές εφαρμογές, τα modular components υλοποιούνται συνήθως με JavaScript σε συνδυασμό με HTML και CSS και μπορούν να φορτώνονται δυναμικά με χρήση τεχνικών όπως το import/export. Ένα τέτοιο σύστημα επιτρέπει την αποδοτική οργάνωση του frontend, καθώς κάθε τμήμα της διεπαφής χρήστη αποτελεί ανεξάρτητο δομικό στοιχείο που μπορεί να επαναχρησιμοποιηθεί και να προσαρμοστεί.

Η χρησιμότητα των αρθρωτών στοιχείων είναι ιδιαίτερα εμφανής σε εφαρμογές που αντλούν δεδομένα από APIs, αφού κάθε component μπορεί να αναλαμβάνει τη συλλογή και παρουσίαση μιας συγκεκριμένης πληροφορίας (π.χ. λίστα καταστημάτων, προϊόντα, δρομολόγια λεωφορείων) μέσω asynchronous requests. Έτσι, επιτυγχάνεται καλύτερη οργάνωση του κώδικα, βελτιωμένη επεκτασιμότητα και πιο διαδραστική εμπειρία χρήστη.

2.20 Ενσωματωμένη εφαρμογή/πλαίσιο (widget/iframe)

Το ενσωματωμένο πλαίσιο (widget/iframe) είναι μια τεχνική που επιτρέπει την ενσωμάτωση έτοιμου περιεχομένου ή λειτουργικότητας από εξωτερικούς διακομιστές (servers) μέσα σε μια διαδικτυακή εφαρμογή. Με αυτόν τον τρόπο, η εφαρμογή μπορεί να προσφέρει στους χρήστες πρόσθετες δυνατότητες χωρίς να απαιτείται η εκ νέου ανάπτυξη των αντίστοιχων μηχανισμών. Για παράδειγμα, μέσω ενός iframe μπορεί να προβάλλεται ένας χάρτης από το Google Maps, μια φόρμα κράτησης, ή ακόμη και ένα βίντεο από το YouTube. Τα widgets συχνά χρησιμοποιούνται για την εμφάνιση πληροφοριών όπως ο καιρός, οι συναλλαγματικές ισοτιμίες ή οι τελευταίες ειδήσεις [20].

Η χρήση ενσωματωμένων πλαισίων ενισχύει την διαλειτουργικότητα (interoperability) των εφαρμογών, επιτρέποντας την αξιοποίηση υπηρεσιών από τρίτους παρόχους μέσω έτοιμων εργαλείων. Επιπλέον, παρέχει ευελιξία και επεκτασιμότητα, αφού η εφαρμογή μπορεί εύκολα να εμπλουτιστεί με νέο περιεχόμενο χωρίς τροποποίηση του βασικού κώδικα.

Στην περίπτωση διαδικτυακών εφαρμογών που στοχεύουν στην άμεση και δυναμική πληροφόρηση, τα widgets/iframes αποτελούν έναν απλό αλλά ισχυρό μηχανισμό εμπλουτισμού του περιβάλλοντος χρήστη. Μέσω αυτών, το frontend μπορεί να εμφανίζει δεδομένα σε πραγματικό χρόνο από εξωτερικές πηγές, βελτιώνοντας την εμπειρία και την αλληλεπίδραση του χρήστη με το σύστημα.

Κεφάλαιο 3ο: Προγράμματα Υποστήριξης και Τεχνολογίες Υλοποίησης

Η κατασκευή της διαδικτυακής εφαρμογής (web application) για το αεροδρόμιο Μακεδονία βασίζεται αποκλειστικά σε λογισμικό (software). Για την υλοποίησή της απαιτείται η συγγραφή κώδικα σε διάφορες γλώσσες προγραμματισμού, προκειμένου να καλυφθούν όλες οι λειτουργικές και διαδραστικές ανάγκες της εφαρμογής. Η χρήση πολλαπλών γλωσσών και τεχνολογιών επιτρέπει την ανάπτυξη ενός ολοκληρωμένου συστήματος που συνδυάζει την εμφάνιση, την επεξεργασία δεδομένων και την επικοινωνία με τον χρήστη με αποτελεσματικό και αξιόπιστο τρόπο.

3.1 Εργαλεία και Προγράμματα Υποστήριξης της Υλοποίησης

Για τη συγγραφή του κώδικα χρησιμοποιήθηκε το περιβάλλον ανάπτυξης κώδικα Visual Studio Code (VS Code) της εταιρείας Microsoft, το οποίο προσφέρει ευέλικτα εργαλεία, υποστήριξη πολλών γλωσσών προγραμματισμού και πληθώρα επεκτάσεων που διευκολύνουν την ανάπτυξη της εφαρμογής. Για την παρακολούθηση και διαχείριση της βάσης δεδομένων χρησιμοποιήθηκε το λογισμικό DB Browser for SQLite, το οποίο παρέχει ένα φιλικό γραφικό περιβάλλον για τη δημιουργία, την επεξεργασία και τον έλεγχο των δεδομένων, καθιστώντας τη διαχείριση της βάσης πιο απλή και αποτελεσματική..

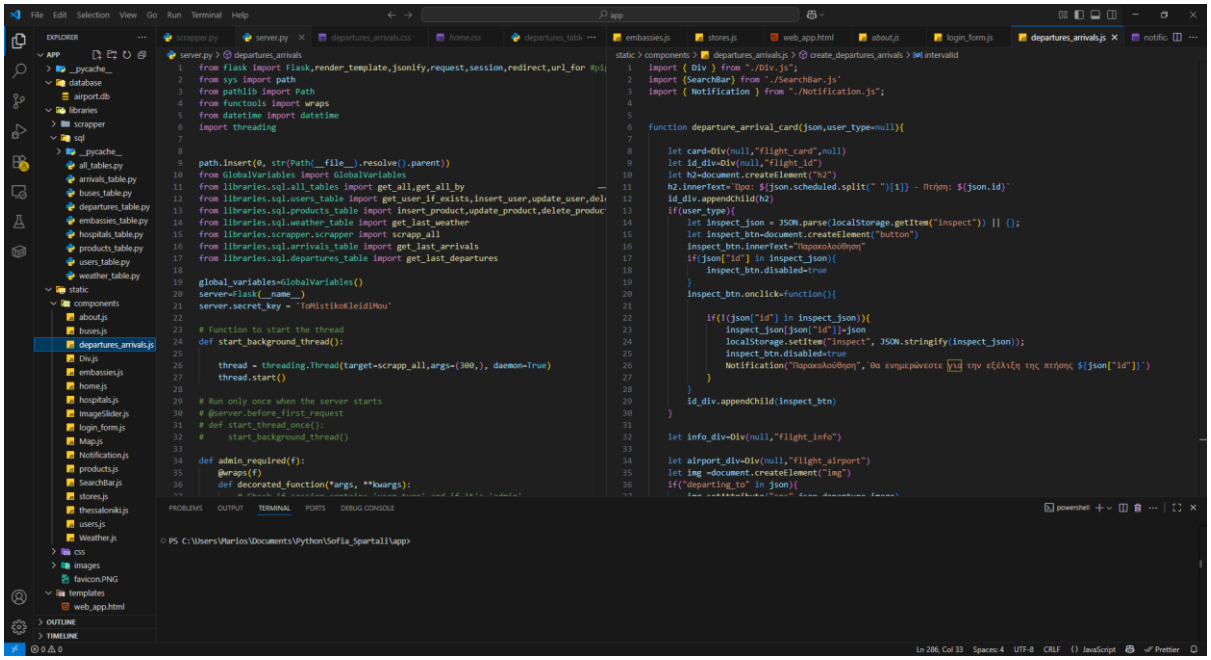
3.1.1 Visual Studio Code (VS Code)

Το Visual Studio Code (VS Code) είναι ένα ελαφρύ, αλλά ισχυρό περιβάλλον ανάπτυξης κώδικα (IDE) που έχει δημιουργηθεί από την εταιρεία Microsoft. Πρόκειται για ένα εργαλείο ανοιχτού κώδικα, διαθέσιμο για Windows, Linux και macOS, το οποίο υποστηρίζει μεγάλο αριθμό γλωσσών προγραμματισμού, όπως Python, JavaScript, C++, HTML, CSS και πολλές άλλες. Η ευελιξία του το καθιστά ιδανικό για την ανάπτυξη τόσο μικρών όσο και μεγάλων έργων λογισμικού [21].

Ένα από τα κύρια πλεονεκτήματα του VS Code είναι η υποστήριξη επεκτάσεων (extensions), που επιτρέπουν στους χρήστες να προσθέτουν λειτουργίες όπως εντοπισμό σφαλμάτων (debugging), αυτόματη συμπλήρωση κώδικα, μορφοποίηση, ενσωμάτωση με συστήματα ελέγχου εκδόσεων (π.χ. Git) και πολλά άλλα. Αυτό καθιστά το περιβάλλον πλήρως προσαρμόσιμο στις ανάγκες κάθε προγραμματιστή.

Το VS Code παρέχει επίσης προηγμένα χαρακτηριστικά όπως syntax highlighting, intellisense, multi-cursor editing, terminal ενσωματωμένο στο IDE και δυνατότητες παράλληλης εργασίας με πολλαπλά αρχεία και έργα ταυτόχρονα. Αυτά τα εργαλεία βελτιώνουν σημαντικά την παραγωγικότητα και διευκολύνουν τη διαχείριση πολύπλοκων έργων.

Επιπλέον, το VS Code υποστηρίζει ενσωμάτωση με εξωτερικά εργαλεία και βάσεις δεδομένων, διευκολύνοντας τη σύνδεση με υπηρεσίες web, τη διαχείριση βάσεων δεδομένων και την ανάπτυξη εφαρμογών που συνδυάζουν frontend και backend. Αυτό το καθιστά ένα ολοκληρωμένο εργαλείο για την ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών (web apps).



Εικόνα 3.1 Vscode

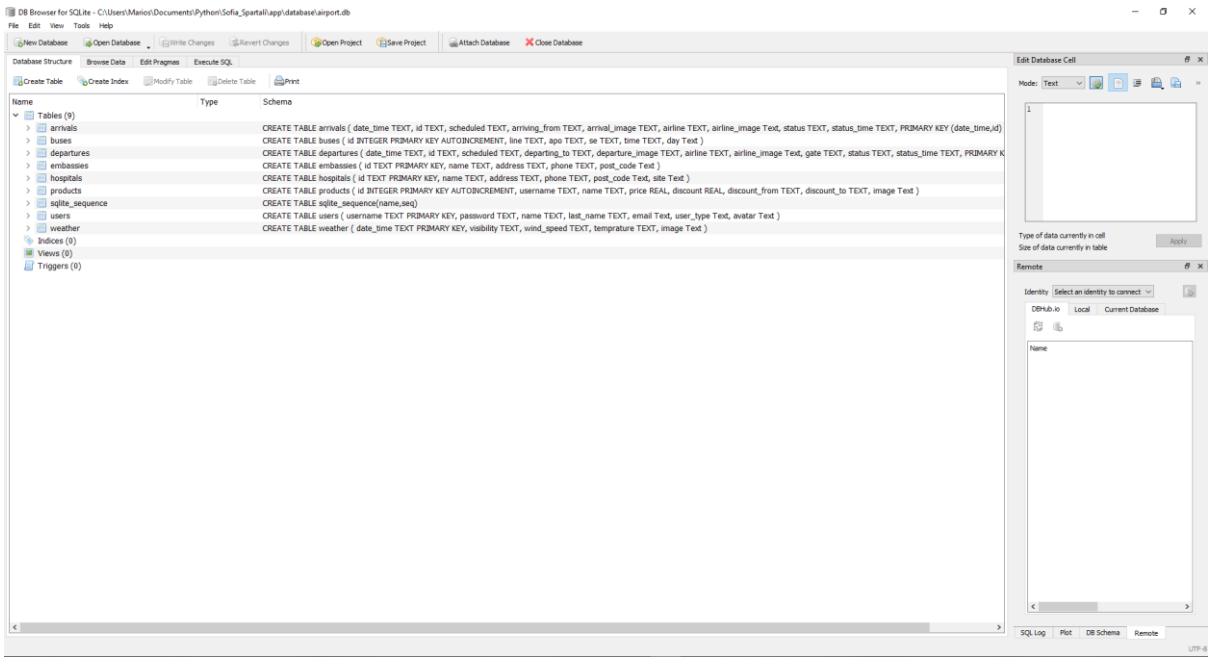
3.1.2 DB Browser for SQLite

Το DB Browser for SQLite είναι ένα ελεύθερο και ανοιχτού κώδικα εργαλείο που επιτρέπει τη δημιουργία, τη διαχείριση και την ανάλυση βάσεων δεδομένων SQLite μέσω γραφικού περιβάλλοντος. Πρόκειται για ένα φιλικό προς τον χρήστη εργαλείο, το οποίο καθιστά εύκολη τη διαχείριση των δεδομένων χωρίς να απαιτείται γνώση πολύπλοκων εντολών SQL [22].

Με το DB Browser for SQLite, οι χρήστες μπορούν να δημιουργούν νέες βάσεις δεδομένων, να εισάγουν και να τροποποιούν πίνακες, να εκτελούν ερωτήματα SQL και να εξάγουν ή να εισάγουν δεδομένα σε διάφορες μορφές. Αυτές οι δυνατότητες καθιστούν το εργαλείο ιδανικό για ανάπτυξη εφαρμογών που απαιτούν αξιόπιστη και εύκολα προσβάσιμη βάση δεδομένων.

Επιπλέον, το DB Browser for SQLite παρέχει οπτική αναπαράσταση των δεδομένων και δυνατότητες πλοήγησης μέσα στους πίνακες, διευκολύνοντας την ανάλυση και την παρακολούθηση των δεδομένων. Αυτό είναι ιδιαίτερα χρήσιμο για την ανάπτυξη και τον έλεγχο μικρών έως μεσαίων έργων λογισμικού, όπως διαδικτυακές εφαρμογές (web apps), που χρειάζονται γρήγορη και ασφαλή αποθήκευση πληροφοριών.

Το εργαλείο υποστηρίζει επίσης αποθήκευση και ανάκτηση δεδομένων με ασφαλή τρόπο, διατηρώντας την ακεραιότητα της βάσης και επιτρέποντας παράλληλη χρήση σε περιβάλλοντα ανάπτυξης και δοκιμών. Έτσι, αποτελεί ένα απαραίτητο εργαλείο για κάθε προγραμματιστή που επιθυμεί γρήγορη, αξιόπιστη και οπτικά κατανοητή διαχείριση βάσεων δεδομένων SQLite.



Εικόνα 3.2 DB Browser for SQLite

3.2 Τεχνολογίες Υλοποίησης

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν οι τεχνολογίες Python, JavaScript και SQLite. Η Python αποτέλεσε τη βασική γλώσσα προγραμματισμού για την ανάπτυξη της λογικής του συστήματος, προσφέροντας ευελιξία και απλότητα στον χειρισμό δεδομένων. Η JavaScript χρησιμοποιήθηκε για τη δυναμική αλληλεπίδραση στο περιβάλλον χρήστη, επιτρέποντας μια πιο φιλική και λειτουργική εμπειρία. Η SQLite επιλέχθηκε ως σύστημα διαχείρισης βάσης δεδομένων λόγω της ευκολίας χρήσης, της φορητότητας και της καταλληλότητάς της για ελαφριές εφαρμογές χωρίς την ανάγκη πολύπλοκης εγκατάστασης.

3.2.1 Python και Βιβλιοθήκες

Η Python [23] επιλέχθηκε γιατί αποτελεί μια γλώσσα προγραμματισμού που συνδυάζει απλότητα και ευκολία εκμάθησης, καθιστώντας την ιδανική για ακαδημαϊκή και εκπαιδευτική χρήση. Παράλληλα, διαθέτει μια πλούσια συλλογή από δωρεάν βιβλιοθήκες ανοιχτού κώδικα και ισχυρές δυνατότητες,

γεγονός που την καθιστά ιδιαίτερα κατάλληλη για την ανάπτυξη πρακτικών εφαρμογών, με ευελιξία και υψηλή αναγνωσιμότητα κώδικα.

Για την υλοποίηση της εργασίας αξιοποιήθηκαν διάφορες βιβλιοθήκες της Python, οι οποίες διευκόλυναν την ανάπτυξη και την οργάνωση της εφαρμογής. Συγκεκριμένα, χρησιμοποιήθηκαν η Flask για την υλοποίηση του web περιβάλλοντος και την επικοινωνία μέσω HTTP, η sqlite3 για τη διαχείριση της βάσης δεδομένων, η threading για την ταυτόχρονη εκτέλεση διεργασιών και την καλύτερη απόκριση του συστήματος και η Selenium, η οποία χρησιμοποιήθηκε για την αυτοματοποίηση ενεργειών και τον έλεγχο του browser (π.χ. Chrome) με σκοπό τη συλλογή δεδομένων (web scraping) από εξωτερικές ιστοσελίδες.

3.2.1.1 Διαχειριστής πακέτων (pip)

Ο pip είναι ο προεπιλεγμένος διαχειριστής πακέτων για την Python. Χρησιμοποιείται για την εγκατάσταση και τη διαχείριση λογισμικού πακέτων που έχουν γραφτεί σε Python. Αυτά τα πακέτα περιλαμβάνουν βιβλιοθήκες και άλλα εξαρτήματα που μπορεί να χρειαστούν για την ανάπτυξη και εκτέλεση προγραμμάτων σε Python. Είναι γραμμένος στην Python και είναι ο προτεινόμενος διαχειριστής πακέτων από τους κατασκευαστές της γλώσσας. Ο pip συνδέεται στην διαδικτυακή αποθήκη βιβλιοθηκών (Online-repository) με την ονομασία Python Package Index (PyPI) η οποία τον Μάιο του 2025 περιείχε περισσότερες από 530.000 δωρεάν βιβλιοθήκες έτοιμες για χρήση. Ο pip έχει την δυνατότητα να διαμορφωθεί και να τροποποιηθεί έτσι ώστε να μπορεί να συνδεθεί και σε άλλες αποθήκες βιβλιοθηκών στο διαδίκτυο ή τοπικά σε κάποιον ηλεκτρονικό υπολογιστή.

Ο pip εγκαθίσταται συνήθως μαζί με τη python και ο έλεγχος της ύπαρξής του γίνεται με χρήση της εντολής pip –version. Σε περίπτωση που δεν εγκατασταθεί μαζί με την Python, ο διαχειριστής μπορεί να εγκατασταθεί κατεβάζοντας και εκτελώντας το πρόγραμμα (script) get-pip.py από την επίσημη ιστοσελίδα του .

Χαρακτηριστικά του pip:

Εγκατάσταση Πακέτων: Ο pip επιτρέπει την εύκολη εγκατάσταση πακέτων από το Python Package Index (PyPI) και άλλες αποθετηριακές πηγές με χρήση της εντολής pip install <package_name>.

Αναβάθμιση Πακέτων: Μπορεί να αναβαθμίσει πακέτα στην πιο πρόσφατη διαθέσιμη έκδοση με χρήση της εντολής pip install --upgrade <package_name>.

Απεγκατάσταση Πακέτων: Ο pip επιτρέπει την αφαίρεση πακέτων που δεν είναι χρήσιμα πλέον με χρήση της εντολής pip uninstall <package_name>.

Λίστα Εγκατεστημένων Πακέτων: Ο χρήστης μπορεί να δει λίστα με όλα τα διαθέσιμα εγκατεστημένα πακέτα με χρήση της εντολής pip list .

Δημιουργία Αρχείων Requirements: Ο χρήστης μπορεί να δημιουργήσει ένα αρχείο requirements.txt που περιέχει όλα τα απαραίτητα πακέτα για το έργο του (project) με χρήση της εντολής pip freeze > requirements.txt.

Εγκατάσταση Αρχείων Requirements: Σε ήδη υπάρχοντα έργα (projects), ο χρήστης μπορεί να εγκαταστήσει τις απαραίτητες βιβλιοθήκες ενός έργου από υπάρχον αρχείο requirements.txt με χρήση της εντολής pip install –r requirements.txt.

Αναζήτηση Πακέτων: Ο χρήστης μπορεί να αναζητήσει πακέτα στο Python Package Index (PyPI) με χρήση της εντολής pip search <package_name>.

3.2.1.2 Βιβλιοθήκη Flask

Η Flask [24] είναι ένα ελαφρύ και ευέλικτο web framework της Python, σχεδιασμένο ώστε να διευκολύνει την ανάπτυξη εφαρμογών ιστού. Χάρη στην απλότητά της, επιτρέπει τη γρήγορη δημιουργία web εφαρμογών χωρίς περιττή πολυπλοκότητα, προσφέροντας παράλληλα τη δυνατότητα επέκτασης με πρόσθετα εργαλεία όταν απαιτείται.

Ένα από τα βασικά πλεονεκτήματα της Flask είναι η ευκολία μάθησης και χρήσης, κάτι που την καθιστά ιδιαίτερα δημοφιλή τόσο σε εκπαιδευτικό όσο και σε επαγγελματικό περιβάλλον. Η σύνταξη είναι κατανοητή και κοντά στη «φιλοσοφία της Python», γεγονός που επιτρέπει την ανάπτυξη καθαρού και ευανάγνωστου κώδικα. Ο τρόπος προγραμματισμού είναι κυρίως function-based, όπου κάθε endpoint ορίζεται ως ξεχωριστή συνάρτηση, διευκολύνοντας την οργάνωση και την αναγνωσιμότητα του κώδικα.

Επιπροσθέτως, η Flask είναι δωρεάν και ανοιχτού κώδικα, με μια μεγάλη κοινότητα που αναπτύσσει συνεχώς νέες επεκτάσεις. Παρέχει επίσης τη δυνατότητα προστασίας των endpoints μέσω συστημάτων login βασισμένων σε cookies, επιτρέποντας τον έλεγχο πρόσβασης των χρηστών και την ασφαλή διαχείριση sessions.

Εν κατακλείδι, χάρη στην απλότητά της και στη φορητότητά της, η Flask είναι κατάλληλη για μικρές έως μεσαίες εφαρμογές, παραμένοντας αρκετά ισχυρή για να υποστηρίξει πιο σύνθετα έργα, όταν συνδυάζεται με άλλες βιβλιοθήκες ή frameworks.

3.2.1.3 Βιβλιοθήκη Sqlite3

Η sqlite3 [25] είναι η ενσωματωμένη βιβλιοθήκη της Python για τη διαχείριση βάσεων δεδομένων SQLite, προσφέροντας άμεση και εύκολη πρόσβαση σε ελαφριές βάσεις δεδομένων χωρίς την ανάγκη εξωτερικής εγκατάστασης. Οι βάσεις δεδομένων αποθηκεύονται ως απλά αρχεία, γεγονός που διευκολύνει τη μεταφορά και την ενσωμάτωσή τους σε διαφορετικά περιβάλλοντα. Η sqlite3 είναι δωρεάν και ανοιχτού κώδικα· η ενσωμάτωσή της στην Python επιτρέπει τη γρήγορη δημιουργία, ανάγνωση, ενημέρωση και διαγραφή δεδομένων με απλό και κατανοητό τρόπο.

Η βιβλιοθήκη υποστηρίζει τη χρήση SQL εντολών με πλήρη έλεγχο συναλλαγών, καθιστώντας την κατάλληλη για εφαρμογές που απαιτούν αξιοπιστία και συνέπεια στα δεδομένα. Επιπλέον, η χρήση της sqlite3 σε συνδυασμό με thread-safe προσεγγίσεις επιτρέπει την ασφαλή εκτέλεση πολλαπλών διεργασιών ταυτόχρονα, διατηρώντας την απόδοση και τη σταθερότητα της εφαρμογής.

3.2.1.4 Βιβλιοθήκη Threading

Η threading [26] είναι μια βιβλιοθήκη της Python που επιτρέπει την ταυτόχρονη εκτέλεση πολλαπλών διεργασιών μέσα στην ίδια εφαρμογή, βελτιώνοντας την απόκριση και την αποδοτικότητα του συστήματος. Χρησιμοποιείται για εργασίες που μπορούν να τρέχουν παράλληλα, όπως η διαχείριση πολλαπλών χρηστών ή η ταυτόχρονη εκτέλεση background διαδικασιών, χωρίς να μπλοκάρει η κύρια ροή της εφαρμογής.

Η threading είναι δωρεάν και ενσωματωμένη στην Python, γεγονός που καθιστά εύκολη τη χρήση της χωρίς επιπλέον εγκαταστάσεις. Επιτρέπει τον συντονισμό και την επικοινωνία μεταξύ διαφορετικών threads μέσω μηχανισμών όπως locks και events, διασφαλίζοντας ότι τα κοινά δεδομένα παραμένουν ασφαλή. Η δυνατότητα αυτή είναι ιδιαίτερα χρήσιμη σε web εφαρμογές ή εφαρμογές με απαιτήσεις real-time επεξεργασίας, όπου η ταυτόχρονη εκτέλεση βελτιώνει σημαντικά την απόδοση και την εμπειρία του χρήστη.

3.2.1.5 Βιβλιοθήκη Selenium

Η Selenium είναι μία από τις πιο διαδεδομένες βιβλιοθήκες της Python για αυτοματοποίηση διαδικασιών σε φυλλομετρητές ιστού (web browsers). Χρησιμοποιείται ευρέως για τον έλεγχο και την προσομοίωση της αλληλεπίδρασης του χρήστη με ιστοσελίδες, καθιστώντας τη ιδανική για σκοπούς δοκιμών λογισμικού (testing) αλλά και για συλλογή δεδομένων (web scraping) από δυναμικές ιστοσελίδες. Με την Selenium είναι εφικτό να ανοιχτεί αυτόματα ένας browser, να συμπληρωθούν φόρμες, να πατηθούν κουμπιά και να πλοηγηθεί η εφαρμογή σε πολλαπλές σελίδες.

Ένα από τα σημαντικότερα πλεονεκτήματα της Selenium είναι ότι παρέχει πλήρη έλεγχο του browser, επιτρέποντας την εκτέλεση ενεργειών σαν να τις πραγματοποιούσε ένας πραγματικός χρήστης. Αυτό την καθιστά ιδιαίτερα χρήσιμη σε περιπτώσεις όπου τα δεδομένα μιας ιστοσελίδας φορτώνονται δυναμικά μέσω JavaScript και δεν είναι άμεσα διαθέσιμα στον απλό κώδικα HTML. Ο προγραμματιστής μπορεί να χρησιμοποιήσει εργαλεία όπως το ChromeDriver ή το GeckoDriver για να συνδέσει τη Selenium με browsers όπως το Google Chrome ή το Mozilla Firefox.

Επιπλέον, η Selenium είναι δωρεάν και ανοιχτού κώδικα, με μια πολύ δραστήρια κοινότητα που συνεισφέρει συνεχώς στη βελτίωσή της. Προσφέρει υποστήριξη για πολλές γλώσσες προγραμματισμού (π.χ. Python, Java, C#) και για διαφορετικά λειτουργικά συστήματα, γεγονός που την καθιστά ιδιαίτερα ευέλικτη και φορητή. Η απλότητά της στη χρήση και η δυνατότητα

αυτοματοποίησης σύνθετων σεναρίων πλοήγησης την καθιστούν απαραίτητο εργαλείο σε εκπαιδευτικά και επαγγελματικά έργα.

Η χρήση της Selenium σε εφαρμογές όπως η παρούσα εργασία δίνει τη δυνατότητα συνεχούς παρακολούθησης εξωτερικών ιστοσελίδων και λήψης δεδομένων σε πραγματικό χρόνο, εξασφαλίζοντας ότι η διαδικτυακή εφαρμογή τροφοδοτείται με τις πιο πρόσφατες πληροφορίες. Αυτό την καθιστά ιδιαίτερα χρήσιμη για έργα που απαιτούν ενημέρωση σε πραγματικό χρόνο, όπως τα συστήματα παρακολούθησης πτήσεων ή καιρού.

3.2.2 Javascript

Η JavaScript επιλέχθηκε διότι αποτελεί τη βασική γλώσσα για τη δημιουργία δυναμικών και διαδραστικών ιστοσελίδων. Χάρη στη μεγάλη κοινότητα υποστήριξης και τις πολυάριθμες δωρεάν βιβλιοθήκες ανοιχτού κώδικα, δίνει τη δυνατότητα ανάπτυξης φιλικών προς τον χρήστη διεπαφών και βελτιώνει σημαντικά την εμπειρία αλληλεπίδρασης με την εφαρμογή.

Επιπλέον, η JavaScript υποστηρίζει τη δημιουργία modular components, δηλαδή ανεξάρτητων και επαναχρησιμοποιήσιμων τμημάτων κώδικα που μπορούν να οργανώσουν καλύτερα τη δομή της εφαρμογής. Κάθε component μπορεί να περιλαμβάνει HTML, CSS και JavaScript λογική, διευκολύνοντας τη συντήρηση και την επεκτασιμότητα του έργου. Η εισαγωγή και η εξαγωγή αυτών των components γίνεται μέσω των import και export μεθόδων, επιτρέποντας την εύκολη σύνδεση και επαναχρησιμοποίηση κώδικα μεταξύ διαφορετικών αρχείων και τμημάτων της εφαρμογής. Επιπλέον, η JavaScript υποστηρίζει παραλληλία και ασύγχρονη επικοινωνία, όπως η ανανέωση περιεχομένου των components μέσω fetch requests, χωρίς να χρειάζεται επαναφόρτωση της σελίδας. Αυτό βελτιώνει την απόκριση και την εμπειρία του χρήστη, επιτρέποντας τη δυναμική ενημέρωση δεδομένων σε πραγματικό χρόνο.

Η επιλογή του Modular design ως βασικής μεθόδου σχεδιασμού της εφαρμογής έγινε αφού αποτελεί σύγχρονη και αποδεδειγμένα αποτελεσματική προσέγγιση στην ανάπτυξη εφαρμογών. Όλα τα δημοφιλή frameworks για web ανάπτυξη, όπως το React, που δημιουργήθηκε από το Facebook και το Angular, που αναπτύσσεται από την Google, βασίζονται σε αυτόν τον τρόπο προγραμματισμού, δημιουργώντας ανεξάρτητα και επαναχρησιμοποιήσιμα components. Η modular προσέγγιση διευκολύνει την οργάνωση του κώδικα, τη συντήρηση και την επεκτασιμότητα της εφαρμογής, επιτρέποντας στους προγραμματιστές να αναπτύσσουν σύνθετες και δυναμικές διεπαφές πιο αποτελεσματικά, χωρίς να μπερδεύονται με τον συνολικό όγκο της εφαρμογής.

3.2.3 SQLite

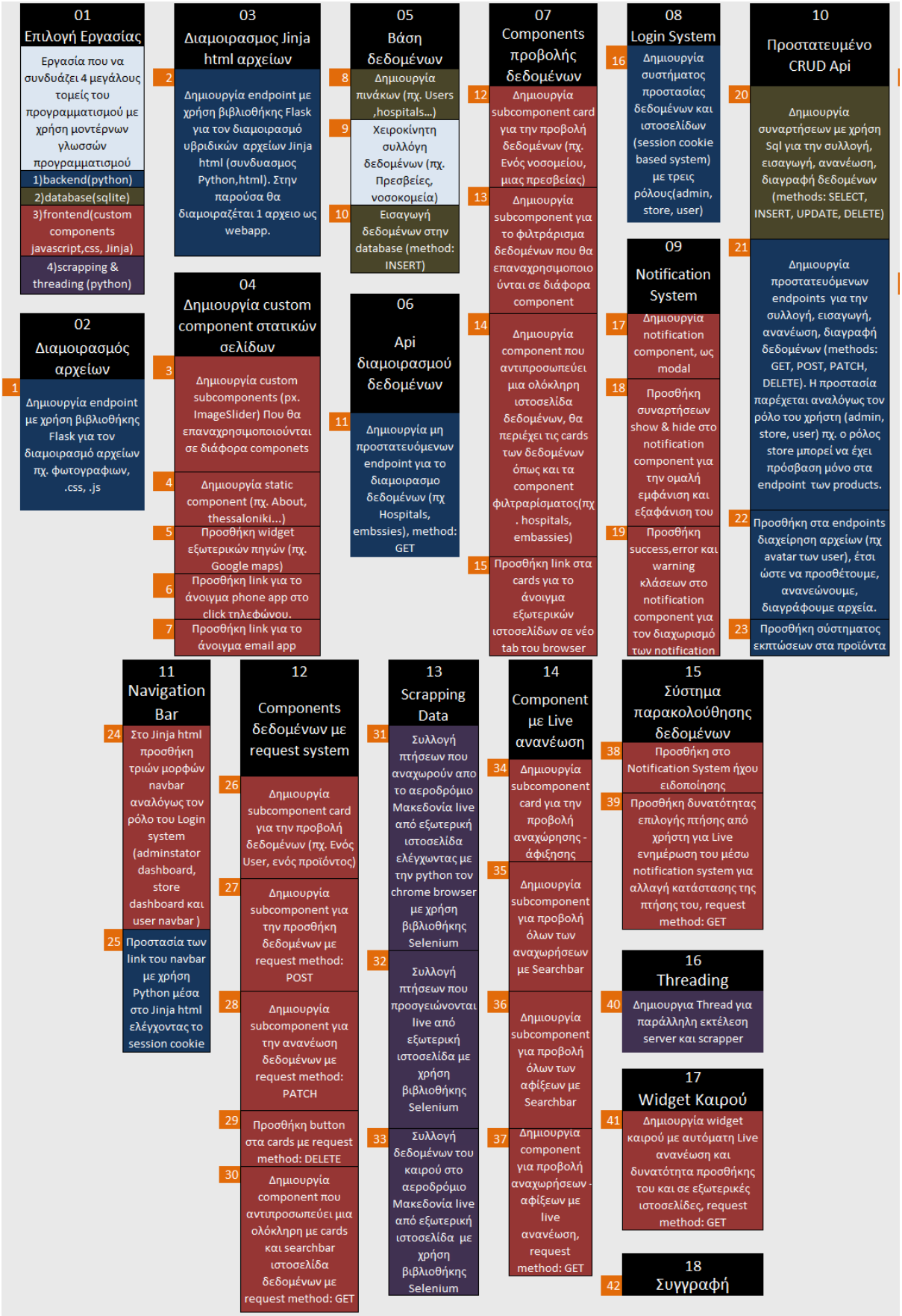
Η SQLite [27] επιλέχθηκε επειδή είναι ένα ελαφρύ και αποδοτικό σύστημα διαχείρισης βάσεων δεδομένων, το οποίο δεν απαιτεί περίπλοκη εγκατάσταση ή παραμετροποίηση. Οι βάσεις δεδομένων αποθηκεύονται ως απλά αρχεία, γεγονός που διευκολύνει σημαντικά τη μεταφορά και την

ενσωμάτωσή τους σε διαφορετικά περιβάλλοντα. Επιπλέον, είναι πλήρως δωρεάν και ανοιχτού κώδικα και στην Python η βιβλιοθήκη sqlite3 είναι διαθέσιμη αυτόματα, προσφέροντας άμεση και εύκολη διασύνδεση με τη βάση δεδομένων.

Κεφάλαιο 4ο: Το Σύστημα

4.1 Εισαγωγή – Διάγραμμα Δραστηριοτήτων

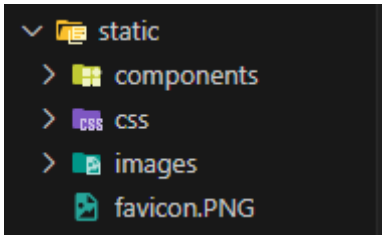
Πριν από την έναρξη της υλοποίησης της παρούσας εργασίας, καταρτίστηκε διάγραμμα δραστηριοτήτων (Εικόνα 4.1), το οποίο αποτύπωνε με σαφήνεια όλες τις διαδικασίες και ενέργειες που απαιτούνται για την ανάπτυξη της διαδικτυακής εφαρμογής. Οι δραστηριότητες ακολουθήθηκαν συστηματικά από την αρχή έως την ολοκλήρωση του έργου. Το συνολικό έργο διαχωρίστηκε σε 42 επιμέρους δραστηριότητες, οι οποίες οργανώθηκαν σε 18 θεματικές κατηγορίες.



Εικόνα 4.1 Διάγραμμα Δραστηριοτήτων

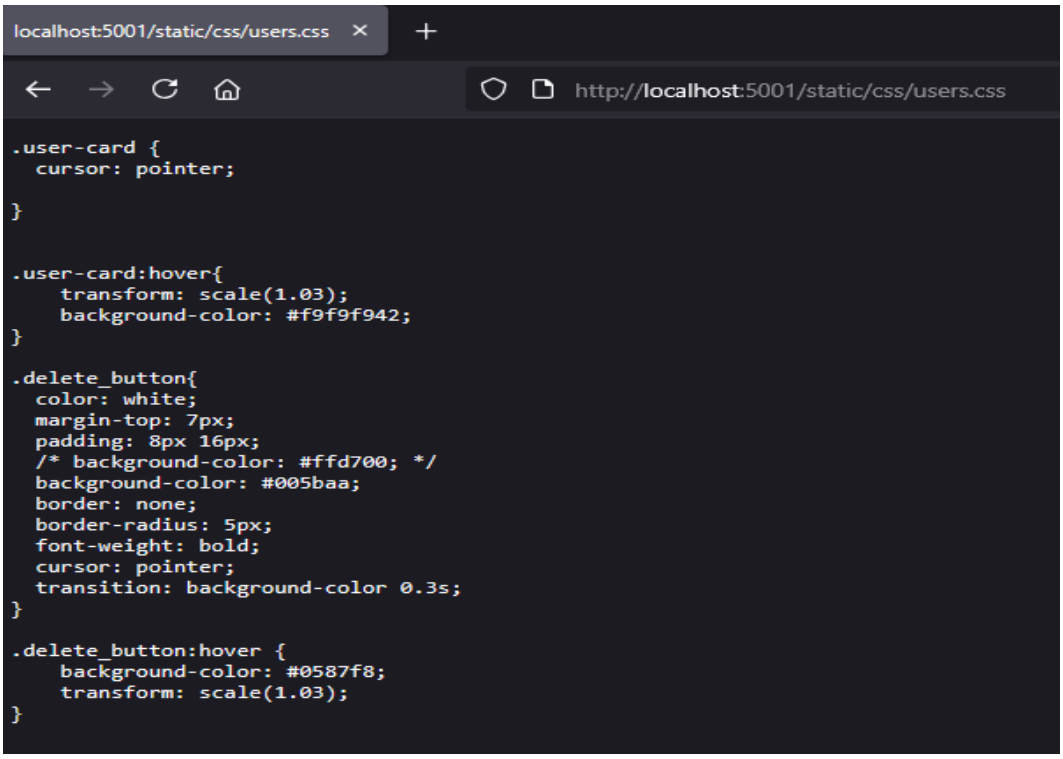
4.2 Διαμοιρασμός Αρχείων

Για τον διαμοιρασμό στατικών αρχείων, η βιβλιοθήκη Flask παρέχει αυτόματα το endpoint /static/, μέσω του οποίου εξυπηρετούνται όλα τα αρχεία που βρίσκονται στον φάκελο static (Εικόνα 4.2). Επιπλέον, τα αρχεία μπορούν να οργανωθούν σε υποφακέλους εντός του φακέλου static και η Flask δημιουργεί αυτόματα τα αντίστοιχα endpoints για κάθε υποφάκελο, π.χ. /static/css/. Η δυνατότητα αυτή διευκολύνει τη δομημένη οργάνωση των αρχείων και την αποτελεσματική διαχείριση των πόρων της εφαρμογής.



Εικόνα 4.2 Δομή φακέλου static

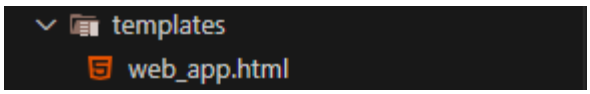
Τα στατικά αρχεία μπορούν να είναι οποιουδήποτε τύπου. Στην παρούσα εργασία, διαμοιράστηκαν εικόνες (.jpg, .png, .svg), αρχεία γλώσσας προγραμματισμού JavaScript (.js) και αρχεία μορφοποίησης (.css) (Εικόνα 4.3). Η οργανωμένη διαχείριση αυτών των αρχείων συνέβαλε στην αποτελεσματική λειτουργία και συντήρηση της διαδικτυακής εφαρμογής.



Εικόνα 4.3 Παράδειγμα διαμοιρασμού αρχείου .css

4.3 Διαμοιρασμός Jinja HTML Αρχείων

Για τον διαμοιρασμό δυναμικού περιεχομένου, η βιβλιοθήκη Flask χρησιμοποιεί τον φάκελο templates, μέσα στον οποίο τοποθετούνται τα αρχεία HTML. Στην παρούσα εργασία, διαμοιράζεται ένα αρχείο Jinja HTML, υλοπιώντας όλο το frontend της εφαρμογής με modular components. Η δομή αυτή επιτρέπει την επαναχρησιμοποίηση των στοιχείων της διεπαφής και διευκολύνει τη συντήρηση και επέκταση της διαδικτυακής εφαρμογής.



Εικόνα 4.4 Δομή φακέλου templates

Για τον διαμοιρασμό των αρχείων .html, η βιβλιοθήκη Flask δεν δημιουργεί αυτόματα κάποιο endpoint· η δημιουργία του απαιτείται να υλοποιηθεί από τον διαχειριστή του server. Για την επεξεργασία και αποστολή του υβριδικού HTML (με ενσωμάτωση Jinja templates), η Flask παρέχει την έτοιμη συνάρτηση render_template (Εικόνα 4.5). Στο πλαίσιο της παρούσας εργασίας, το αρχείο HTML αποδίδεται μέσω του endpoint /.

```
#endpoint that returns web_app.html
@server.route("/",methods=['GET'])
def home():
    return render_template("web_app.html")
```

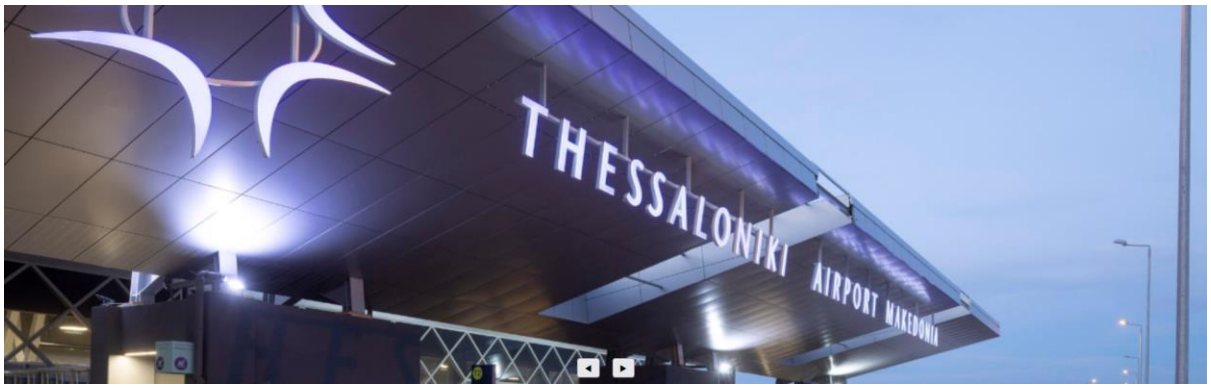
Εικόνα 4.5 Παράδειγμα διαμοιρασμού Jinja .html αρχείου σε endpoint /

4.4 Δημιουργία modular component στατικών σελίδων

Στην παρούσα εργασία, ο επισκέπτης της διαδικτυακής εφαρμογής έχει πρόσβαση σε components που αντιπροσωπεύουν στατικές σελίδες, δηλαδή σελίδες με περιεχόμενο το οποίο παραμένει αμετάβλητο και δεν διαφοροποιείται δυναμικά κατά την πλοήγηση.

4.4.1 Δημιουργία sub-component

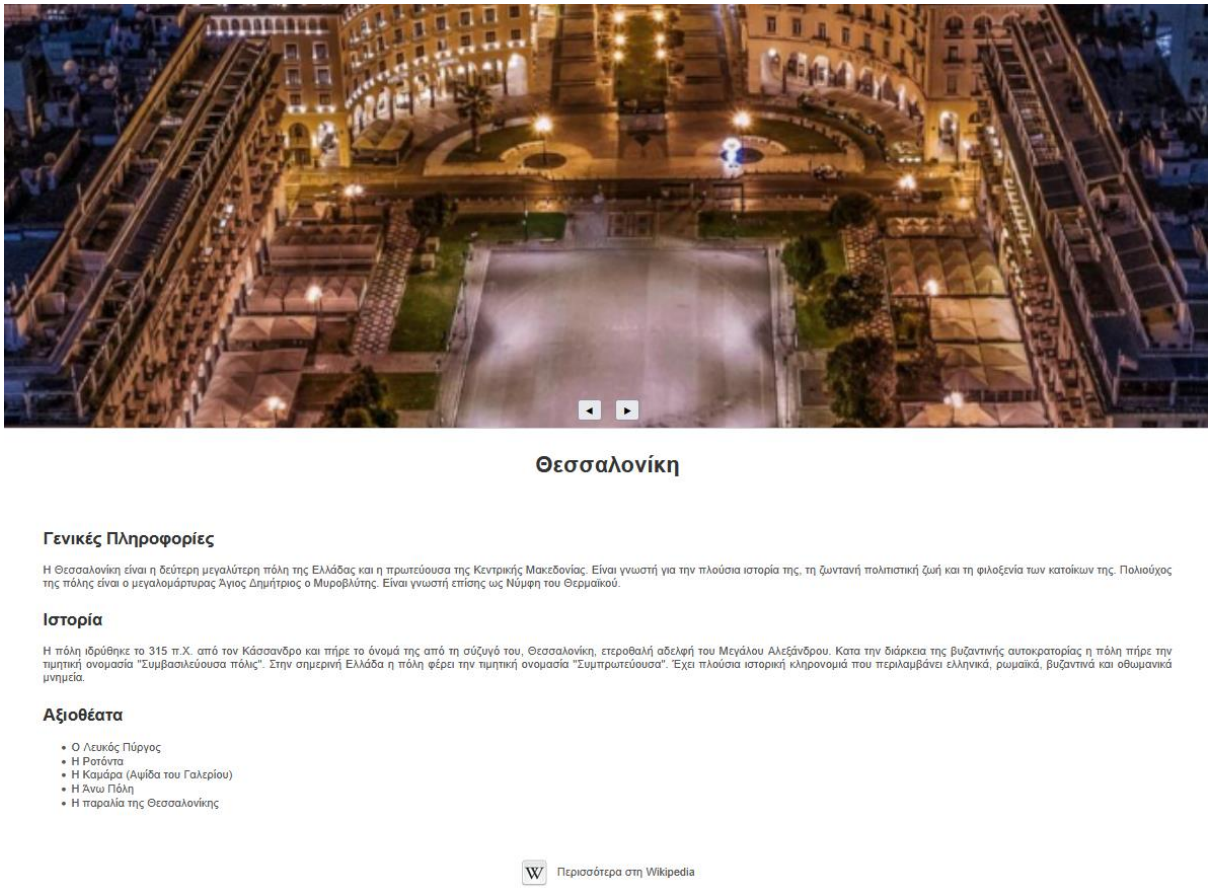
Στο πλαίσιο της υλοποίησης του frontend, δημιουργήθηκαν sub-components, τα οποία επαναχρησιμοποιούνται σε διαφορετικά components της εφαρμογής. Για παράδειγμα, το sub-component ImageSlider.js (Εικόνα 4.6) αξιοποιείται σε πολλαπλά σημεία του συστήματος, συμβάλλοντας τόσο στη βελτίωση της οργάνωσης του κώδικα όσο και στη διευκόλυνση της συντήρησης και επεκτασιμότητας της εφαρμογής.



Εικόνα 4.6 ImageSlider στο component home

4.4.2 Δημιουργία component ολόκληρων σελίδων

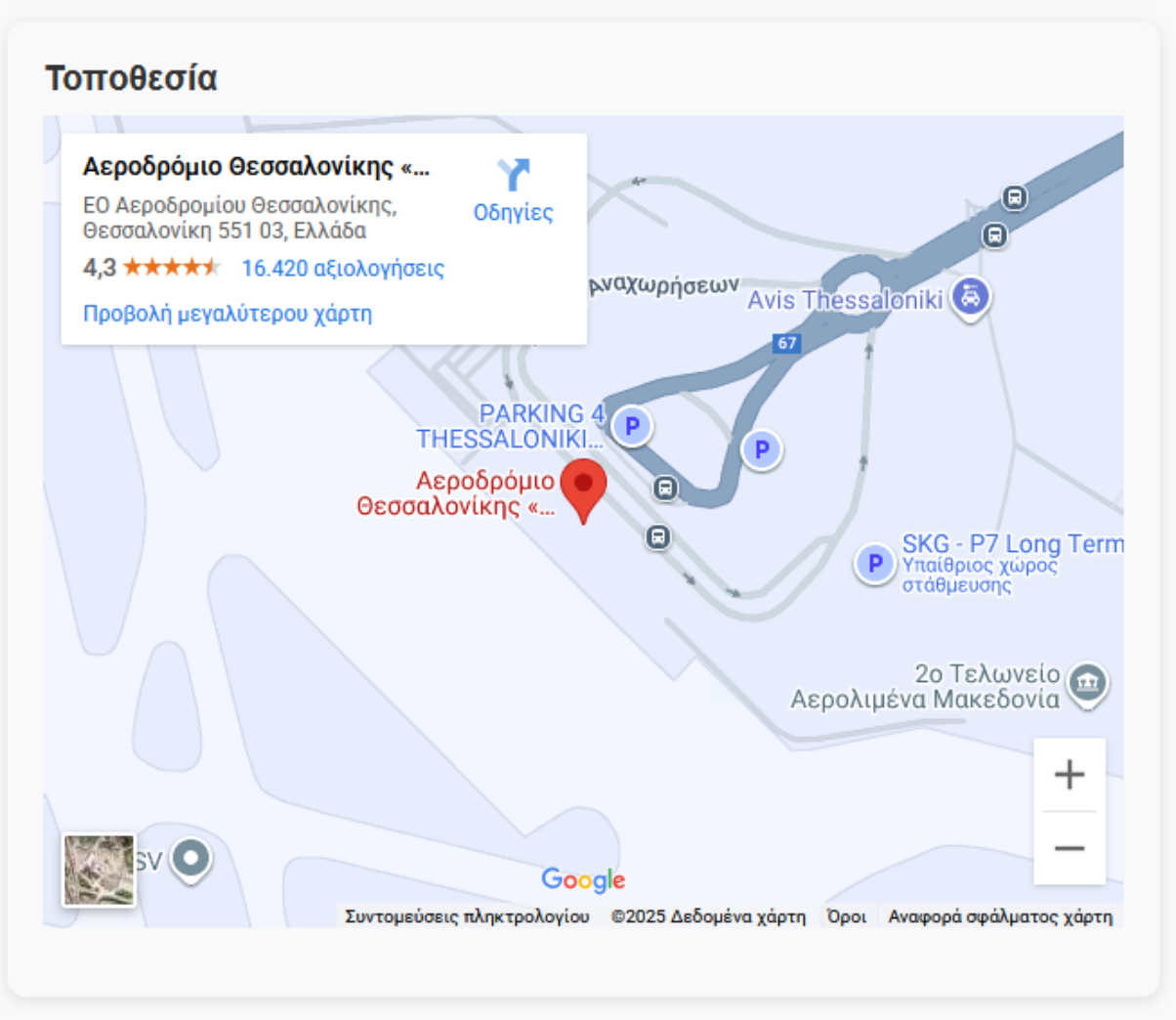
Για την υλοποίηση του frontend, δημιουργήθηκαν components στατικών σελίδων με στόχο την παρουσίαση πληροφοριών. Ενδεικτικά, το component about.js παρέχει πληροφορίες σχετικά με την εργασία και το component thessaloniki.js (Εικόνα 4.7) περιλαμβάνει πληροφορίες για την πόλη της Θεσσαλονίκης. Η χρήση τέτοιων components επιτρέπει την οργάνωση του περιεχομένου σε διακριτές ενότητες, ενισχύοντας τόσο την αναγνωσιμότητα όσο και τη συντηρησιμότητα της εφαρμογής.



Εικόνα 4.7 Component thessaloniki.js

4.4.3 Προσθήκη εξωτερικών πλαισίων (widget/iframe)

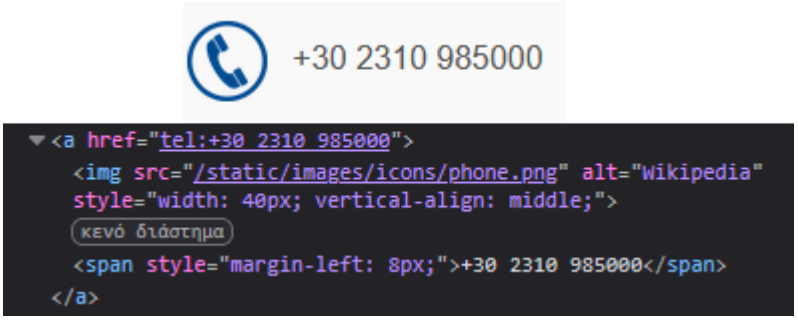
Στο πλαίσιο της ανάπτυξης του frontend, πραγματοποιήθηκε ενσωμάτωση πλαισίων (widget/iframes) σε επιλεγμένα components, τα οποία προβάλλουν έτοιμο περιεχόμενο που φιλοξενείται σε εξωτερικούς διακομιστές. Ενδεικτικά, ενσωματώθηκε χάρτης της Θεσσαλονίκης μέσω της υπηρεσίας Google Maps, εστιασμένος στο αεροδρόμιο «Μακεδονία» (Εικόνα 4.8). Η προσέγγιση αυτή επιτρέπει την παροχή δυναμικού και επικαιροποιημένου περιεχομένου χωρίς την ανάγκη επιπλέον υλοποίησης στο εσωτερικό της εφαρμογής.



Εικόνα 4.8 iframe Google Maps

4.4.4 Άνοιγμα phone app και email app

Επιπλέον, προστέθηκαν σύνδεσμοι (links) (Εικόνα 4.9) οι οποίοι επιτρέπουν την άμεση αλληλεπίδραση με εφαρμογές συστήματος της συσκευής του χρήστη. Συγκεκριμένα, στην επιλογή Email υλοποιήθηκε σύνδεσμος για το άνοιγμα της εφαρμογής ηλεκτρονικής αλληλογραφίας και στην επιλογή Τηλέφωνο προστέθηκε σύνδεσμος για την εκκίνηση της αντίστοιχης εφαρμογής κλήσεων. Η λειτουργικότητα αυτή διευκολύνει την επικοινωνία του χρήστη με τον φορέα της εφαρμογής, βελτιώνοντας παράλληλα την εμπειρία χρήσης.



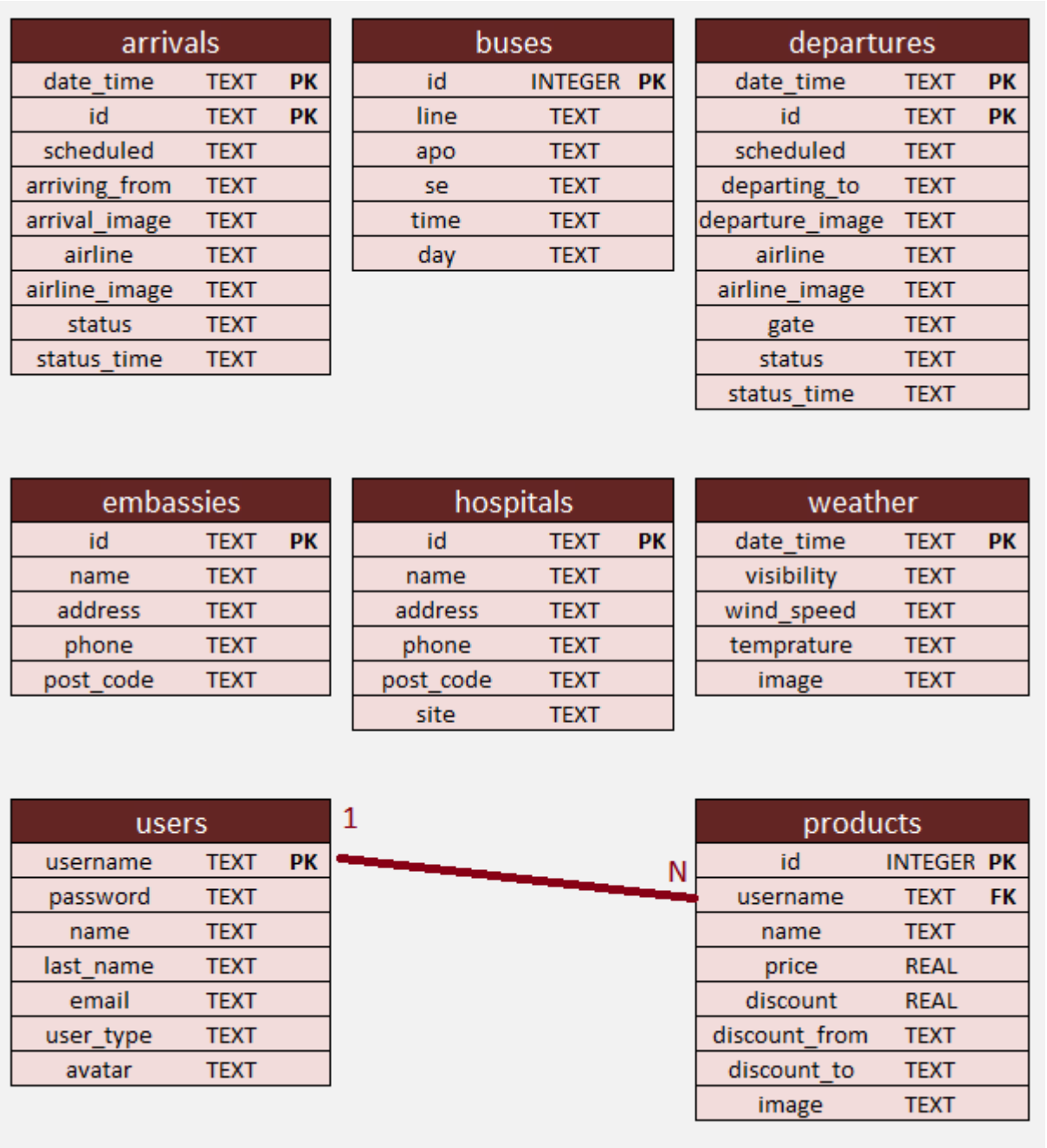
Εικόνα 4.9 Παράδειγμα συνδέσμου εκκίνησης εφαρμογής κλήσεων

4.5 Δημιουργία Βάσης Δεδομένων

Για την αποθήκευση και διαχείριση δεδομένων, αξιοποιήθηκε η βάση δεδομένων SQLite, η οποία επιλέχθηκε λόγω της απλότητας, της φορητότητας και της ενσωμάτωσής της στη γλώσσα προγραμματισμού Python. Στο πλαίσιο της παρούσας εργασίας, δημιουργήθηκαν πίνακες όπως arrivals, departures, buses, embassies, hospitals, weather, users και products. Η δημιουργία των πινάκων πραγματοποιήθηκε με τη χρήση κατάλληλων συναρτήσεων σε Python, οι οποίες αυτοματοποιούν τη διαδικασία και διασφαλίζουν τη σωστή αρχικοποίηση της βάσης δεδομένων. Η δομή αυτή επιτρέπει την οργανωμένη αποθήκευση των δεδομένων και την αποδοτική ανάκτησή τους κατά τη λειτουργία της διαδικτυακής εφαρμογής.

4.5.1 Διάγραμμα Οντοτήτων – Συσχετίσεων (ERD)

Επιπλέον, για την καλύτερη κατανόηση και οργάνωση της βάσης δεδομένων, κατασκευάστηκε Διάγραμμα Οντοτήτων – Συσχετίσεων (ERD) (Εικόνα 4.10), το οποίο αποτυπώνει τις σχέσεις μεταξύ των πινάκων. Στο παρόν μοντέλο, η μοναδική συσχέτιση αφορά τον πίνακα users και τον πίνακα products, όπου το πεδίο username του πίνακα users συνδέεται με το αντίστοιχο πεδίο username του πίνακα products. Η σχέση αυτή είναι 1–N (one-to-many), αφού κάθε χρήστης μπορεί να συνδέεται με πολλαπλά προϊόντα. Το διάγραμμα συμβάλλει στη σαφή αποτύπωση της δομής της βάσης και διευκολύνει τόσο την κατανόηση όσο και την περαιτέρω επέκτασή της.



Εικόνα 4.10 Διάγραμμα Οντοτήτων – Συσχετίσεων (ERD) βάσης δεδομένων

4.5.2 Χειροκίνητη συλλογή δεδομένων

Για την τροφοδότηση της βάσης δεδομένων με πληροφορίες, πραγματοποιήθηκε χειροκίνητη συλλογή δεδομένων από διαδικτυακές πηγές. Συγκεκριμένα, συγκεντρώθηκαν στοιχεία που αφορούν τις πρεσβείες, τα νοσοκομεία και τις γραμμές λεωφορείων του ΟΑΣΘ στη Θεσσαλονίκη. Τα δεδομένα αυτά ενσωματώθηκαν στη βάση μέσω της μεθόδου INSERT, η οποία επέτρεψε την αρχικοποίηση των αντίστοιχων πινάκων. Η διαδικασία αυτή διασφάλισε την ακρίβεια και αξιοπιστία των πληροφοριών που είναι διαθέσιμες μέσω της διαδικτυακής εφαρμογής.

4.6 API Διαμοιρασμού δεδομένων

Για την αποτελεσματική επικοινωνία μεταξύ του frontend και του backend της διαδικτυακής εφαρμογής, υλοποιήθηκαν APIs διαμοιρασμού δεδομένων. Τα APIs παρέχουν καθορισμένα endpoints μέσω των οποίων τα δεδομένα της εφαρμογής μπορούν να ανακτηθούν ή να ενημερωθούν με τυποποιημένο τρόπο, ανεξαρτήτως της πλατφόρμας του χρήστη. Η χρήση αυτής της προσέγγισης επιτρέπει την ασφαλή, αποδοτική και ευέλικτη διάθεση των πληροφοριών, διατηρώντας τη δομή της εφαρμογής modular και επεκτάσιμη.

Αρχικά, δημιουργήθηκαν μη προστατευόμενα endpoints για το διαμοιρασμό δεδομένων που αφορούν πρεσβείες, νοσοκομεία και γραμμές λεωφορείων. Τα δεδομένα αυτά παρέχονται υπό μορφή λίστας σε JSON format (Εικόνα 4.11), επιτρέποντας την άμεση πρόσβαση και επεξεργασία τους από το frontend ή από άλλες εφαρμογές. Η χρήση JSON εξασφαλίζει την τυποποιημένη και ευέλικτη αναπαράσταση των δεδομένων, ταυτοχρόνως η ύπαρξη endpoints διευκολύνει την ολοκληρωμένη λειτουργία της διαδικτυακής εφαρμογής.

Εικόνα 4.11 Παράδειγμα μη προστατευμένου endpoint /embassies

4.7 Components προβολής δεδομένων

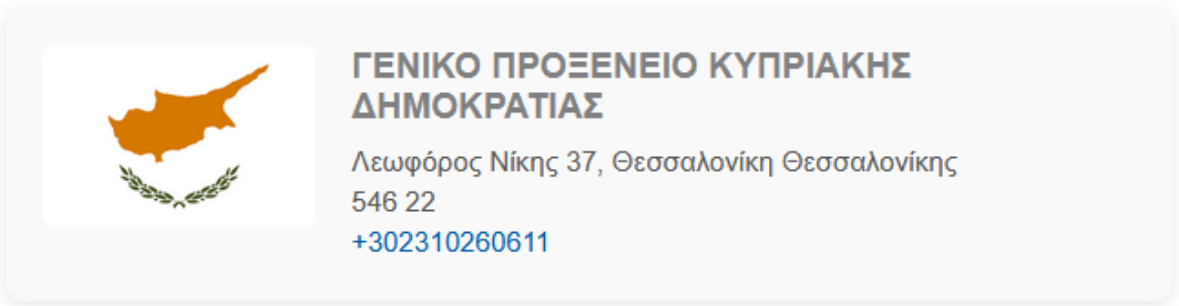
Στο frontend της εφαρμογής, δημιουργήθηκαν components για την προβολή δεδομένων, τα οποία οργανώνονται σύμφωνα με το modular design ώστε να διευκολύνουν την επαναχρησιμοποίηση και την επεκτασιμότητα. Κάθε component αποτελείται από δύο βασικά υποσυστατικά (sub-components):

1) SearchBar, που επιτρέπει στο χρήστη την αναζήτηση συγκεκριμένων στοιχείων εντός της λίστας δεδομένων.
2) Λίστα με Cards, όπου κάθε card παρουσιάζει ένα αντικείμενο (Item) από τη λίστα JSON που παρέχεται μέσω των μη προστατευόμενων endpoints.

Η δομή αυτή επιτρέπει την άμεση, δυναμική και ευανάγνωστη παρουσίαση των δεδομένων στον χρήστη, εξασφαλίζοντας παράλληλα ευελιξία για μελλοντικές επεκτάσεις ή τροποποιήσεις.

4.7.1 Subcomponent Card

Για την πιο λεπτομερή και επαναχρησιμοποιήσιμη παρουσίαση των δεδομένων, δημιουργήθηκε ένα sub-component τύπου Card (Εικόνα 4.12). Κάθε card είναι υπεύθυνη για την προβολή των πληροφοριών ενός μεμονωμένου στοιχείου της λίστας JSON, όπως για παράδειγμα ενός νοσοκομείου, μιας πρεσβείας ή των ωρών δρομολογίων των λεωφορείων που εξυπηρετούν το αεροδρόμιο. Η χρήση αυτού του sub-component διασφαλίζει ομοιομορφία στην εμφάνιση των δεδομένων, απλοποιεί τη συντήρηση του κώδικα και επιτρέπει την εύκολη ενσωμάτωσή του σε διαφορετικά components προβολής.



Εικόνα 4.12 Παράδειγμα Subcomponent Card Προξενείου

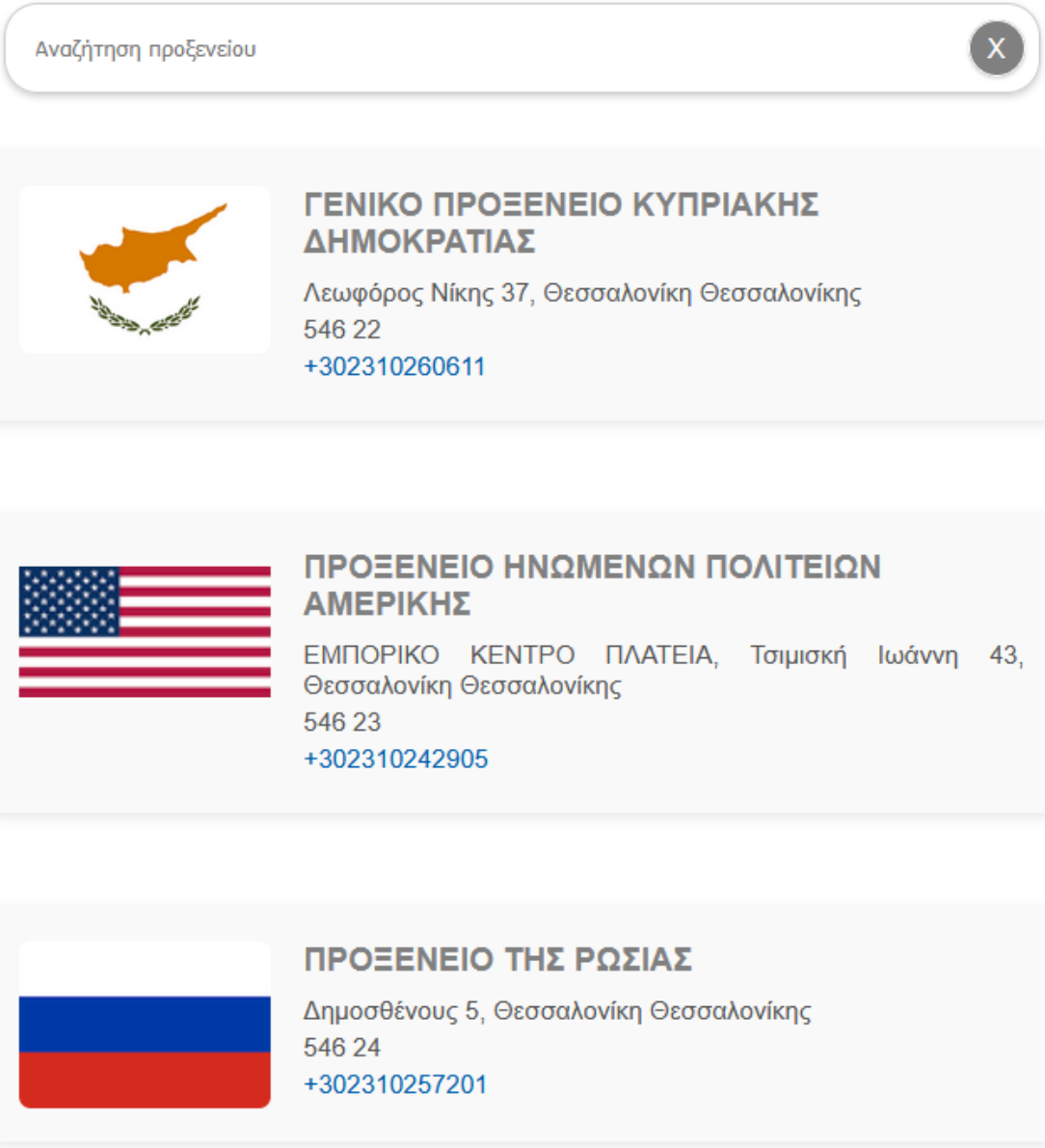
4.7.2 Subcomponent SearchBar

Για την υποστήριξη της δυναμικής αναζήτησης και φιλτραρίσματος δεδομένων, δημιουργήθηκε ένα sub-component τύπου SearchBar (Εικόνα 4.13), το οποίο είναι επαναχρησιμοποιήσιμο σε διάφορα components. Το SearchBar λειτουργεί αναζητώντας μέσα στη λίστα JSON των δεδομένων. Κάθε αντικείμενο JSON μετατρέπεται σε string και συγκρίνεται με το κείμενο που εισάγεται στο search bar, με τρόπο μη ευαίσθητο σε πεζά-κεφαλαία (case-insensitive). Αντιστοίχως, τα cards που παρουσιάζουν τα αντικείμενα εμφανίζονται ή αποκρύπτονται δυναμικά ανάλογα με το αποτέλεσμα της αναζήτησης.

Εικόνα 4.13 SearchBar αναζήτησης προξενείων

4.7.3 Υλοποίηση Component Ιστοσελίδας με Ενσωμάτωση Δεδομένων από μη προστατευόμενα endpoints

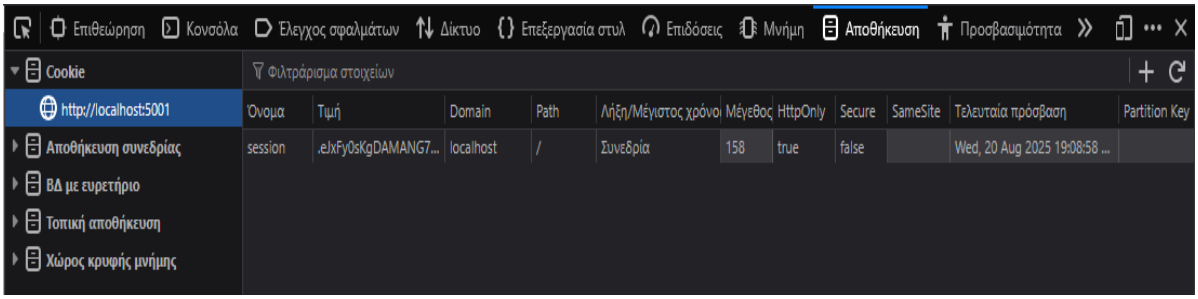
Στο πλαίσιο της αρχιτεκτονικής του frontend, δημιουργήθηκε ένα component που αντιπροσωπεύει μια ολόκληρη ιστοσελίδα δεδομένων (Εικόνα 4.14), η οποία αντλεί πληροφορίες από τα μη προστατευόμενα endpoints (π.χ. πρεσβείες, νοσοκομεία, λεωφορεία). Το συγκεκριμένο component ενσωματώνει τα sub-components Cards, που είναι υπεύθυνα για την παρουσίαση των επιμέρους στοιχείων της λίστας JSON και το SearchBar, το οποίο επιτρέπει τη δυναμική αναζήτηση και φιλτράρισμα των δεδομένων. Με αυτόν τον τρόπο, κάθε σελίδα δεδομένων παρέχει στον χρήστη μια οργανωμένη, ευανάγνωστη και διαδραστική προβολή των διαθέσιμων πληροφοριών, συνδυάζοντας λειτουργικότητα και ευχρηστία.



Εικόνα 4.14 Component Ιστοσελίδας Πρεσβειών

4.8 Σύστημα προστασίας δεδομένων με ρόλους (Login System)

Για την προστασία των δεδομένων και τον έλεγχο πρόσβασης στις ιστοσελίδες της εφαρμογής, υλοποιήθηκε ένα σύστημα διαχείρισης συνεδρίας (session cookie based system). Το σύστημα αυτό βασίζεται στη χρήση session cookies (Εικόνα 4.15), τα οποία αποθηκεύονται προσωρινά στον φυλλομετρητή (browser) του χρήστη και επιτρέπουν την ασφαλή ταυτοποίηση και διαχείριση των δικαιωμάτων του κατά τη διάρκεια της σύνδεσής του στην εφαρμογή. Με αυτόν τον τρόπο, εξασφαλίζεται ότι μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση σε ευαίσθητες πληροφορίες και σελίδες.



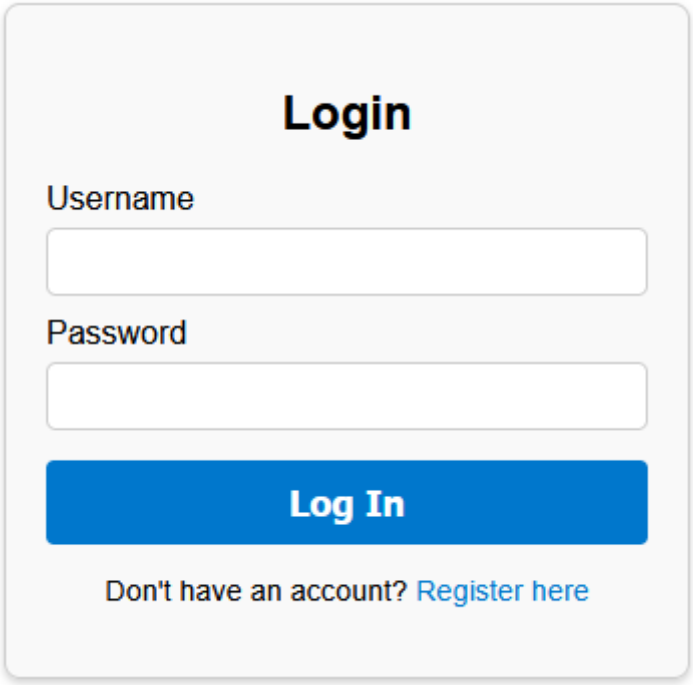
Εικόνα 4.15 Σύστημα διαχείρισης συνεδρίας (session cookie)

Η υλοποίηση του συστήματος στηρίζεται σε τρία διαφορετικά επίπεδα ρόλων:

- Διαχειριστής (admin): διαθέτει πλήρη πρόσβαση σε όλα τα δεδομένα και τις λειτουργίες της εφαρμογής. Έχει τη δυνατότητα να διαχειρίζεται χρήστες, να επεξεργάζεται δεδομένα και να εποπτεύει τη σωστή λειτουργία του συστήματος.
- Κατάστημα (store): έχει πρόσβαση σε λειτουργίες που σχετίζονται με την καταχώριση, διαχείριση προϊόντων και το σύστημα εκπτώσεων αυτών, περιοριζόμενος σε ενέργειες που αφορούν τον ρόλο του.
- Χρήστης (user): διαθέτει τα βασικά δικαιώματα πλοήγησης και προβολής δεδομένων, με δυνατότητες περιορισμένες στην κατανάλωση της πληροφορίας και όχι στη διαχείρισή της. Επιπλέον, ο χρήστης έχει τη δυνατότητα να επιλέγει μία ή περισσότερες πτήσεις (αφίξεις ή αναχωρήσεις) και να λαμβάνει live ενημερώσεις σχετικά με την κατάσταση της/των πτήσης/πτήσεων που έχει επιλέξει. Η λειτουργία αυτή ενισχύει την αλληλεπίδραση με την εφαρμογή και παρέχει στον χρήστη προσωποποιημένη πληροφόρηση σε πραγματικό χρόνο.

Η ύπαρξη διακριτών ρόλων σε συνδυασμό με τη χρήση session cookies προσφέρει πολυεπίπεδο μηχανισμό ασφαλείας, διασφαλίζοντας τόσο την ακεραιότητα των δεδομένων όσο και την ομαλή λειτουργία της εφαρμογής. Παράλληλα, η συγκεκριμένη αρχιτεκτονική είναι επεκτάσιμη, επιτρέποντας την προσθήκη νέων ρόλων ή την προσαρμογή των δικαιωμάτων πρόσβασης ανάλογα με τις ανάγκες μελλοντικής εξέλιξης του συστήματος.

Για την πρόσβαση στο σύστημα υλοποιήθηκε ειδικό Login component (Εικόνα 4.16), μέσω του οποίου οι χρήστες εισάγουν τα διαπιστευτήριά τους (όνομα χρήστη και κωδικό πρόσβασης) προκειμένου να συνδεθούν. Το component αυτό αποτελεί το σημείο εκκίνησης της διαδικασίας αυθεντικοποίησης, εξασφαλίζοντας ότι κάθε χρήστης αποκτά τα δικαιώματα που αντιστοιχούν στον ρόλο του (admin, store, user).



Εικόνα 4.16 Component Login

4.9 Notification System

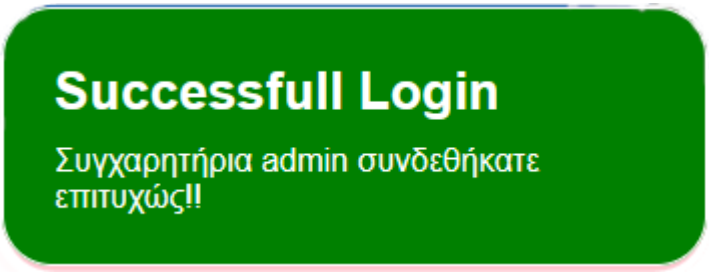
Στο πλαίσιο της εφαρμογής αναπτύχθηκε σύστημα ειδοποιήσεων (notification system), το οποίο παρέχει στους χρήστες ενημερώσεις σε πραγματικό χρόνο για σημαντικά γεγονότα. Οι ειδοποιήσεις εμφανίζονται δυναμικά στο περιβάλλον χρήστη και αφορούν, ενδεικτικά, αλλαγές στην κατάσταση πτήσεων, μηνύματα συστήματος ή ενημερώσεις που σχετίζονται με τις επιλογές του χρήστη. Η λειτουργικότητα αυτή βελτιώνει την εμπειρία χρήσης, ενισχύει την αλληλεπίδραση με την εφαρμογή και συμβάλλει στην έγκαιρη πρόσβαση σε κρίσιμες πληροφορίες.

4.9.1 Component Notification

Για την υλοποίηση του συστήματος ειδοποιήσεων αναπτύχθηκε ειδικό notification component, το οποίο εμφανίζεται με τη μορφή modal παραθύρου. Η επιλογή του modal σχεδιασμού επιτρέπει την άμεση και ευδιάκριτη προβολή κρίσιμων ενημερώσεων, χωρίς να αποσπάται η προσοχή του χρήστη από την υπόλοιπη διεπαφή. Το component ενεργοποιείται δυναμικά όταν προκύπτει νέα ειδοποίηση (π.χ. αλλαγή κατάστασης πτήσης), προσφέροντας έτσι έναν αποδοτικό μηχανισμό επικοινωνίας μεταξύ συστήματος και χρήστη.

Για την καλύτερη οργάνωση και κατηγοριοποίηση των ειδοποιήσεων, το notification component επεκτάθηκε με την προσθήκη εξειδικευμένων κλάσεων, όπως success, error και warning. Η διάκριση αυτή επιτρέπει την οπτική διαφοροποίηση των μηνυμάτων, παρέχοντας στον χρήστη άμεση κατανόηση του περιεχομένου της ειδοποίησης. Συγκεκριμένα, οι ειδοποιήσεις τύπου success (Εικόνα 4.17) χρησιμοποιούνται για την επιβεβαίωση επιτυχών ενεργειών (π.χ. ολοκλήρωση καταχώρησης), οι error για την αναφορά προβλημάτων ή αποτυχιών και οι warning λειτουργούν ως προειδοποιητικά μηνύματα για πιθανές ανωμαλίες ή καταστάσεις που απαιτούν προσοχή. Η υλοποίηση αυτή ενισχύει

τη χρηστικότητα του συστήματος, επιτρέποντας στους χρήστες να λαμβάνουν πιο άμεση και κατανοητή πληροφόρηση.



Εικόνα 4.17 Παράδειγμα success notification

4.10 Προστατευμένο CRUD API

Για τη διαχείριση των δεδομένων και αρχείων υλοποιήθηκε προστατευμένο CRUD API (Create–Read–Update–Delete), στο οποίο η πρόσβαση επιτρέπεται μόνο σε αυθεντικοποιημένους χρήστες μέσω του υφιστάμενου session-based συστήματος. Η αυθεντικοποίηση πραγματοποιείται κατά τη σύνδεση και η εξουσιοδότηση εφαρμόζεται με Role-Based Access Control (RBAC) βάσει των ρόλων admin, store και user.

4.10.1 Επικοινωνία εφαρμογής με βάση δεδομένων

Για την επικοινωνία της εφαρμογής με τη βάση δεδομένων υλοποιήθηκαν Python συναρτήσεις που αξιοποιούν SQL queries για τη διαχείριση των δεδομένων. Συγκεκριμένα, δημιουργήθηκαν μέθοδοι που καλύπτουν όλες τις βασικές λειτουργίες ενός CRUD API:

- SELECT για την αναζήτηση και συλλογή δεδομένων από τους πίνακες (π.χ. λίστα user).
- INSERT για την εισαγωγή νέων εγγραφών, μέσω φόρμας χρήστη.
- UPDATE για την ανανέωση υπαρχόντων δεδομένων.
- DELETE για τη διαγραφή εγγραφών που δεν ισχύουν πλέον.

Οι συναρτήσεις αυτές υλοποιήθηκαν με παραμετροποιημένα ερωτήματα (parameterized queries), ώστε να διασφαλίζεται η προστασία από SQL injection και να διατηρείται η ακεραιότητα της βάσης. Επιπλέον, η υλοποίηση περιλαμβάνει χειρισμό εξαιρέσεων (error handling) και κατάλληλα return messages για την ενημέρωση του διαχειριστή ή των endpoints σχετικά με το αποτέλεσμα της πράξης.

4.10.2 Προστατευόμενα endpoints

Στην παρούσα εργασία, αναπτύχθηκαν προστατευόμενα endpoints για τη διαχείριση δεδομένων και αρχείων της διαδικτυακής εφαρμογής, επιτρέποντας λειτουργίες συλλογής, εισαγωγής, ανανέωσης και διαγραφής μέσω των HTTP μεθόδων GET, POST, PATCH και DELETE, που καλύπτουν όλες τις βασικές λειτουργίες ενός CRUD API. Η πρόσβαση σε κάθε endpoint ελέγχεται αυστηρά βάσει του ρόλου του χρήστη (admin, store, user), διασφαλίζοντας ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν συγκεκριμένες ενέργειες. Σε περίπτωση που κάποιος χρήστης προσπαθήσει να προσπελάσει ένα endpoint χωρίς τον απαιτούμενο ρόλο, εφαρμόζεται αυτόματα ανακατεύθυνση (Redirection) στην αρχική σελίδα της εφαρμογής, εξασφαλίζοντας την ασφάλεια και την ακεραιότητα των δεδομένων και αρχείων.

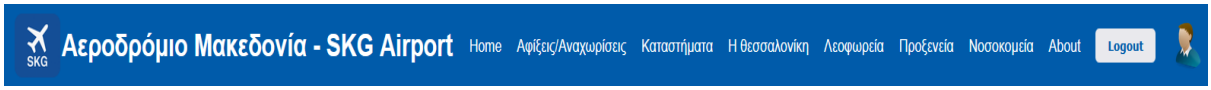
4.11 Navigation Bar

Το navigation bar αποτελεί ένα από τα πιο σημαντικά στοιχεία της διαδικτυακής εφαρμογής, λειτουργώντας ως βασικό μέσο πλοήγησης και επιτρέπει στον χρήστη να μετακινείται άμεσα και εύκολα ανάμεσα στις διαφορετικές ενότητες και λειτουργίες του συστήματος. Στο πλαίσιο της παρούσας εργασίας υλοποιήθηκαν τρεις εκδοχές navigation bar, προσαρμοσμένες στον εκάστοτε ρόλο χρήστη: administrator dashboard (Εικόνα 4.18), store dashboard (Εικόνα 4.19) και user navigation bar (Εικόνα 4.20). Η επιλογή του κατάλληλου navigation bar πραγματοποιείται δυναμικά κατά το rendering της σελίδας, μέσω ελέγχου του ρόλου του χρήστη που αποθηκεύεται στο session cookie και διαχειρίζεται από τον Flask server.

Επιπλέον, όλα τα links που περιλαμβάνονται στο navigation bar προστατεύονται με server-side έλεγχο πρόσβασης, εξασφαλίζοντας ότι μόνο οι χρήστες με τα κατάλληλα δικαιώματα μπορούν να αποκτήσουν πρόσβαση στις αντίστοιχες σελίδες. Με τον τρόπο αυτό, διασφαλίζεται τόσο η ασφάλεια και ακεραιότητα του συστήματος όσο και η σωστή και στοχευμένη εμπειρία πλοήγησης ανάλογα με τον ρόλο κάθε χρήστη.

Εικόνα 4.18 Administrator dashboard

Εικόνα 4.19 Store dashboard



Εικόνα 4.20 User navigation bar

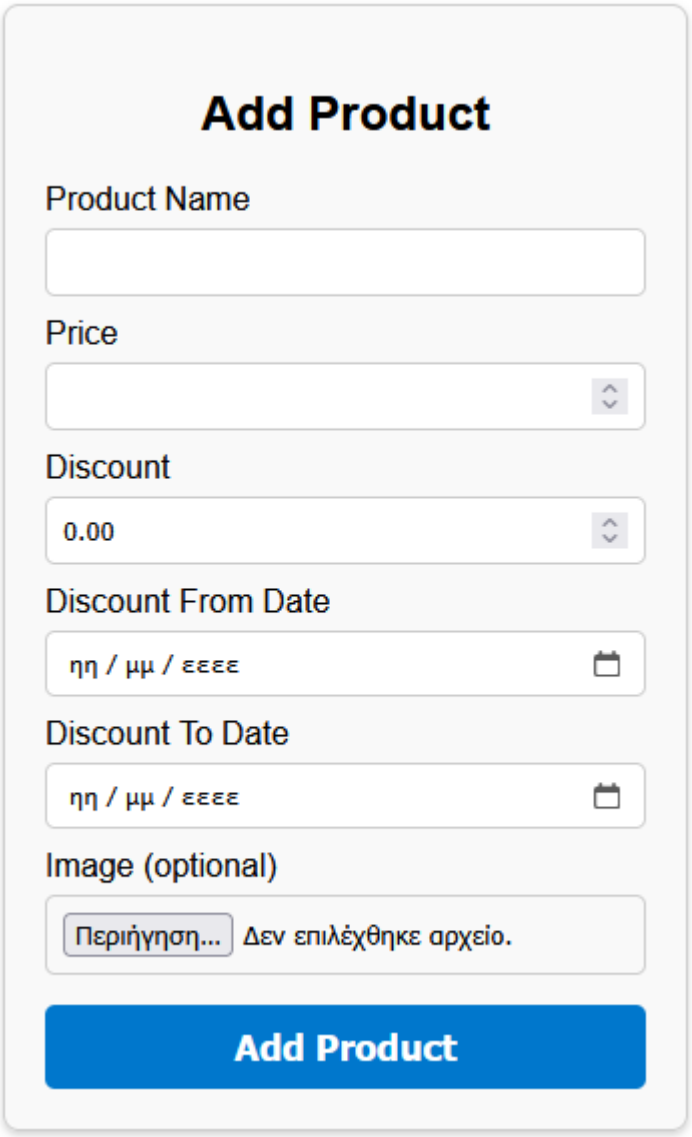
4.12 Δυναμικά Components Δεδομένων με Σύστημα Αιτήματος/Απάντησης

Δημιουργία δυναμικών component δεδομένων που επικοινωνουν με τον διακομιστή μέσω του πρωτοκόλλου HTTP, χρησιμοποιώντας διαφορετικούς τύπους αιτημάτων ανάλογα με τη λειτουργία του. Το κεντρικό component εκτελεί GET αιτήματα για την ανάκτηση όλων των δεδομένων, όπως τα προϊόντα του αεροδρομίου και τα υπόλοιπα components μπορούν να στέλνουν POST αιτήματα για δημιουργία νέων εγγραφών, PATCH για την ενημέρωση υφιστάμενων δεδομένων και DELETE για τη διαγραφή τους. Αυτό το σύστημα αιτημάτων/απαντήσεων επιτρέπει την αρθρωτή και δυναμική διαχείριση των δεδομένων, διασφαλίζοντας ταυτόχρονα την ακεραιότητα και την επικαιρότητά τους.

4.12.1 Δημιουργία Component για Προσθήκη Δεδομένων

Δημιουργία component (Εικόνα 4.21) που έχει ως στόχο την εισαγωγή νέων δεδομένων στην εφαρμογή αξιοποιεί το HTTP POST request για την αποστολή πληροφοριών στον διακομιστή. Η αποστολή των δεδομένων προστατεύεται μέσω συστήματος session cookie, διασφαλίζοντας ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν τις ενέργειες αυτές. Μέσω αυτού του

μηχανισμού, τα δεδομένα υποβάλλονται με ασφαλή και ελεγχόμενο τρόπο, ενεργοποιώντας αντίστοιχες διαδικασίες επεξεργασίας και αποθήκευσης στον server. Η χρήση subcomponents για τις λειτουργίες δημιουργίας επιτρέπει την αρθρωτή οργάνωση του frontend, εξασφαλίζοντας ευκολία στην επαναχρησιμοποίηση και συντήρηση των στοιχείων, διατηρώντας παράλληλα την επικοινωνία με τον κεντρικό διακομιστή αποδοτική και συνεπή.

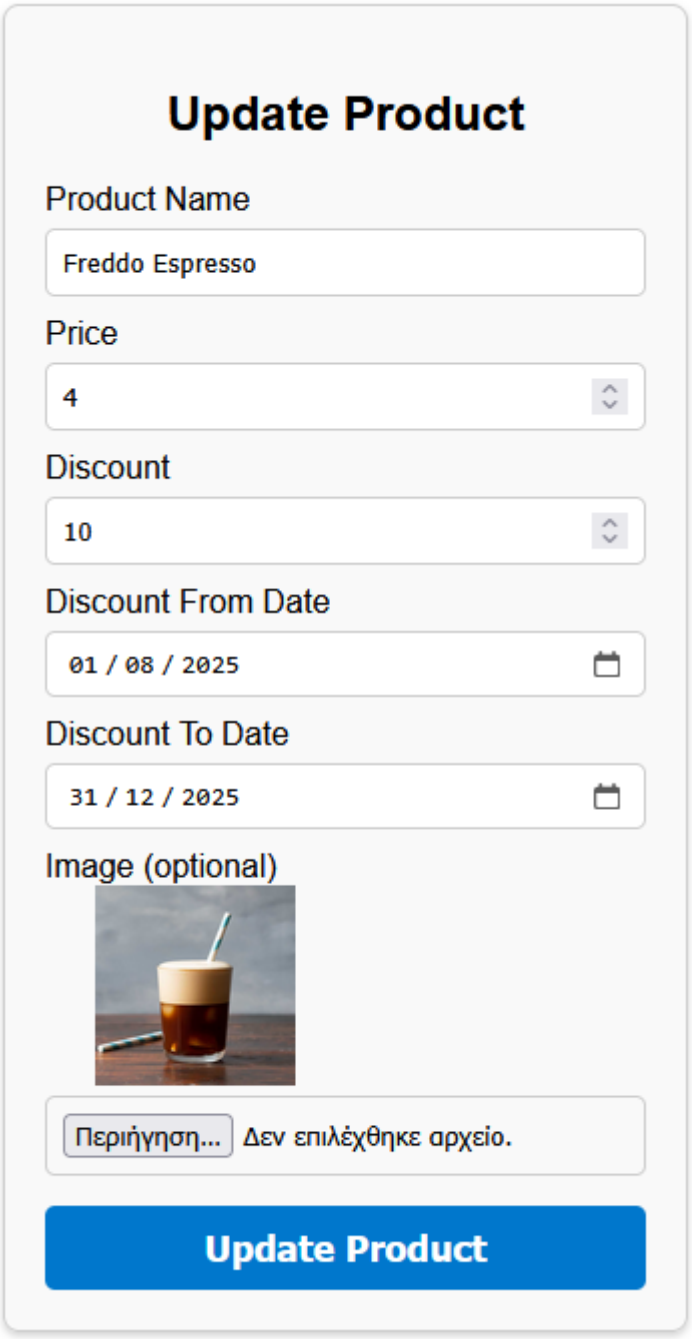


Εικόνα 4.21 Component Προσθηκης προϊόντος από ρόλο store

4.12.2 Δημιουργία Component για Ανανέωση Δεδομένων

Δημιουργία component (Εικόνα 4.22) που έχει ως στόχο την ενημέρωση υπαρχόντων δεδομένων στην εφαρμογή χρησιμοποιεί το HTTP PATCH request για την αποστολή τροποποιήσεων στον διακομιστή. Η αποστολή των ενημερώσεων προστατεύεται μέσω session cookie, εξασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να τροποποιούν τα δεδομένα. Αυτός ο μηχανισμός διατηρεί την ασφάλεια και την ακεραιότητα των πληροφοριών, επιτρέποντας ταυτόχρονα την ταχεία και στοχευμένη ενημέρωση συγκεκριμένων πεδίων χωρίς να επηρεάζονται άλλα δεδομένα. Η χρήση components για τις λειτουργίες ανανέωσης συμβάλλει στην αρθρωτή και οργανωμένη σχεδίαση του

frontend, διευκολύνοντας την επαναχρησιμοποίηση των στοιχείων και την ομαλή επικοινωνία με τον κεντρικό διακομιστή.

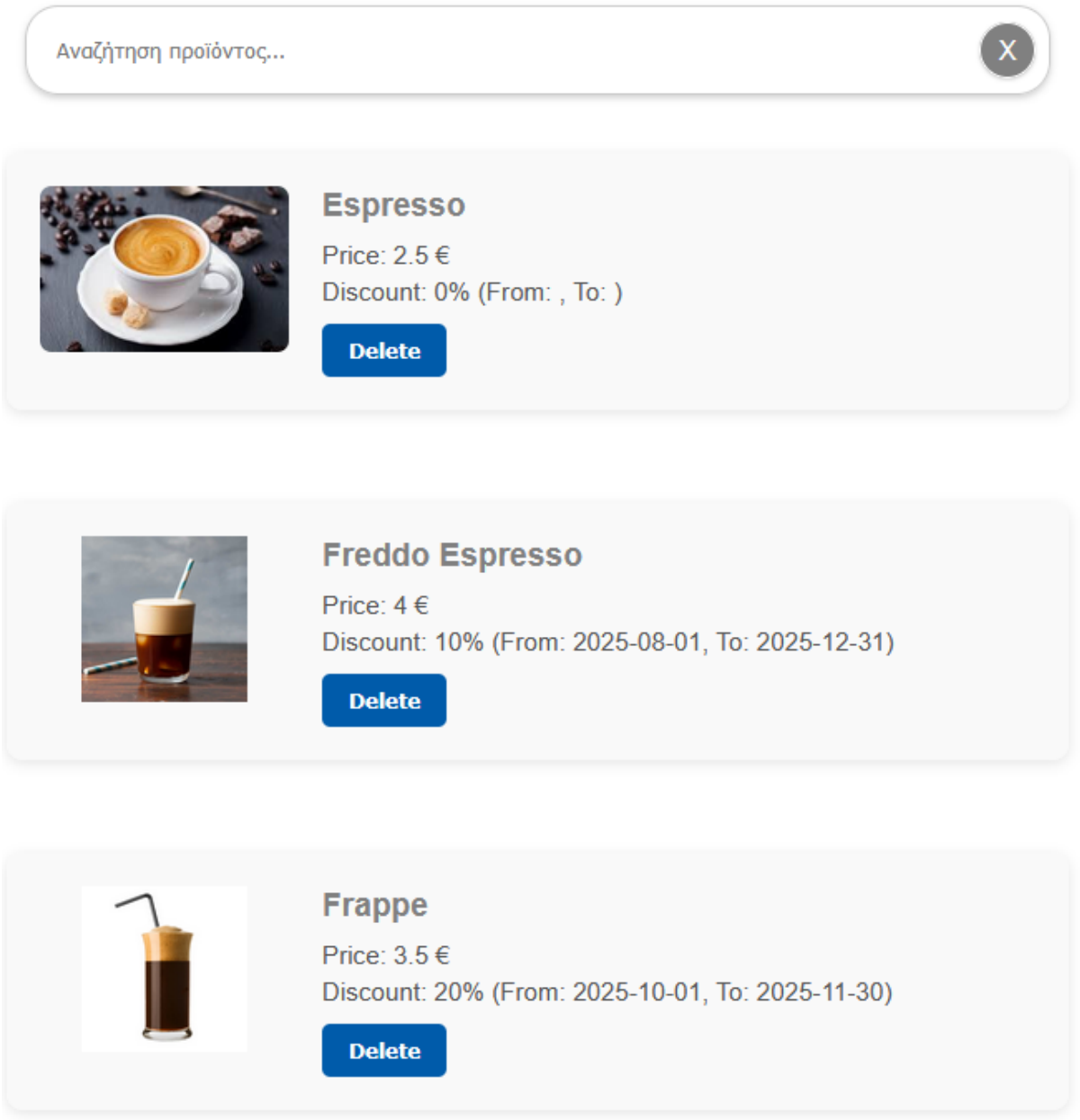


Εικόνα 4.22 Component Ανανέωσης προϊόντος από ρόλο store

4.12.3 Υλοποίηση Component Ιστοσελίδας Δυναμικών Δεδομένων

Δημιουργία component (Εικόνα 4.23) που ια ολοκληρωμένη ιστοσελίδα δεδομένων, όπου τα στοιχεία εμφανίζονται σε μορφή cards και συνοδεύονται από μια search bar για εύκολη αναζήτηση. Η λήψη των δεδομένων πραγματοποιείται μέσω HTTP GET request, επιτρέποντας στο component να συλλέγει και να παρουσιάζει όλες τις διαθέσιμες πληροφορίες από τον διακομιστή. Κάθε card

περιλαμβάνει κουμπί Delete, το οποίο στέλνει HTTP DELETE request στον διακομιστή, επιτρέποντας την άμεση διαγραφή συγκεκριμένων εγγραφών.

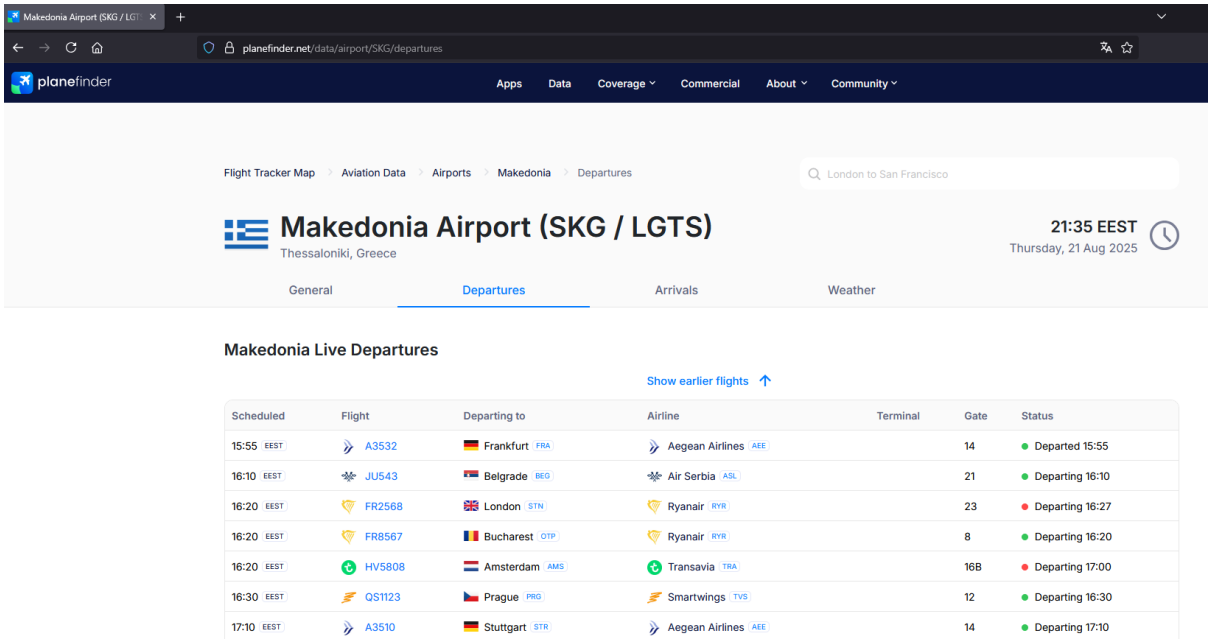


Εικόνα 4.23 Component προβολης προϊόντων χρήστη με ρόλο Store

4.13 Συλλογή Δεδομένων (Data Scrapping) μέσω βιβλιοθηκης Selenium

Η διαδικασία συλλογής δεδομένων της εφαρμογής υλοποιείται με τη χρήση της βιβλιοθήκης Selenium σε Python, η οποία επιτρέπει τον έλεγχο του browser Chrome για την εξαγωγή πληροφοριών από εξωτερικές ιστοσελίδες σε πραγματικό χρόνο. Συγκεκριμένα, γίνεται συλλογή δεδομένων για τις πτήσεις που αναχωρούν (Εικόνα 4.24) και προσγειώνονται στο αεροδρόμιο «Μακεδονία» μέσω της ιστοσελίδας PlaneFinder, η οποία παρέχει ζωντανές πληροφορίες για θέσεις,

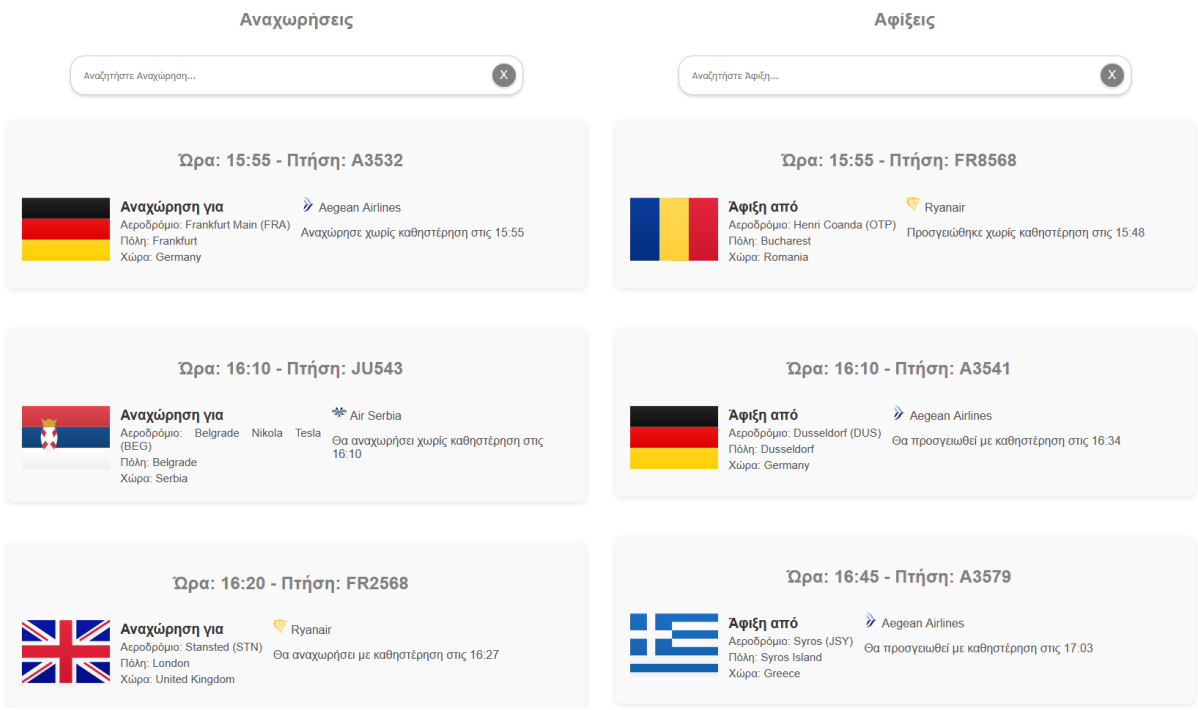
δρομολόγια και αναμενόμενους χρόνους πτήσεων αεροσκαφών σε όλο τον κόσμο. Παράλληλα, συλλέγονται δεδομένα για τις τρέχουσες καιρικές συνθήκες στο αεροδρόμιο, επιτρέποντας την ενσωμάτωσή τους στην εφαρμογή. Η διαδικασία περιλαμβάνει το άνοιγμα της ιστοσελίδας, την πλοήγηση στα κατάλληλα τμήματα που περιέχουν τις πληροφορίες πτήσεων ή καιρού, την εξαγωγή των απαιτούμενων δεδομένων από τα στοιχεία της σελίδας (HTML elements) και την αποθήκευσή τους σε μορφή κατάλληλη στην βάση δεδομένων για χρήση από την εφαρμογή. Η συλλογή πραγματοποιείται αυτόματα κάθε πέντε λεπτά, εξασφαλίζοντας ότι τα δεδομένα ανανεώνονται συνεχώς και παρέχουν ζωντανή ενημέρωση. Η χρήση της Selenium εξασφαλίζει αυτοματοποίηση, αξιοπιστία και ακριβή ενημέρωση, χωρίς να απαιτείται χειροκίνητη εισαγωγή δεδομένων.



Εικόνα 4.24 Αναχωρήσεις planefinder.net

4.14 Component με ζωντανή ανανέωση δεδομένων πτήσεων

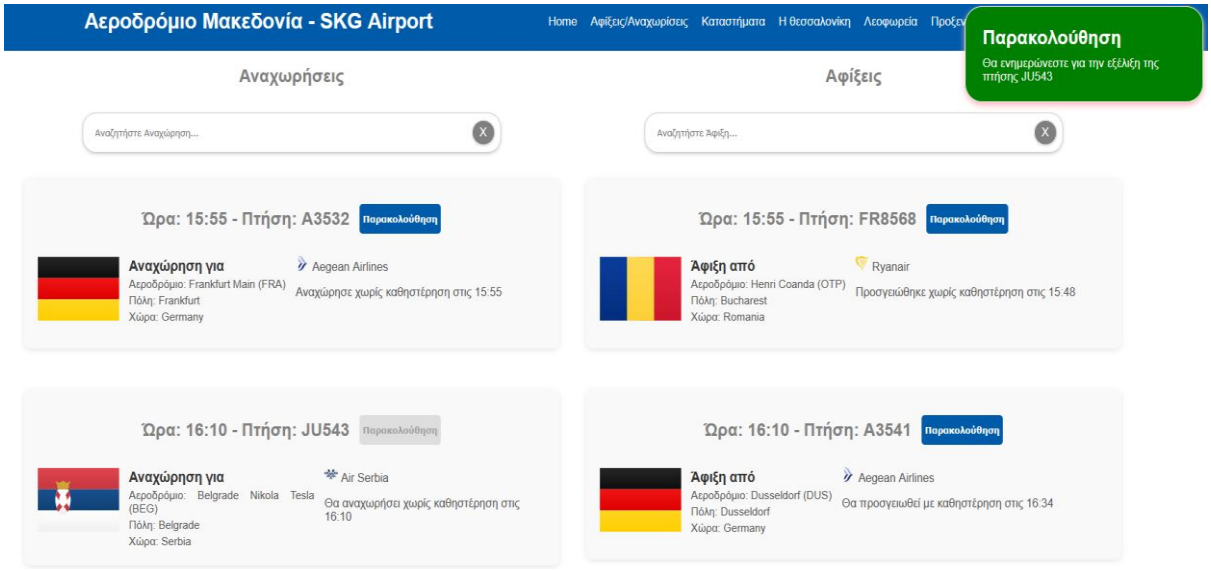
Το component με ζωντανή ανανέωση δεδομένων αποτελεί τον πυρήνα της παρουσίασης πληροφοριών πτήσεων στο αεροδρόμιο Μακεδονία. Αποτελείται από διάφορα subcomponents: οι κάρτες (cards) παρουσιάζουν αναλυτικά κάθε πτήση με στοιχεία όπως ώρα αναχώρησης ή άφιξης, προορισμό ή αφετηρία και τυχόν καθυστερήσεις και τα subcomponents με searchbar δίνουν τη δυνατότητα στους χρήστες να εντοπίζουν εύκολα συγκεκριμένες πτήσεις ανάμεσα σε όλες τις αναχωρήσεις ή αφίξεις. Το κύριο component (Εικόνα 4.25) συγκεντρώνει όλα τα δεδομένα και εκτελεί GET requests για την ανάκτηση των πιο πρόσφατων πληροφοριών από τον server. Η ανανέωση πραγματοποιείται αυτόματα σε κάθε πέντε λεπτά, διασφαλίζοντας ότι οι χρήστες λαμβάνουν ζωντανή ενημέρωση για τις πτήσεις, χωρίς να χρειάζεται να ανανεώνουν χειροκίνητα τη σελίδα. Η προσέγγιση αυτή βελτιώνει σημαντικά την εμπειρία του χρήστη, παρέχοντας ένα διαδραστικό και ενημερωμένο περιβάλλον.



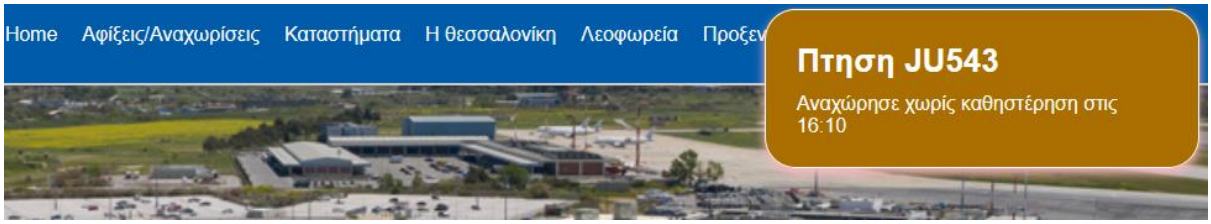
Εικόνα 4.25 Component με ζωντανή ανανέωση δεδομένων πτήσεων

4.15 Σύστημα Παρακολούθησης Δεδομένων με Ζωντανές Ειδοποιήσεις

Το σύστημα παρακολούθησης δεδομένων ενσωματώνεται στο Notification System της εφαρμογής, προσφέροντας στους χρήστες τη δυνατότητα να λαμβάνουν ζωντανές ειδοποιήσεις για τις πτήσεις που τους ενδιαφέρουν. Προστέθηκε ήχος ειδοποίησης ώστε να γίνεται άμεση ενημέρωση σε περίπτωση αλλαγής κατάστασης πτήσης. Οι συνδεδεμένοι χρήστες με ρόλο user μπορούν να επιλέξουν συγκεκριμένες πτήσεις για παρακολούθηση (Εικόνα 4.26) και το σύστημα στέλνει αυτόματα GET requests στον server, κάθε πέντε λεπτά, για να ενημερώνεται σχετικά με τυχόν αλλαγές στην κατάσταση των επιλεγμένων πτήσεων (Εικόνα 4.27),. Η παρακολούθηση συνεχίζεται ακόμη και όταν ο χρήστης μεταβεί σε άλλο component της εφαρμογής, πέρα από αυτό των αναχωρήσεων και αφίξεων. Έτσι, οι χρήστες λαμβάνουν ζωντανή ενημέρωση για καθυστερήσεις, ακυρώσεις ή αλλαγές πύλης, εξασφαλίζοντας πλήρη έλεγχο και άμεση πληροφόρηση χωρίς να χρειάζεται συνεχής χειροκίνητος έλεγχος της εφαρμογής.



Εικόνα 4.26 Παράδειγμα παρακολούθησης πτήσης



Εικόνα 4.27 Παράδειγμα ενημέρωσης αλλαγής κατάστασης πτήσης

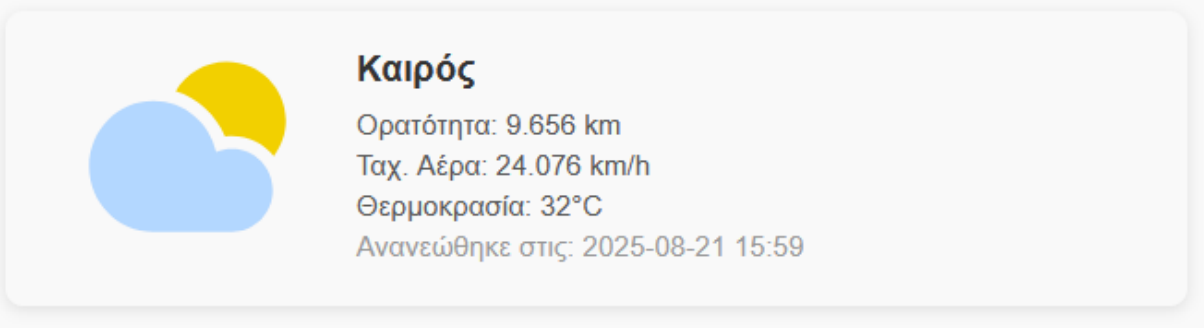
4.16 Δημιουργία Thread για Παράλληλη Εκτέλεση διακομιστή ιστού και συλλογή δεδομένων

Για την απρόσκοπτη λειτουργία της διαδικτυακής εφαρμογής και την παροχή ζωντανά ανανεωμένων δεδομένων, υλοποιήθηκε μηχανισμός παραλληλισμού μέσω της βιβλιοθήκης threading της Python. Συγκεκριμένα, δημιουργήθηκαν ξεχωριστά threads, εκ των οποίων το πρώτο αναλαμβάνει την εκτέλεση του web server (Flask) και την εξυπηρέτηση αιτημάτων των χρηστών, την ώρα που το δεύτερο εκτελεί σε τακτά χρονικά διαστήματα την συλλογή δεδομένων με χρήση της βιβλιοθήκης Selenium, συλλέγοντας δεδομένα πτήσεων και καιρού από την εξωτερική πηγή (planefinder.net). Η παράλληλη εκτέλεση εξασφαλίζει ότι η διαδικασία συλλογής δεδομένων δεν επιβαρύνει την απόδοση του διακομιστή, επιτυγχάνοντας ταυτόχρονα συνεχή ενημέρωση των δεδομένων που εμφανίζονται στον τελικό χρήστη. Με τον τρόπο αυτό, το σύστημα συνδυάζει υψηλή διαθεσιμότητα και αμεσότητα πληροφορίας, χαρακτηριστικά κρίσιμα για μια δια εφαρμογή πραγματικού χρόνου.

4.17 Widget Καιρού με Αυτόματη Ανανέωση

Στο πλαίσιο της εφαρμογής αναπτύχθηκε ειδικό widget καιρού (Εικόνα 4.28), το οποίο συλλέγει δεδομένα μέσω GET request από τον server και τα εμφανίζει στον τελικό χρήστη σε μορφή δυναμικά ανανεούμενου component. Η ανανέωση πραγματοποιείται αυτόματα σε προκαθορισμένα χρονικά

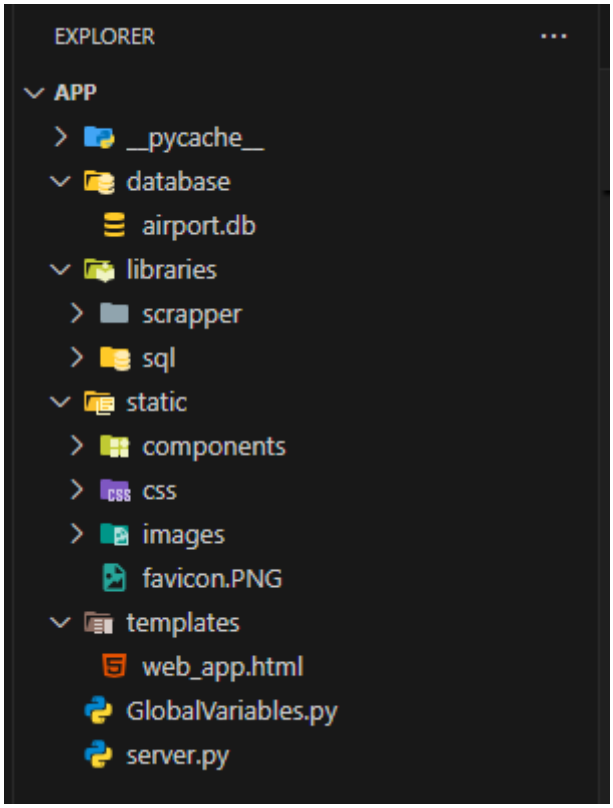
διαστήματα, εξασφαλίζοντας έτσι την ζωντανή ενημέρωση των καιρικών συνθηκών στο αεροδρόμιο «Μακεδονία». Παράλληλα, το widget σχεδιάστηκε με τρόπο που επιτρέπει την ενσωμάτωσή του σε εξωτερικές ιστοσελίδες μέσω μηχανισμού embedding (π.χ. μέσω iframe ή JavaScript injection), επεκτείνοντας την αξιοποίηση του συστήματος πέρα από τα όρια της κύριας εφαρμογής. Με τον τρόπο αυτό, το widget μπορεί να αποτελέσει εργαλείο ενημέρωσης τόσο για τρίτους ιστότοπους (π.χ. τουριστικούς οδηγούς, ταξιδιωτικά πρακτορεία) όσο και για χρήστες που αναζητούν άμεση πληροφόρηση χωρίς να χρειάζεται να εισέλθουν στο κεντρικό σύστημα.



Εικόνα 4.28 Widget Καιρού με Αυτόματη Ανανέωση

4.18 Δομή Φακέλων και Αρχείων της Εφαρμογής

Η αρχιτεκτονική της εφαρμογής έχει σχεδιαστεί με γνώμονα την οργάνωση, επεκτασιμότητα και καθαρότητα του κώδικα, ακολουθώντας καλές πρακτικές ανάπτυξης λογισμικού. Όλα τα αρχεία και φάκελοι βρίσκονται εντός του βασικού φακέλου APP (Εικόνα 4.29), ο οποίος αποτελεί τον πυρήνα του έργου. Η εσωτερική του δομή χωρίζεται σε υποφακέλους με διακριτούς ρόλους και αρμοδιότητες:



Εικόνα 4.29 Δομή Αρχείων της Εφαρμογής

4.18.1 Φάκελος database

Ο φάκελος αυτός περιέχει τη βάση δεδομένων της εφαρμογής, η οποία είναι υλοποιημένη σε SQLite. Το αρχείο airport.db αποθηκεύει όλα τα απαραίτητα δεδομένα που σχετίζονται με την εφαρμογή, όπως πληροφορίες για πτήσεις, προϊόντα, χρήστες, καταστήματα, κ.ά. Η χρήση SQLite επιλέχθηκε λόγω της απλότητας, της φορητότητας και της συμβατότητάς της με εφαρμογές μικρού και μεσαίου μεγέθους, όπως αυτή.

4.18.2 Φακελος libraries

Ο φάκελος αυτός περιέχει εσωτερικές βιβλιοθήκες Python που κατασκευάστηκαν από την δημιουργό της παρούσας, χωρισμένες σε δύο υποφακέλους:

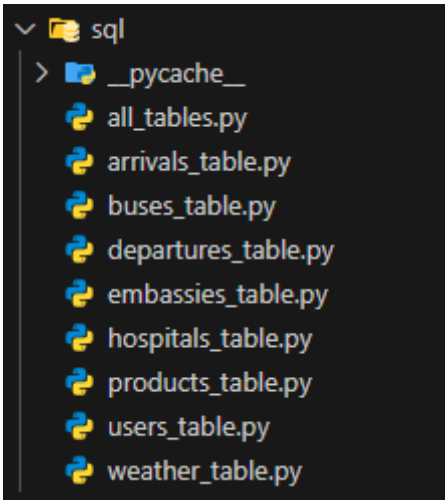
4.18.2.1 Φάκελος scrapper

Περιλαμβάνει τις συναρτήσεις που υλοποιούν τη συλλογή δεδομένων από εξωτερικές πηγές (π.χ. για καιρό, αφίξεις και αναχωρήσεις πτήσεων) με χρήση της βιβλιοθήκης Selenium. Οι διαδικασίες αυτές εκτελούνται αυτόματα σε τακτά χρονικά διαστήματα, αξιοποιώντας μηχανισμούς παράλληλης εκτέλεσης (threading).

4.18.2.2 Φάκελος sql

Περιλαμβάνει τα αρχεία που υλοποιούν τις συναρτήσεις πρόσβασης και διαχείρισης των δεδομένων της βάσης. Οι συναρτήσεις αυτές καλύπτουν λειτουργίες όπως η δημιουργία πινάκων, οι βασικές CRUD εντολές (Create, Read, Update, Delete) και ειδικοί μηχανισμοί υποστήριξης της πρόσβασης

στα δεδομένα (data access helpers). Ο διαχωρισμός αυτός εξασφαλίζει καθαρό διαχωρισμό της επιχειρησιακής λογικής από την αποθήκευση δεδομένων, γεγονός που ενισχύει τόσο τη σαφήνεια όσο και τη συντηρησιμότητα του κώδικα. Για κάθε πίνακα της βάσης δεδομένων δημιουργήθηκε ξεχωριστό αρχείο με τις αντίστοιχες συναρτήσεις. Επιπροσθέτως υπάρχει και ένα αρχείο που συγκεντρώνει συναρτήσεις γενικής χρήσης, οι οποίες είναι κοινές και αξιοποιούνται από όλα τα υπόλοιπα αρχεία του φακέλου.



Εικόνα 4.30 Δομή φακέλου sql

4.18.3 Φάκελος static

Ο φάκελος static περιέχει όλα τα στατικά αρχεία της εφαρμογής, δηλαδή πόρους που εξυπηρετούνται απευθείας από τον διακομιστή χωρίς περαιτέρω επεξεργασία.

4.18.3.1 Φάκελος components

Περιέχει modular αρχεία JavaScript που υλοποιούν ξεχωριστά τμήματα του frontend, όπως cards, widgets και SearchBar. Επιπλέον, περιλαμβάνει αρχεία που υλοποιούν components τα οποία αντιπροσωπεύουν ολόκληρες σελίδες ή σημαντικά τμήματα της διεπαφής χρήστη. Αυτή η δομή ενισχύει την επαναχρησιμοποίηση του κώδικα και υποστηρίζει την αρθρωτή σχεδίαση (modular design), καθιστώντας την ανάπτυξη πιο οργανωμένη και ευέλικτη.

Με την τμηματοποίηση των components, η συντήρηση και η επέκταση του κώδικα γίνονται πιο εύκολες και η συνεργασία σε μεγαλύτερες ομάδες προγραμματιστών απλοποιείται, οπότε κάθε developer μπορεί να δουλεύει σε διαφορετικά components ανεξάρτητα. Επιπλέον, η modular προσέγγιση διευκολύνει την ενσωμάτωση νέων λειτουργιών χωρίς να επηρεάζεται η υπάρχουσα δομή της εφαρμογής.

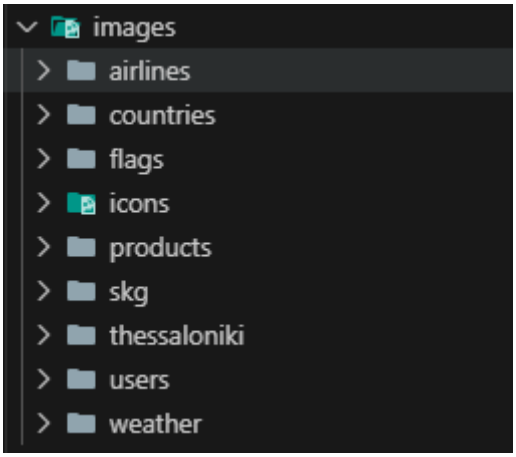
4.18.3.2 Φάκελος css

Περιέχει αρχεία μορφοποίησης (CSS) που καθορίζουν τις σχεδιαστικές οδηγίες της εφαρμογής. Συγκεκριμένα, ορίζουν τη μορφοποίηση, τη διάταξη και το οπτικό στυλ του περιβάλλοντος χρήστη (UI), εξασφαλίζοντας ομοιομορφία και συνεκτικότητα σε όλα τα στοιχεία της εφαρμογής. Επιπλέον, τα CSS αρχεία μπορούν να χρησιμοποιηθούν για εύκολη προσαρμογή του θέματος και της εμφάνισης χωρίς να επηρεάζεται η λειτουργικότητα.

4.18.3.3 Φάκελος images

Ο φάκελος images περιέχει εικόνες που χρησιμοποιούνται στο frontend, όπως avatars χρηστών, φωτογραφίες αεροδρομίων, σημαίες χωρών κ.ά. Οι εικόνες αυτές υποστηρίζουν την οπτική αναπαράσταση της εφαρμογής και βελτιώνουν την εμπειρία του χρήστη. Οι φωτογραφίες είναι οργανωμένες σε υποφακέλους.

Η οργάνωση σε υποφακέλους διευκολύνει την επαναχρησιμοποίηση των εικόνων, τη συντήρηση του κώδικα και την προσθήκη νέων εικόνων στο μέλλον. Παράλληλα, η σαφής κατηγοριοποίηση βελτιώνει την αναγνωσιμότητα και την ευκολία πλοήγησης στον φάκελο, ιδιαίτερα σε μεγάλες εφαρμογές με πολλές οπτικές πηγές.



Εικόνα 4.31 Δομή φακέλου images

4.18.4 Φάκελος templates

Ο φάκελος templates περιέχει το αρχείο Jinja web_app.html, το οποίο αξιοποιείται από το Flask για τη δυναμική δημιουργία ιστοσελίδων. Μέσω αυτού υποστηρίζεται η απόδοση διαφορετικών navigation bars ανάλογα με τον ρόλο του χρήστη (administrator, store ή απλός user) και η ενσωμάτωση προστατευμένων λειτουργιών που απαιτούν πιστοποίηση (session cookies).

Η χρήση των templates διευκολύνει την οργάνωση του κώδικα, διότι οι επαναλαμβανόμενες δομές σελίδων (π.χ. headers, footers, navigation bars) συντηρούνται κεντρικά, μειώνοντας την πολυπλοκότητα και αυξάνοντας την αποδοτικότητα. Επιπλέον, η modular προσέγγιση επιτρέπει την εύκολη επέκταση ή τροποποίηση της διεπαφής χωρίς να επηρεάζεται η υπόλοιπη εφαρμογή, βελτιώνοντας την αναγνωσιμότητα και τη συντήρηση του κώδικα.

4.18.5 Αρχείο GlobalVariables.py

Το αρχείο αυτό περιέχει σταθερές και κοινές μεταβλητές που χρησιμοποιούνται σε διαφορετικά μέρη της εφαρμογής, όπως διαδρομές αρχείων, χρονικά διαστήματα ενημέρωσης, ρυθμίσεις για τον scrapper ή τη βάση δεδομένων. Η συγκέντρωση όλων των παραμέτρων σε ένα κεντρικό σημείο διευκολύνει τη συντήρηση και την επεκτασιμότητα του κώδικα, επιτρέποντας εύκολη τροποποίηση των ρυθμίσεων χωρίς να επηρεάζονται άμεσα τα υπόλοιπα μέρη της εφαρμογής. Επιπλέον, ενισχύει την αναγνωσιμότητα και μειώνει την πιθανότητα λαθών κατά την αλλαγή τιμών κοινών μεταβλητών.

4.18.6 Αρχείο server.py

Πρόκειται για το κύριο αρχείο εκτέλεσης της εφαρμογής, το οποίο υλοποιεί τον web server με χρήση του Flask. Στο αρχείο αυτό πραγματοποιούνται οι εξής ενέργειες:

- Αρχικοποίηση του server
- Ορισμός των routes (endpoints)
- Σύνδεση με τη βάση δεδομένων
- Φόρτωση του Jinja HTML template
- Διαχείριση των αιτημάτων (requests) από το frontend
- Εκκίνηση thread που εκτελεί τον scrapper για τη συλλογή δεδομένων

Το server.py αποτελεί το κεντρικό σημείο ελέγχου της εφαρμογής, συντονίζοντας τη ροή των δεδομένων μεταξύ frontend, backend και βάσης δεδομένων. Η χρήση threads επιτρέπει την παράλληλη εκτέλεση εργασιών χωρίς να επηρεάζεται η απόκριση του server.

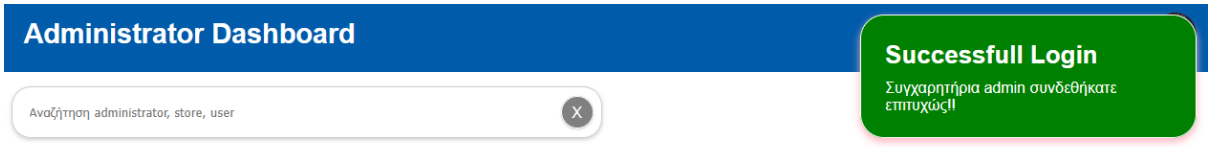
4.19 Εγχειρίδιο Λειτουργίας Εφαρμογής (Manual)

Στην ενότητα αυτή παρέχεται αναλυτική επεξήγηση του τρόπου λειτουργίας της εφαρμογής. Το εγχειρίδιο περιγράφει τα κύρια χαρακτηριστικά και τις δυνατότητες της εφαρμογής, τον τρόπο εκκίνησης και χρήσης της και τις διαδικασίες που ακολουθούνται για την αλληλεπίδραση με τον χρήστη και τη διαχείριση δεδομένων. Επιπλέον, περιλαμβάνει οδηγίες για την αντιμετώπιση κοινών προβλημάτων και τη βέλτιστη εκμετάλλευση των λειτουργιών της εφαρμογής, διευκολύνοντας τόσο τους τελικούς χρήστες όσο και τους προγραμματιστές που εμπλέκονται στη συντήρηση ή την επέκταση του συστήματος.

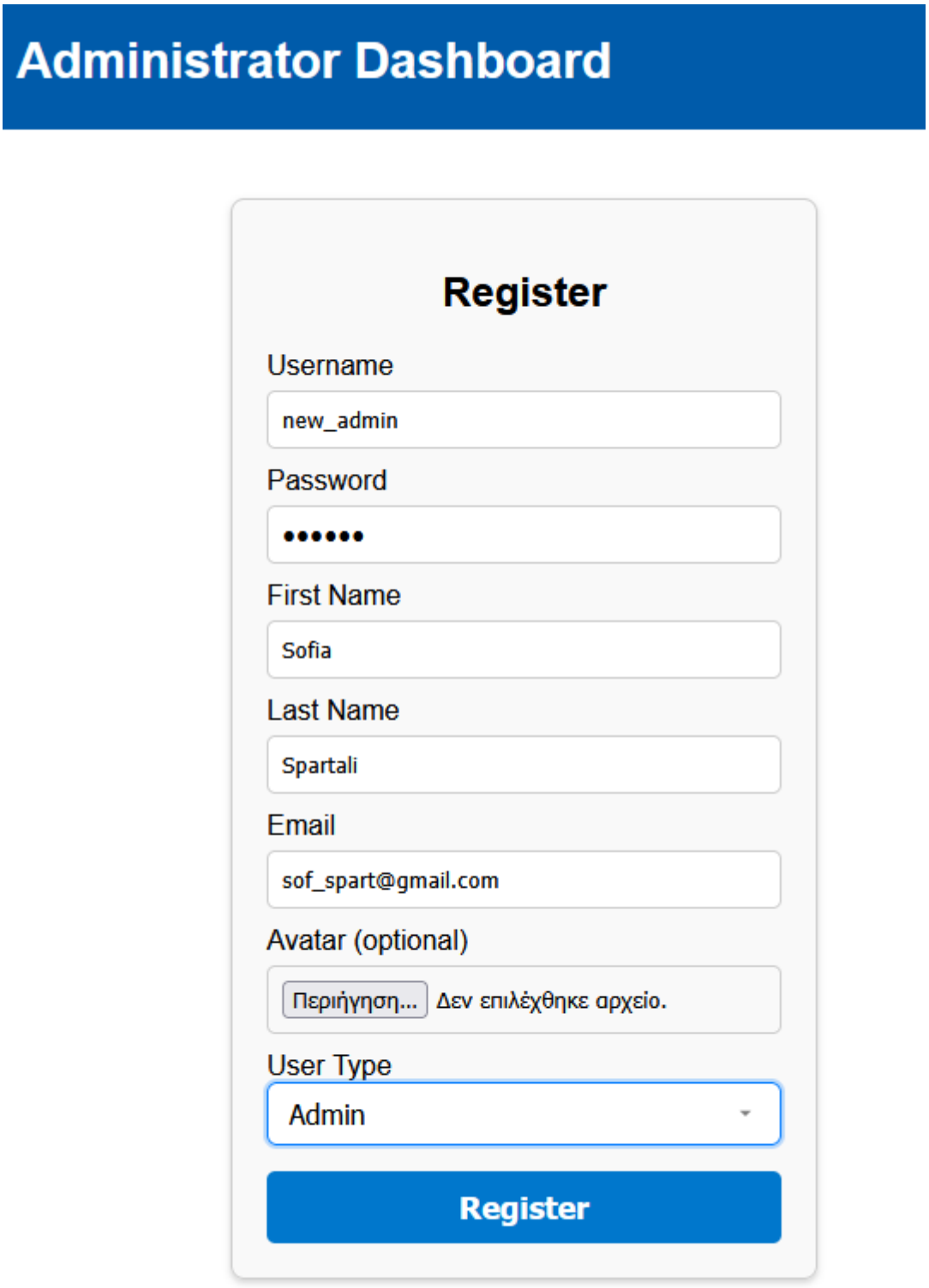
4.19.1 Χρήστης με ρόλο Administrator

Η δημιουργός της παρούσας εφαρμογής έχει τον ρόλο του administrator. Για την εκκίνηση της εφαρμογής, πρέπει μέσα από τερματικό (terminal), στον φάκελο app, να εκτελεστεί η εντολή: python server.py . Απαραίτητη προϋπόθεση είναι να έχουν προηγουμένως εγκατασταθεί όλες οι απαιτούμενες βιβλιοθήκες μέσω του διαχειριστή πακέτων pip. Κατά την εκτέλεση της εντολής αυτής, εκκινεί ο web server. Παράλληλα ξεκινά και ο scrapper για τη συλλογή εξωτερικών δεδομένων, χρησιμοποιώντας Chrome browser ανά πέντε λεπτά.

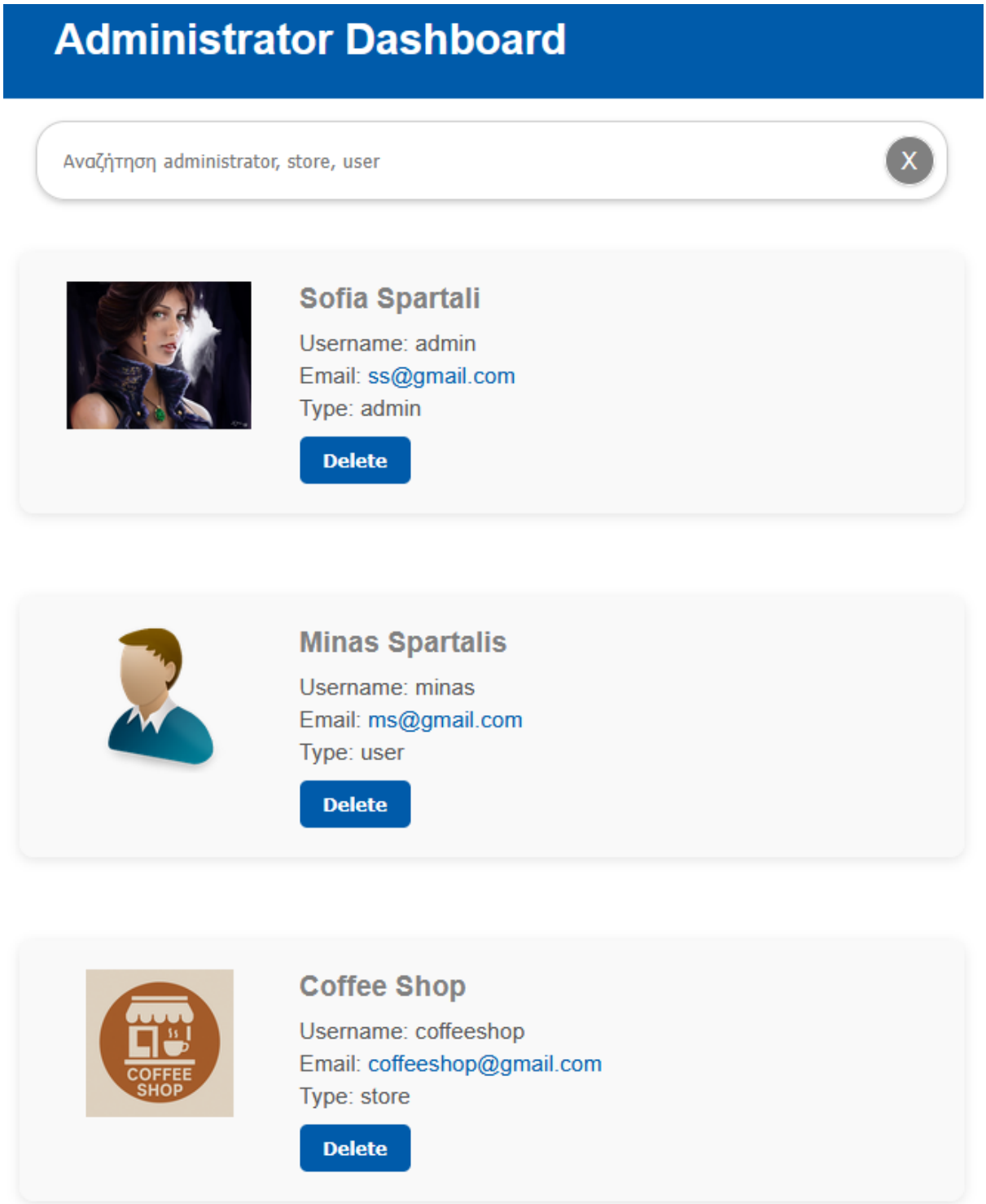
Στη συνέχεια, ο administrator επισκέπτεται την εφαρμογή στο endpoint “/”, συνδέεται με τα διαπιστευτήριά του (Εικόνα 4.32) και μπορεί να αρχίσει να προσθέτει χρήστες (Register) οποιουδήποτε ρόλου (Εικόνα 4.33). Επίσης, έχει πρόσβαση στη λίστα όλων των χρηστών (Εικόνα 4.34), ανεξαρτήτως ρόλου, ώστε να τους διαγράφει ή να τους αναβαθμίζει πατώντας πάνω στην κάρτα τους (Εικόνα 4.35).



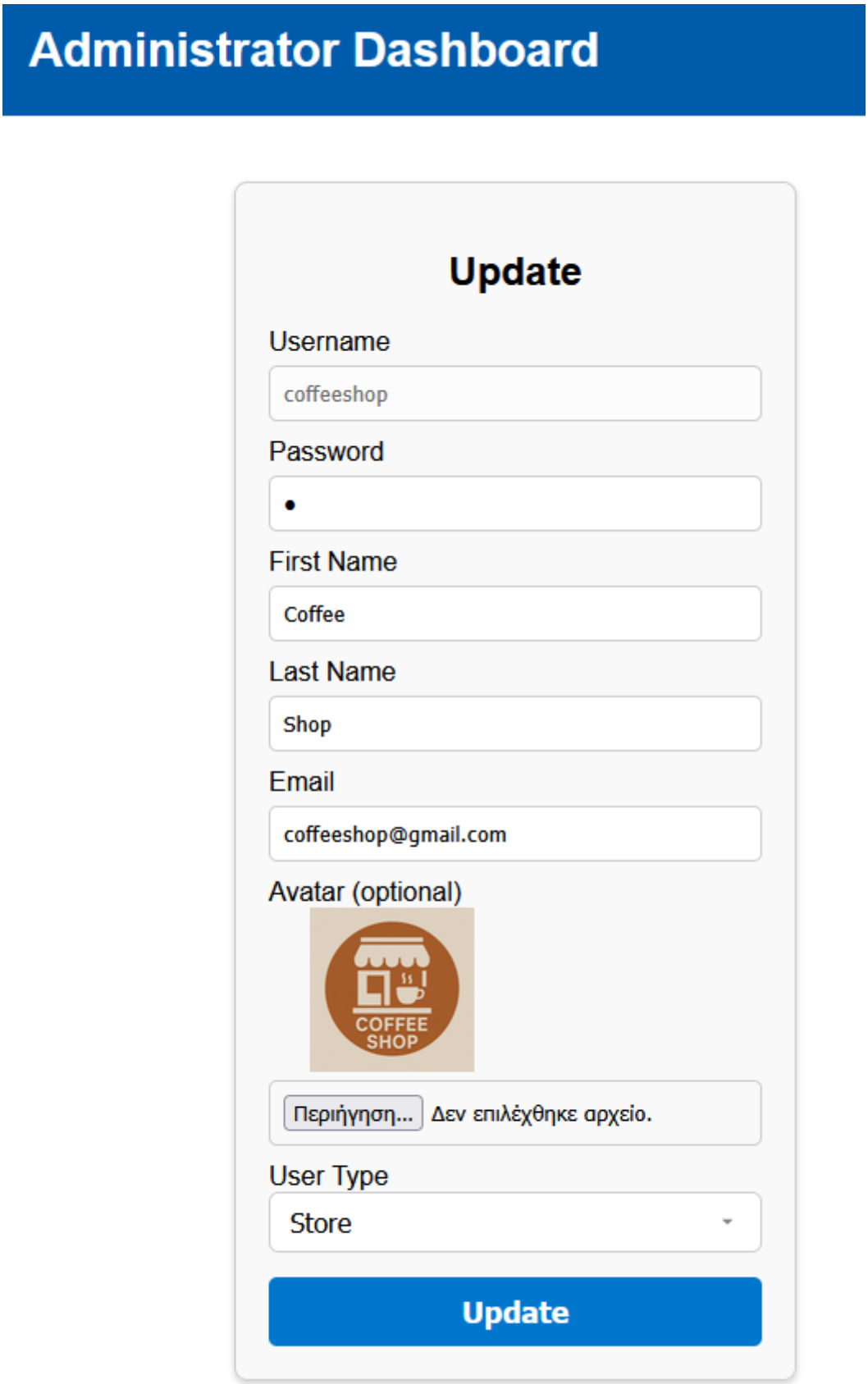
Εικόνα 4.32 Παράδειγμα επιτυχημένης σύνδεσης Administrator



Εικόνα 4.33 Παράδειγμα εισαγωγής νέου administrator



Εικόνα 4.34 Όλοι οι user στο administrator dashboard



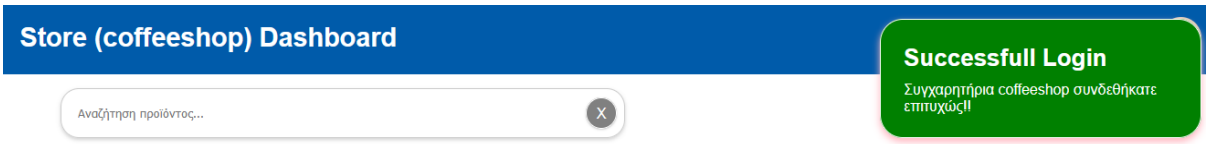
Εικόνα 4.35 Παράδειγμα αναβάθμισης χρήστη με ρόλο store

4.19.2 Χρήστης με ρόλο Store

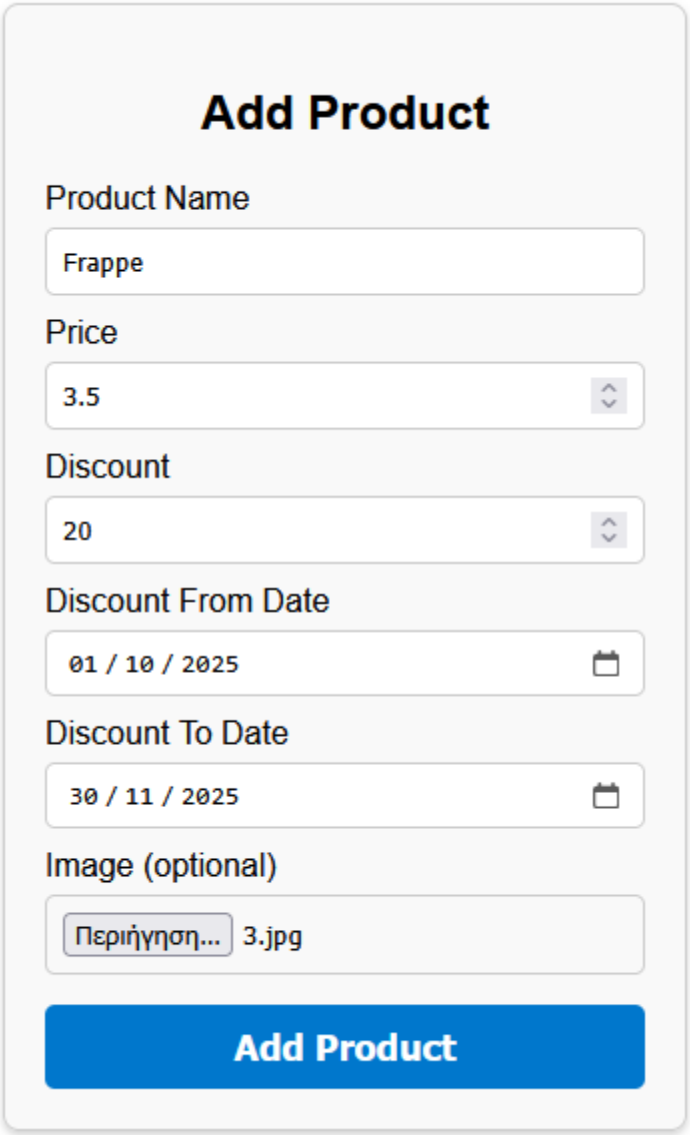
Ο χρήστης με ρόλο store πρέπει αρχικά να απευθυνθεί σε κάποιον administrator της εφαρμογής, ώστε να αποκτήσει τα απαραίτητα διαπιστευτήρια (username και password) για την είσοδό του.

Στη συνέχεια, ο χρήστης με ρόλο store επισκέπτεται την εφαρμογή στο endpoint "/", συνδέεται με τα διαπιστευτήριά του (Εικόνα 4.36) και μπορεί να αρχίσει να προσθέτει προϊόντα του καταστήματός του μέσω της λειτουργίας Add Product. Κατά την εισαγωγή προϊόντων, παρέχεται ενσωματωμένο σύστημα εκπτώσεων (Εικόνα 4.37).

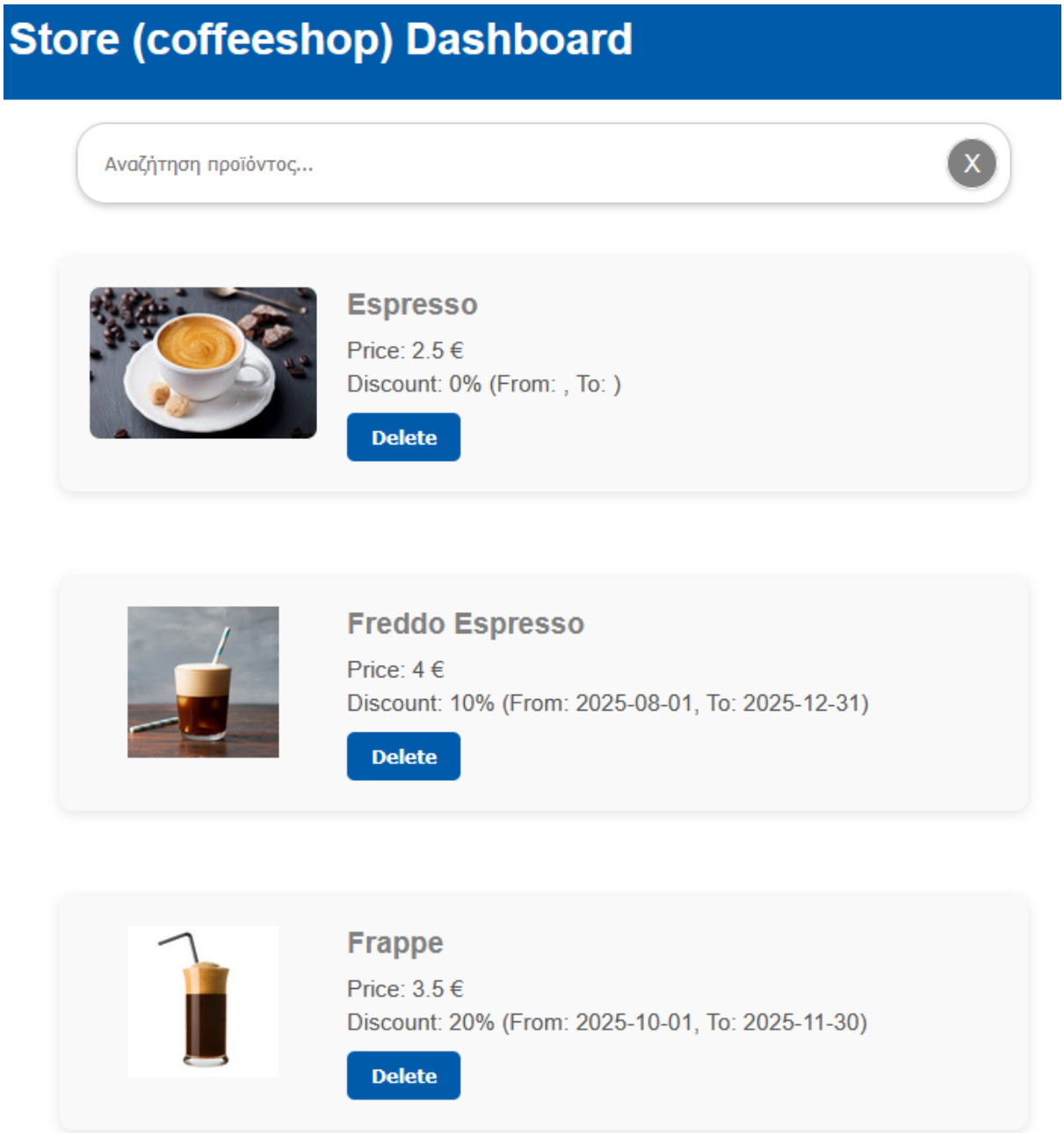
Επιπλέον, έχει πρόσβαση στη λίστα όλων των προϊόντων του καταστήματός του (Εικόνα 4.38) και μπορεί να τα διαγράφει ή να τα ενημερώνει εύκολα, πατώντας επάνω στην κάρτα κάθε προϊόντος (Εικόνα 4.39).



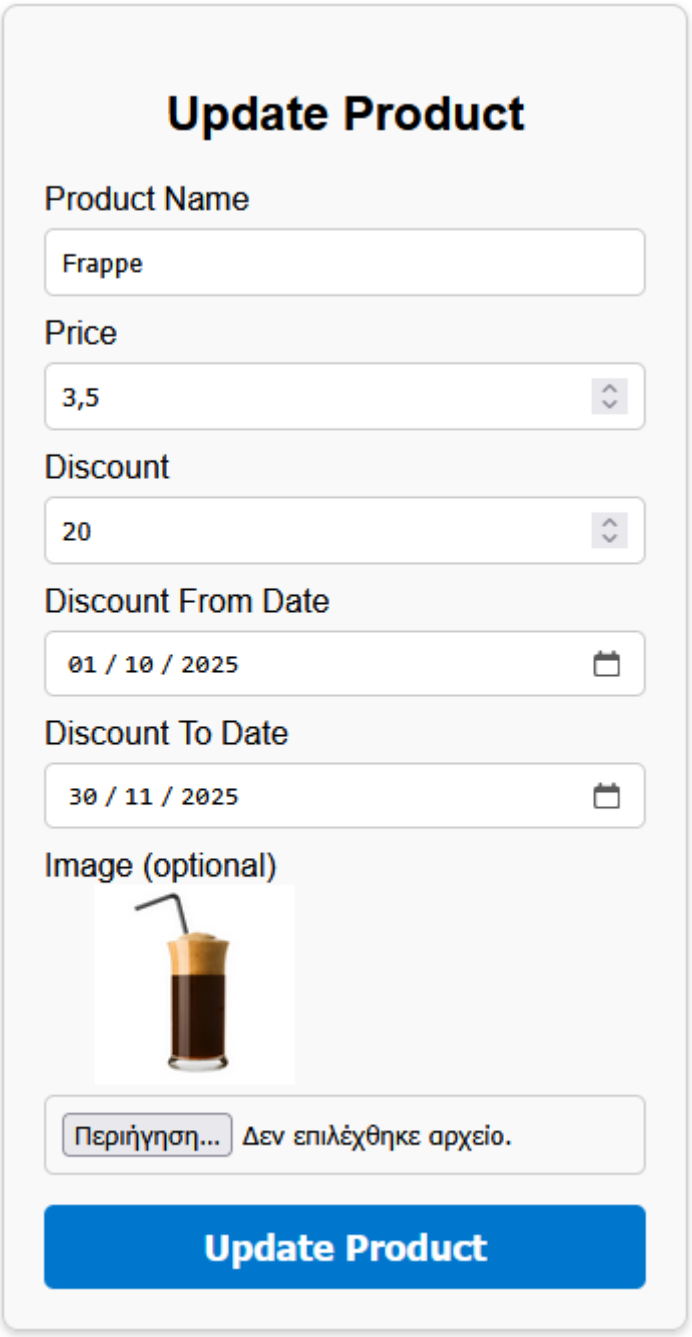
Εικόνα 4.36 Παράδειγμα επιτυχημένης σύνδεσης store



Εικόνα 4.37 Παράδειγμα εισαγωγής νέου προϊόντος



Εικόνα 4.38 Όλα τα προϊόντα καταστήματος στο store dashboard



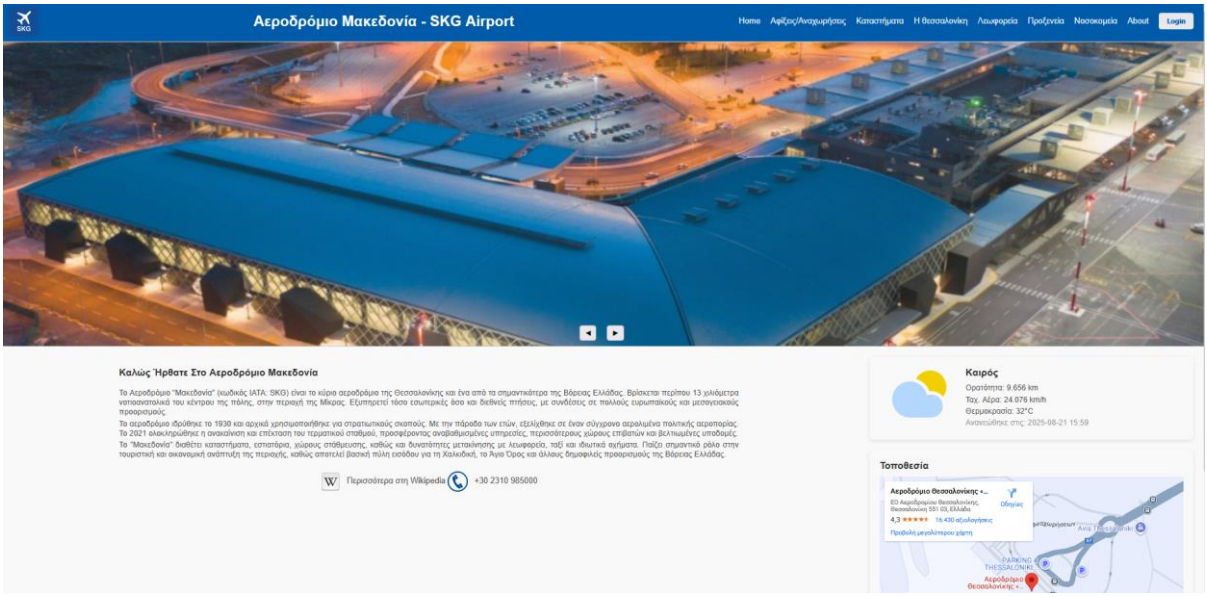
Εικόνα 4.39 Παράδειγμα αναβάθμισης προϊόντος

4.19.3 Μη εγγεγραμμένος χρήστης (Guest)

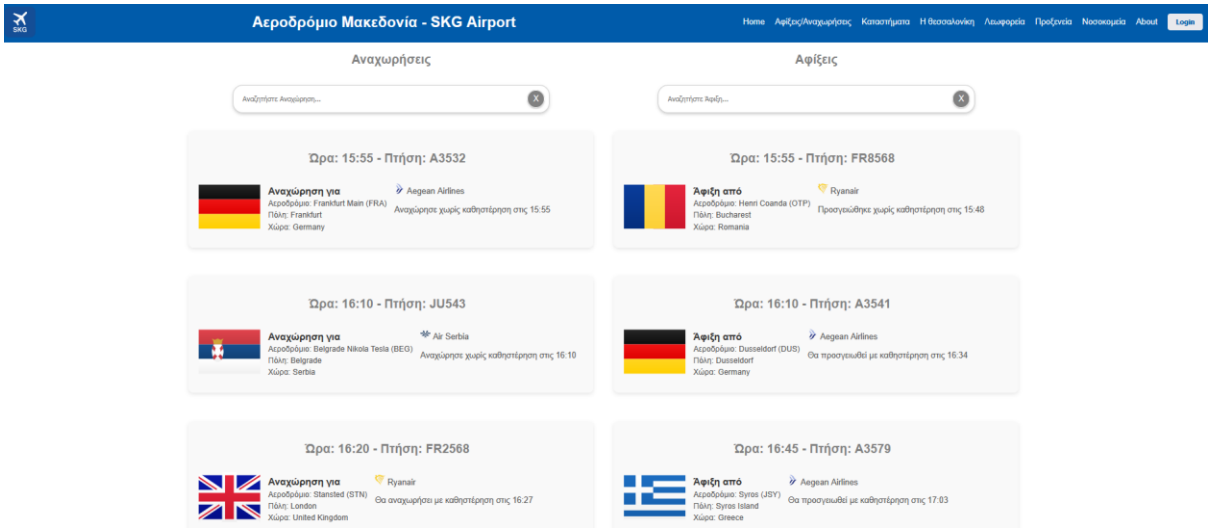
Οποιοσδήποτε μπορεί να επισκεφθεί την εφαρμογή της παρούσας εργασίας χωρίς να χρειάζεται να συνδεθεί στο σύστημα, έχοντας τη δυνατότητα να πλοηγηθεί στα διαθέσιμα components με πληροφορίες σχετικά με:

- Το αεροδρόμιο (Home) (Εικόνα 4.40)
- Τις πτήσεις του αεροδρομίου (Αφίξεις / Αναχωρήσεις) (Εικόνα 4.41)
- Τα καταστήματα του αεροδρομίου (Καταστήματα) (Εικόνα 4.42)
- Την πόλη της Θεσσαλονίκης (Η Θεσσαλονίκη) (Εικόνα 4.43)
- Τα λεωφορεία που εξυπηρετούν το αεροδρόμιο για συγκεκριμένη ημέρα (Λεωφορεία) (Εικόνα 4.44)
- Τα προξενεία χωρών με έδρα στη Θεσσαλονίκη (Προξενεία) (Εικόνα 4.45)
- Τα νοσοκομεία της πόλης (Νοσοκομεία) (Εικόνα 4.46)
- Πληροφορίες για την παρούσα εργασία (About) (Εικόνα 4.47)

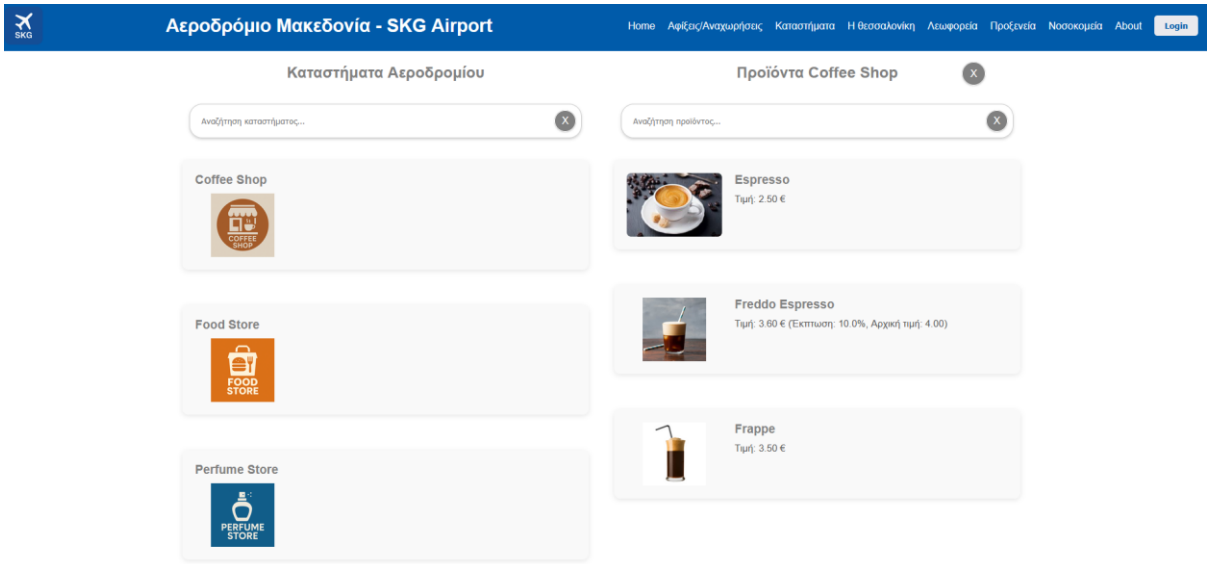
Με αυτόν τον τρόπο, ακόμα και οι μη εγγεγραμμένοι χρήστες μπορούν να επωφεληθούν από τις βασικές λειτουργίες της εφαρμογής, αποκτώντας εύκολη και άμεση πρόσβαση σε χρήσιμες πληροφορίες.



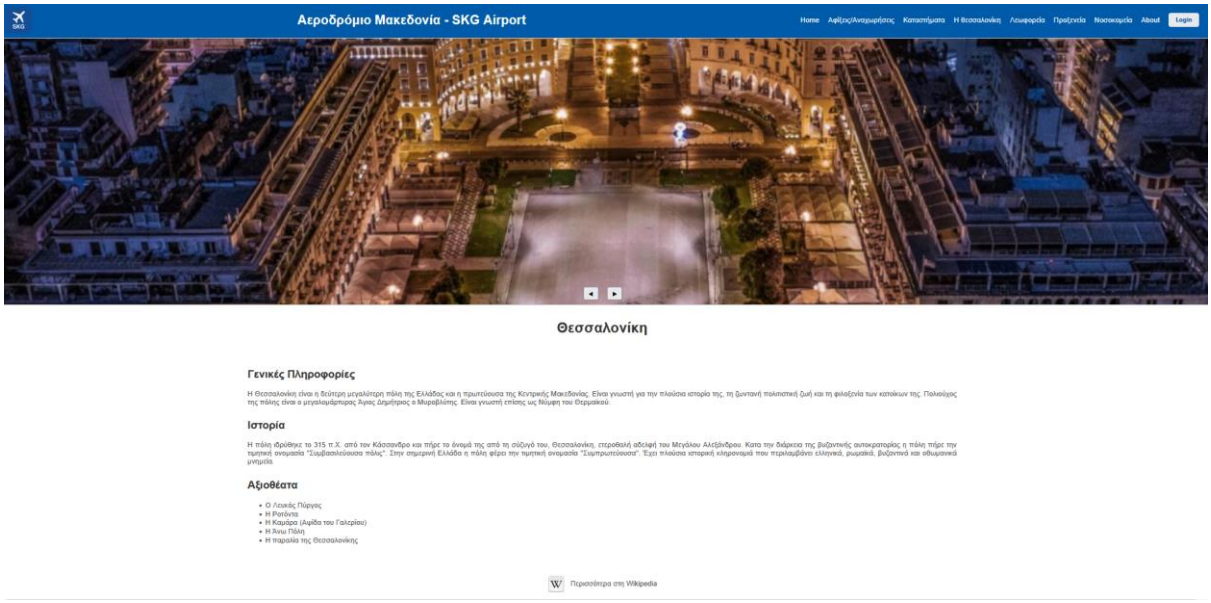
Εικόνα 4.40 Το αεροδρόμιο (Home)



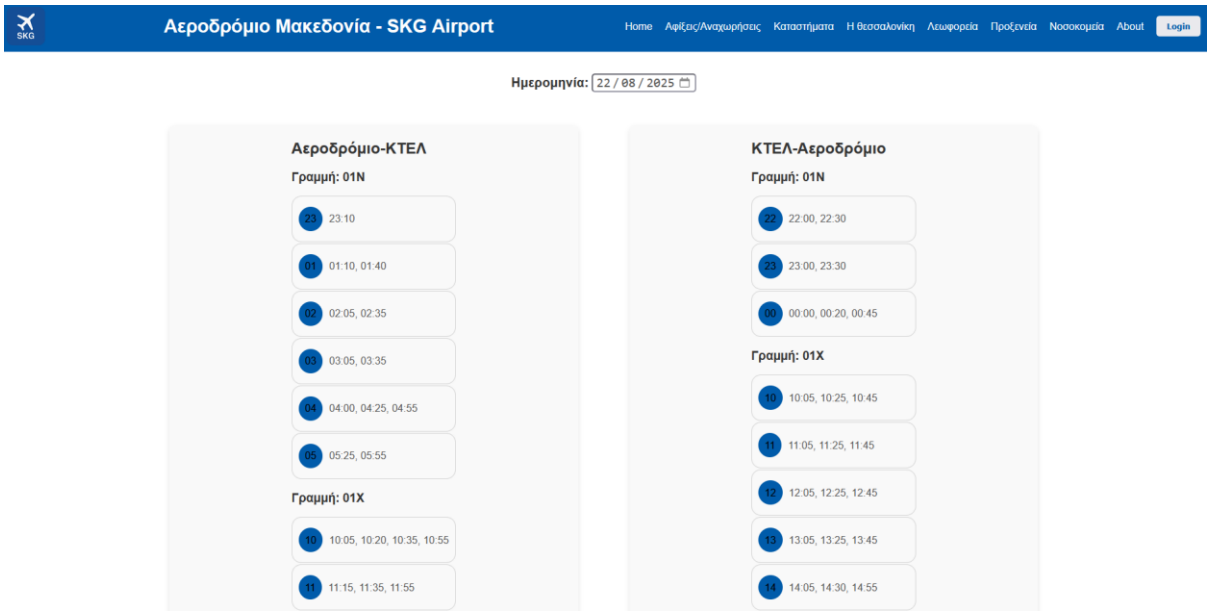
Εικόνα 4.41 Πτήσεις αεροδρομίου (Αφίξεις / Αναχωρήσεις)



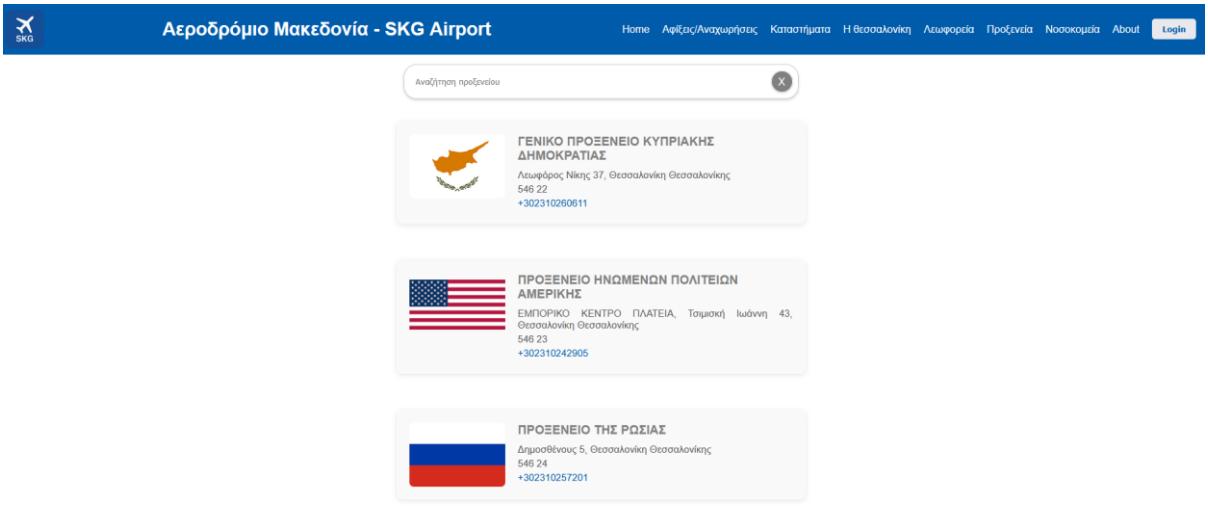
Εικόνα 4.42 Καταστήματα αεροδρομίου (Καταστήματα)



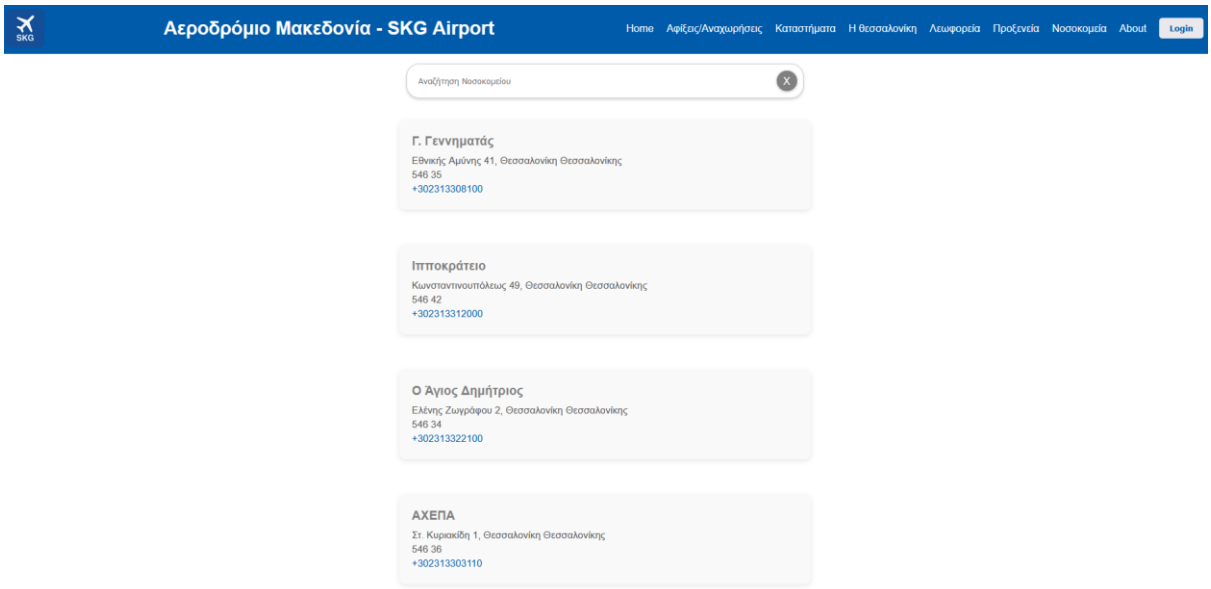
Εικόνα 4.43 Πόλη της Θεσσαλονίκης (Η Θεσσαλονίκη)



Εικόνα 4.44 Λεωφορεία που εξυπηρετούν το αεροδρόμιο (Λεωφορεία)



Εικόνα 4.45 Προξενεία χωρών με έδρα στη Θεσσαλονίκη (Προξενεία)



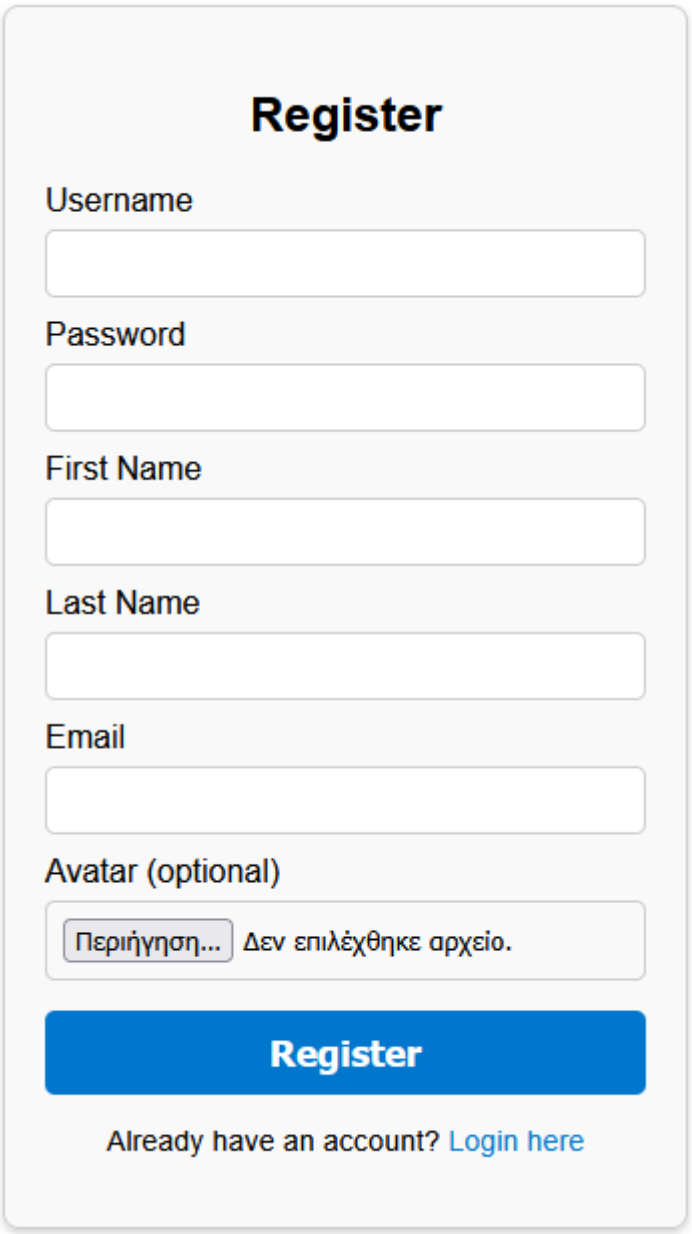
Εικόνα 4.46 Νοσοκομεία της Θεσσαλονίκης (Νοσοκομεία)

Εικόνα 4.47 Πληροφορίες για την παρούσα εργασία (About)

4.19.4 Χρήστης με ρόλο User

Κάθε επισκέπτης της εφαρμογής μπορεί να εγγραφεί μέσω ειδικής φόρμας (Εικόνα 4.48), αποκτώντας τον ρόλο user. Ο ρόλος αυτός διαθέτει το ίδιο σύστημα πλοήγησης με τον απλό επισκέπτη, με μία επιπλέον δυνατότητα: στο component Πτήσεων (Αφίξεις / Αναχωρήσεις), σε κάθε κάρτα προβολής πτήσης εμφανίζεται το κουμπί Προβολή.

Αν ο χρήστης επιλέξει το κουμπί αυτό (Εικόνα 4.49), η συγκεκριμένη πτήση προστίθεται στο σύστημα παρακολούθησης, ώστε ο χρήστης να λαμβάνει ειδοποιήσεις για κάθε αλλαγή στην κατάστασή της. Στο σύστημα παρακολούθησης ο χρήστης μπορεί να προσθέσει απεριόριστο αριθμό πτήσεων, προσαρμόζοντας έτσι την εμπειρία του στις προσωπικές του ανάγκες.



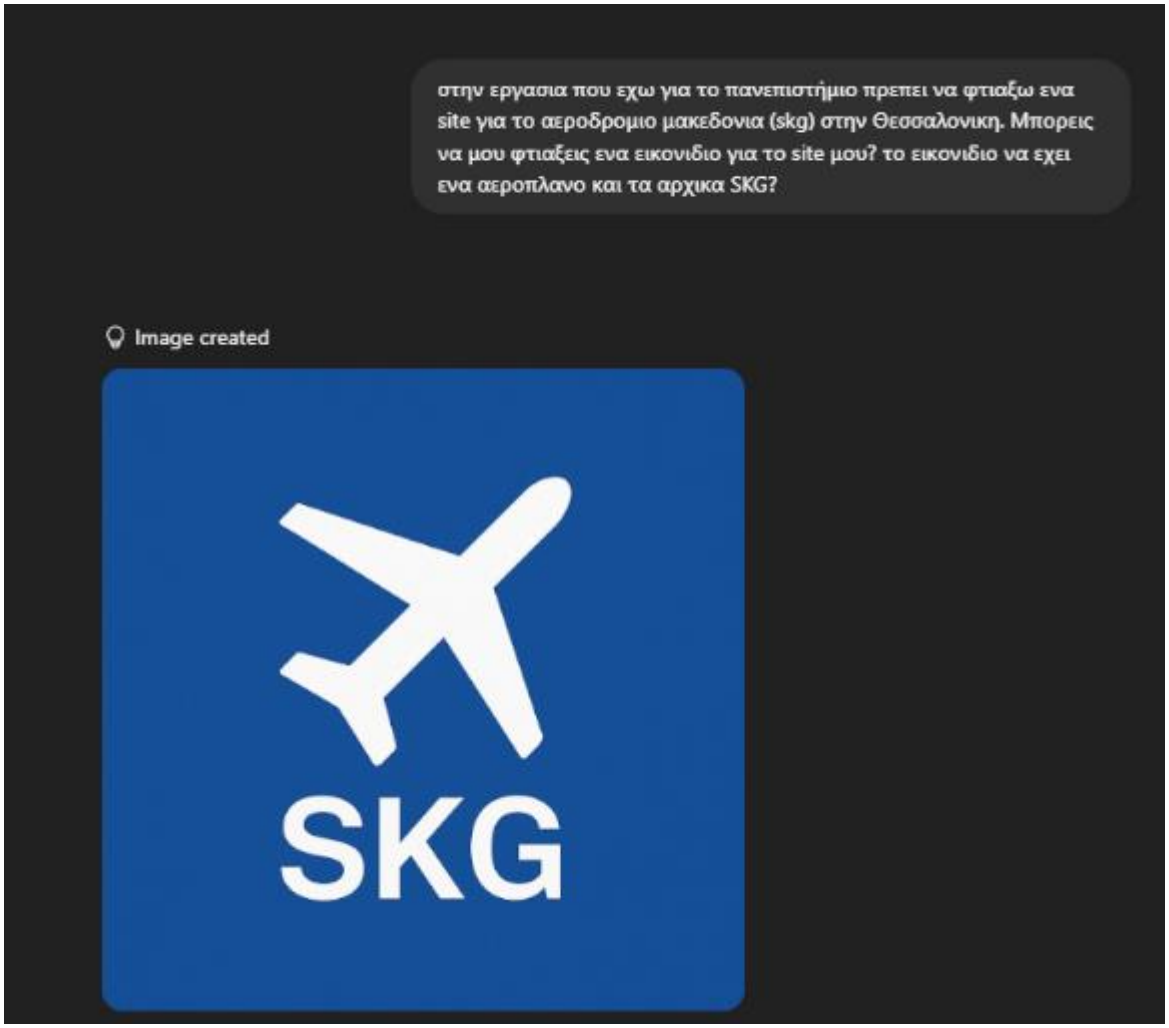
Εικόνα 4.48 Φόρμα εγγραφής χρήστη με ρόλο user

Εικόνα 4.49 Κάρτα προβολής πτήσης με κουμπί παρακολούθησης

4.20 Χρήση τεχνητής νοημοσύνης

Στο πλαίσιο της παρούσας εργασίας αξιοποιήθηκαν εργαλεία Τεχνητής Νοημοσύνης (https://chatgpt.com/) για τη δημιουργία του εικονιδίου της εφαρμογής (Εικόνα 4.50), καθώς και για την παραγωγή εικονιδίων που αντιπροσωπεύουν μη υπαρκτά καταστήματα του αεροδρομίου (Εικόνα 4.51). Η χρήση τέτοιων εργαλείων συνέβαλε στην ταχύτερη και πιο δημιουργική σχεδίαση οπτικών στοιχείων, προσφέροντας παράλληλα συνέπεια στην αισθητική της εφαρμογής. Τα παραγόμενα εικονίδια είναι πρωτότυπα και δεν υπόκεινται σε περιορισμούς πνευματικών δικαιωμάτων, γεγονός που τα καθιστά ελεύθερα προς χρήση στο πλαίσιο της παρούσας εργασίας.

Επίσης η τεχνητή νοημοσύνη χρησιμοποιήθηκε για την βελτίωση της σύνταξης της παρούσας σε επίπεδο ορθογραφίας και συντακτικού.



Εικόνα 4.50 Δημιουργία εικονιδίου εφαρμογής με χρήση τεχνητής νοημοσύνης

Εικόνα 4.51 Δημιουργία εικονιδίου καταστήματος με χρήση τεχνητής νοημοσύνης

Κεφάλαιο 5ο: Συμπεράσματα και Προοπτικές

5.1 Συμπεράσματα επί των εργαλείων

Πριν την εκκίνηση κατασκευής της παρούσας εργασίας, η δημιουργός της επέλεξε εργαλεία (Python, JavaScript, SQLite), το καθένα από τα οποία είναι από τα καλύτερα στο είδος του. Τα εργαλεία αυτά όχι μόνο ανταποκρίθηκαν στις ανάγκες της δημιουργού, αλλά διαθέτουν και δυνατότητες για την επίλυση πιο περίπλοκων προβλημάτων από αυτά που αντιμετωπίστηκαν στην εργασία.

5.2 Συμπεράσματα επί της μεθοδολογίας

Στην παρούσα εργασία πραγματοποιήθηκε παράλληλος προγραμματισμός των επεξεργαστών δύο μηχανημάτων, με σκοπό την επίτευξη αποτελεσματικής επικοινωνίας μεταξύ τους. Το πρώτο μηχάνημα διαθέτει δυνατότητα εκτέλεσης της γλώσσας προγραμματισμού Python, ώστε να επεξεργάζεται εργασίες βάσει εντολών που λαμβάνει από άλλα συστήματα. Το δεύτερο διαθέτει εγκατεστημένο οποιονδήποτε browser, ούτως ώστε να αποστέλλει εντολές στο πρώτο και να εμφανίζει τις απαντήσεις του στον χρήστη.

Για την υλοποίηση των παραπάνω λειτουργιών, στο πρώτο μηχάνημα αναπτύχθηκε διακομιστής (server) που παρέχει endpoints τα οποία εκκινούν τις διάφορες εργασίες, οι οποίες εκτελούνται παράλληλα στους διαθέσιμους επεξεργαστές του. Το δεύτερο μηχάνημα, κατά την πρόσβαση σε κατάλληλο endpoint, λαμβάνει μια διαδικτυακή εφαρμογή με αρθρωτό σχεδιασμό, επιτρέποντας την παράλληλη εκτέλεση των λειτουργιών στους δικούς του επεξεργαστές.

Τα endpoints του πρώτου μηχανήματος προστατεύονται μέσω συστήματος εισόδου (login system), ώστε να διασφαλίζεται ότι μόνο εξουσιοδοτημένοι χρήστες με κατάλληλο ρόλο μπορούν να εκτελέσουν τις εργασίες. Στον αρθρωτό σχεδιασμό του δεύτερου μηχανήματος ενσωματώθηκαν λειτουργίες φιλτραρίσματος των δεδομένων που επιστρέφονται από το πρώτο, σύστημα ειδοποιήσεων για τον χρήστη, ζωντανή ανανέωση των δεδομένων και παρακολούθηση επιλεγμένων πληροφοριών.

Η εργασία αυτή αναδεικνύει τη σημασία του παράλληλου προγραμματισμού και της σωστής επικοινωνίας μεταξύ διαφορετικών συστημάτων. Ο συνδυασμός server με προστατευμένα endpoints και αρθρωτού frontend επιτρέπει την ασφαλή, γρήγορη και αποδοτική διαχείριση δεδομένων και εργασιών.Οι πρόσθετες λειτουργίες φιλτραρίσματος, ειδοποιήσεων και παρακολούθησης βελτιώνουν σημαντικά την εμπειρία του χρήστη και την αξιοπιστία του συστήματος.

5.3 Συμπεράσματα επί της ευρύτερης εφαρμογής του συστήματος

Η ανάπτυξη του παρόντος διαδικτυακού συστήματος έχει αποδείξει τη δυνατότητα ευέλικτης και αποτελεσματικής λειτουργίας σε περιβάλλοντα με διαφορετικές απαιτήσεις και χρήστες. Αν και η εφαρμογή σχεδιάστηκε με επίκεντρο το αεροδρόμιο «Μακεδονία», η αρχιτεκτονική της – με backend σε Python και Flask, frontend αρθρωτής σχεδίασης σε JavaScript, χρήση βάσης δεδομένων, παράλληλο προγραμματισμό με threading και αυτοματοποιημένη συλλογή δεδομένων μέσω Selenium – επιτρέπει την εύκολη προσαρμογή σε άλλες περιοχές. Τέτοιες εφαρμογές μπορεί να περιλαμβάνουν συστήματα Internet of Things (IoT), πλατφόρμες κοινωνικής δικτύωσης, διαδικτυακές υπηρεσίες παρακολούθησης δεδομένων ή οποιοδήποτε περιβάλλον απαιτεί ασφαλή, δυναμική και διαδραστική διαχείριση πληροφοριών σε πραγματικό χρόνο. Η ευελιξία και η επεκτασιμότητα του συστήματος

υπογραμμίζουν τη δυνατότητα αξιοποίησής του πέρα από το αρχικό πλαίσιο, καθιστώντας το ένα εργαλείο με ευρύ φάσμα εφαρμογών.

Για παράδειγμα, σε ένα σύστημα Internet of Things (IoT), ο server θα μπορούσε να τρέχει σε έναν Raspberry Pi (single-board computer), διαχειριζόμενος αισθητήρες και συσκευές.Το frontend θα μπορούσε να εμφανίζει σε πραγματικό χρόνο δεδομένα όπως θερμοκρασία, υγρασία ή κατάσταση συσκευών, παρέχοντας ειδοποιήσεις και δυνατότητες ελέγχου των συσκευών στους χρήστες.

5.4 Προοπτικές/ Επόμενα Βήματα

Ο διακομιστής ιστού θα μπορούσε να μεταφερθεί σε πλατφόρμες υπολογιστικού νέφους (cloud), εξασφαλίζοντας έτσι ότι η διαδικτυακή εφαρμογή θα είναι προσβάσιμη από οποιονδήποτε χρήστη, ανεξαρτήτως γεωγραφικής θέσης.

Μελλοντικά, η εφαρμογή δύναται να εμπλουτιστεί με προηγμένες λειτουργίες ανάλυσης δεδομένων και τεχνητής νοημοσύνης, όπως προβλέψεις καιρικών συνθηκών στη Θεσσαλονίκη, εκτιμήσεις καθυστερήσεων πτήσεων, παρακολούθηση επιβατικής κίνησης, συστήματα ειδοποιήσεων βασισμένα σε συμπεριφορικά δεδομένα χρηστών και ανάλυση προτιμήσεων καταστημάτων και προϊόντων. Τέτοιες επεκτάσεις θα ενίσχυαν τη χρηστικότητα, την ακρίβεια των πληροφοριών και την εμπειρία των χρηστών , καθιστώντας το σύστημα περισσότερο ευέλικτο και προσαρμοστικό σε διαφορετικά περιβάλλοντα και ανάγκες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. [1]. [Ηλεκτρονικό] https://el.wikipedia.org/wiki/Python.
2. [2]. *wiki_monty_python.* [Ηλεκτρονικό] https://el.wikipedia.org/wiki/Monty_Python.
3. [3]. *wiki_javascript.* [Ηλεκτρονικό] https://el.wikipedia.org/wiki/JavaScript.
4. [4]. *wiiki_sql.* [Ηλεκτρονικό] https://el.wikipedia.org/wiki/SQL.
5. [5]. *wiki_html.* [Ηλεκτρονικό] https://el.wikipedia.org/wiki/HTML.
6. [6]. *wiki_css.* [Ηλεκτρονικό] https://el.wikipedia.org/wiki/CSS.
7. [7]. *wiki_server.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Web_server.
8. [8]. *wiki_api.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/API.
9. [9]. *wiki_crud.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Create,_read,_update_and_delete.
10. [10]. *wiki_threading.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Thread_(computing).
11. [11]. *wiki_scraping.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Data_scraping.
12. [12]. *wiki_database.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Database.
13. [13]. *wiki_browser.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Web_browser.
14. [14]. *wiki_web_app.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Web_application.
15. [15]. *wiki_login.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Login.
16. [16]. *wiki_notification.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Notification_system.
17. [17]. *wiki_jinja.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Jinja_(template_engine).
18. [18]. *wii_navbar.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Navigation_bar.
19. [19]. *wiki_modular.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Modular_design.
20. [20]. *wiki_widget.* [Ηλεκτρονικό] https://en.wikipedia.org/wiki/Software_widget.
21. [21]. [Ηλεκτρονικό] https://code.visualstudio.com/.
22. [22]. *sqlitebrowser.* [Ηλεκτρονικό] https://sqlitebrowser.org/.
23. [23]. [Ηλεκτρονικό] https://www.python.org/.
24. [24]. [Ηλεκτρονικό] https://pypi.org/project/Flask/.
25. [25]. [Ηλεκτρονικό] https://docs.python.org/3/library/sqlite3.html.
26. [26]. [Ηλεκτρονικό] https://docs.python.org/3/library/threading.html.
27. [27]. [Ηλεκτρονικό] https://sqlite.org/.

ΠΑΡΑΡΤΗΜΑ Α

1.Python Server

```
from flask import
Flask,render_template,jsonify,request,session,redirect,url_for


#dimiourgia server
server=Flask(__name__)

# sinartisi ekinisis thread
def start_background_thread():
    thread = threading.Thread(target=scrapp_all,args=(300,), daemon=True)
    thread.start()

#decoration function prostasias dedomenon admin
def admin_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        # Check if session contains 'user_type' and if it's 'admin'
        if not session.get('user_type') == 'admin':
            return redirect(url_for('home'))  # Or redirect to home page
        return f(*args, **kwargs)
    return decorated_function


#endpoint pou servirei web_app.html
@server.route("/",methods=['GET'])
def home():
    return render_template("web_app.html")

#endpoint pou servirei ola ta nosokomeia os list of jsons
@server.route("/hospitals/",methods=['GET'])
def hospitals():
    return jsonify(get_all("hospitals"))

#prostateumeno endpoint pou servirei olous tous user (mono o admin exei
prosbasi)
@server.route("/users/",methods=['GET'])
@admin_required
def users():
    return jsonify(get_all("users"))
```

```
if __name__ == "__main__": # main program
    #ekkinisi server
    server.run(port=5001)
```

2.Javascript

```
import { Div } from "./Div.js";
 // sub component ImageSlider
function ImageSlider(image_url_list,time_image_appears_in_sec){
    let currentIndex = 0;
    let slider=Div(null,"slider-wrapper")
    let img=document.createElement("img")
    img.id="slider-image"
    img.src=image_url_list[0]
    img.alt="Slider"
    let controls=Div(null,"slider-controls")
    let prev_but=document.createElement("button")
    prev_but.innerText="◂"
    prev_but.onclick=function(){
        currentIndex = (currentIndex - 1) % image_url_list.length
        if (currentIndex<0){
            currentIndex=image_url_list.length-1
        }

        img.src=image_url_list[currentIndex]
    }
    let next_but=document.createElement("button")
    next_but.id="next_but"
    next_but.innerText="▸"
    next_but.onclick=function(){
        currentIndex = (currentIndex + 1) % image_url_list.length
        img.src=image_url_list[currentIndex]
    }
    controls.appendChild(prev_but)
    controls.appendChild(next_but)
    slider.appendChild(img)
    slider.appendChild(controls)
    const intervalId =setInterval(function(){
        if(slider.parentNode.parentNode){
            next_but.click();
        }
        else{
            clearInterval(intervalId);
        }
    }, 1000*time_image_appears_in_sec);
    return slider
}
export{ImageSlider}
```