ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής ανοιχτού κώδικα για τη διαχείριση κρατήσεων σε καταλύματα

Του φοιτητή
Χατζησπύρου Κωνσταντίνος
Αρ. Μητρώου: It174870

Επιβλέπουσα :
Ιωαννίδου Μελπομένη
Καθηγήτρια

Ημερομηνία 12-09-2025

Τίτλος Δ.Ε. Ανάπτυξη εφαρμογής ανοιχτού κώδικα
για τη διαχείριση κρατήσεων σε καταλύματα
Κωδικός Δ.Ε. 25218
Ονοματεπώνυμο φοιτητή Χατζησπύρου Κωνσταντίνος
Ονοματεπώνυμο εισηγητή Ιωαννίδου Μελίνα
Ημερομηνία ανάληψης Δ.Ε. 29-03-2025
Ημερομηνία περάτωσης Δ.Ε. 12-09-2025

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

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

«Για την οικογένειά μου»

Πρόλογος

Η παρούσα πτυχιακή εργασία εκπονήθηκε στο πλαίσιο ολοκλήρωσης των σπουδών μου και έχει ως αντικείμενο την ανάπτυξη μιας διαδικτυακής εφαρμογής για τη διαχείριση κρατήσεων καταλύματος με την ονομασία *Studio Liotrivi*.

Η επιλογή του θέματος έγινε με σκοπό να συνδυαστεί η θεωρητική γνώση που αποκτήθηκε κατά τη διάρκεια των σπουδών με την πρακτική εφαρμογή σύγχρονων τεχνολογιών ανάπτυξης λογισμικού. Μέσα από την εργασία αυτή επιχειρείται η δημιουργία ενός εύχρηστου, γρήγορου και ασφαλούς συστήματος, το οποίο καλύπτει τις ανάγκες τόσο των επισκεπτών όσο και των διαχειριστών ενός καταλύματος.

Η εκπόνηση της εργασίας αποτέλεσε μια ευκαιρία για εμβάθυνση σε έννοιες όπως η αρχιτεκτονική λογισμικού, η ανάπτυξη RESTful APIs, η διαχείριση βάσεων δεδομένων και η υλοποίηση μηχανισμών ασφαλείας. Παράλληλα, η διαδικασία περιλάμβανε την επίλυση τεχνικών προκλήσεων, την αξιολόγηση εργαλείων και την εφαρμογή βέλτιστων πρακτικών στον σχεδιασμό και την ανάπτυξη.

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς όσους συνέβαλαν στην υλοποίηση της εργασίας αυτής, παρέχοντας γνώσεις, υποστήριξη και ενθάρρυνση καθ’ όλη τη διάρκεια της προσπάθειας.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως στόχο την ανάπτυξη ενός σύγχρονου πληροφοριακού συστήματος online κρατήσεων για τουριστικό κατάλυμα. Το σύστημα βοηθά στο να καλυφθούν απαιτήσεις που έχουν να κάνουν με την ηλεκτρονική διαχείριση δωματίων, την πραγματοποίηση και παρακολούθηση κρατήσεων και την επικοινωνία μεταξύ επισκεπτών και διαχειριστών. Η υλοποίηση βασίζεται σε τεχνολογίες ανοιχτού κώδικα, ακολουθώντας σύγχρονες πρακτικές σχεδίασης και ανάπτυξης RESTful web εφαρμογών.

Η αρχιτεκτονική του συστήματος διαχωρίζει το frontend, το οποίο υλοποιήθηκε με HTML5, CSS3 και JavaScript, από το backend, το οποίο αναπτύχθηκε σε Node.js και Express.js και γίνεται σύνδεση με σχεσιακή βάση δεδομένων PostgreSQL. Παράλληλα, χρησιμοποιούνται μηχανισμοί ασφάλειας όπως JWT authentication και κρυπτογράφηση κωδικών με bcrypt, ενώ το σύστημα υποστηρίζει διαχείριση ρόλων χρηστών και αποστολή ειδοποιήσεων μέσω email.

Η ανάπτυξη ακολούθησε το σπειροειδές μοντέλο, κάνοντας δυνατή την επαναληπτική βελτίωση, τον εντοπισμό σφαλμάτων και την προσθήκη νέων λειτουργιών. Το τελικό αποτέλεσμα της εργασίας είναι μια πλήρως λειτουργική διαδικτυακή πλατφόρμα, η οποία προσφέρει στον χρήστη ένα φιλικό περιβάλλον πλοήγησης, δυνατότητα αναζήτησης διαθεσιμότητας, ολοκληρωμένη διαχείριση κρατήσεων και εργαλεία για επεκτάσεις στο μέλλον, όπως σύστημα αξιολογήσεων ή ενσωμάτωση πληρωμών.

Η εργασία κλείνει με την αξιολόγηση της εφαρμογής, επισήμανση περιορισμών και προτάσεις για μελλοντική ανάπτυξη, ενώ ο πηγαίος κώδικας διατίθεται ανοιχτά στο GitHub, προάγοντας τη συνεργασία και την τεχνολογική αυτονομία.

Development of an open source application for managing accommodation reservations

Chatzispyrou Konstantinos

Abstract

This thesis aims to develop a modern online booking information system for tourist accommodation, called Studio Liotrivi Booking System. The system helps to meet requirements related to electronic room management, making and monitoring reservations and communication between guests and managers. The implementation is based on open source technologies, following modern practices of designing and developing RESTful web applications.

The system architecture separates the frontend, which was implemented with HTML5, CSS3 and JavaScript, from the backend, which was developed in Node.js and Express.js and is connected to a PostgreSQL relational database. At the same time, security mechanisms such as JWT authentication and password encryption with bcrypt are used, while the system supports user role management and sending notifications via email.

The development followed the spiral model, enabling iterative improvement, error detection and the addition of new functions. The final result of the work is a fully functional web platform, which offers the user a friendly navigation environment, availability search capability, integrated booking management and tools for future extensions, such as a ratings system or payment integration.

The work concludes with an evaluation of the application, highlighting limitations and suggestions for future development, while the source code is openly available on GitHub, promoting collaboration and technological autonomy.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες σε όσους συνέβαλαν με οποιονδήποτε τρόπο στην ολοκλήρωση της παρούσας πτυχιακής εργασίας.

Ιδιαίτερη ευγνωμοσύνη οφείλω στην κυρία Μελίνα Ιωαννίδου, για την καθοδήγηση, τη συνεχή υποστήριξη και τις πολύτιμες συμβουλές της σε κάθε στάδιο της διαδικασίας. Ευχαριστώ επίσης τους συμφοιτητές και φίλους μου για την ηθική ενίσχυση, την ανταλλαγή ιδεών και τη συνεργασία μας όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να αφιερώσω την εργασία αυτή στην οικογένειά μου, η οποία με στήριξε αδιάκοπα με υπομονή, κατανόηση και αγάπη, δίνοντάς μου τη δύναμη να φτάσω στον στόχο μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Εικόνα 6.2: Συμπλήρωση και υποβολή φόρμας κράτησης

Εικόνα 6.3: Επεξεργασία κατάστασης κράτησης (έγκριση ή απόρριψη)

Εικόνα 6.4: Αξιολόγηση της εφαρμογής

Συντομογραφίες

Δ.Ε. Διπλωματική Εργασία

ΔΙΠΑΕ Διεθνές Πανεπιστήμιο Ελλάδος

Π.Ε. Πτυχιακή Εργασία

HTML – HyperText Markup Language

HTML5 – Πέμπτη έκδοση της HyperText Markup Language

CSS – Cascading Style Sheets

CSS3 – Τρίτη έκδοση των Cascading Style Sheets

JS – JavaScript

API – Application Programming Interface

RESTful API – API που ακολουθεί την αρχιτεκτονική REST (Representational State Transfer)

SMTP – Simple Mail Transfer Protocol

JWT – JSON Web Token

SQL – Structured Query Language

ERD – Entity Relationship Diagram

DFD – Data Flow Diagram

CORS – Cross-Origin Resource Sharing

AJAX – Asynchronous JavaScript and XML

CRUD – Create, Read, Update, Delete

UI – User Interface

UX – User Experience

MIT License – Massachusetts Institute of Technology License

1 Εισαγωγή

1.1 Αντικείμενο της Εργασίας

Η παρούσα πτυχιακή εργασία αφορά τη δημιουργία μιας διαδικτυακής εφαρμογής κρατήσεων για τουριστικά καταλύματα, που έχει ως κύριο στόχο να κάνει πιο εύκολη την διαχείριση του καταλύματος για τους ιδιοκτήτες, καθώς και να γίνει εύκολη η διαδικασία κράτησης για τους πελάτες. Το περιβάλλον του ιστότοπου επιτρέπει στον χρήστη να δει ποια είναι τα διαθέσιμα δωμάτια, να αναζητήσει δωμάτια με βάση τις ημερομηνίες και τον αριθμό επισκεπτών, καθώς και να υποβάλλει αιτήματα κράτησης.

Ταυτόχρονα, δημιουργήται ένα περιβάλλον backend που δίνει την δυνατότητα στον διαχειριστή να παρακολουθήσει και διαχειριστεί τις κρατήσεις, με μεγαλύτερη ευκολία. Το σύστημα είναι κατασκευασμένο ώστε να μπορεί να λειτουργεί με σχεσιακή βάση δεδομένων, να υποστηρίζει διαφορετικούς ρόλους χρηστών και να διατηρεί την ασφάλεια της εφαρμογής με την χρήση μηχανισμών, όπως αυθεντικοποίηση και επικοινωνία μέσω email. Ιδιαίτερη έμφαση δίνεται στο να γίνει η εφαρμογή εύκολη στη χρήση, να έχει ευέλικτο σχεδιασμό και να μπορεί να υπάρξει μελλοντικά επέκταση της εφαρμογής στο μέλλον .

1.2 Σκοπός και Στόχοι

Ο βασικός στόχος της συγκεκριμένης εργασίας είναι να αναπτυχθεί ένα πληροφοριακό σύστημα online κρατήσεων για τουριστικό κατάλυμα που να είναι σύγχρονο και να χρησιμοποιεί τις νέες τεχνολογίες, με την ονομασία *Studio Liotrivi Booking System*. Η εφαρμογή αυτή προσπαθεί να καλύψει ένα ευρύ φάσμα λειτουργικών απαιτήσεων που έχουν να κάνουν με την διαχείριση δωματίων ηλεκτρονικά από τον διαχειριστή, τις κρατήσεις των πελατών και τους τρόπους με τους οποίους μπορούν να επικοινωνούν οι επισκέπτες και ο διαχειριστής. Παράλληλα, μπορεί να χρησιμοποιηθεί ως ένα καλό παράδειγμα για το πώς μπορεί κάποιος να δημιουργήσει ένα λογισμικό, βασισμένο σε σύγχρονες τεχνολογίες και βέλτιστες πρακτικές υλοποίησης και δημοσίευσης εφαρμογών ανοιχτού κώδικα, το οποίο μπορεί να το δει και να το χρησιμοποιήσει ο καθένας.

Η εργασία παρουσιάζει την δημιουργία μιας πλήρως λειτουργικής διαδικτυακής πλατφόρμας που παρέχει:

- Φιλικό και απλό περιβάλλον πλοήγησης για τον τελικό χρήστη.
- Αναζήτηση διαθεσιμότητας ανάλογα με τις ημερομηνίες και τον αριθμό επισκεπτών που θέλει ο χρήστης.
- Συμπλήρωση και υποβολή φόρμας κράτησης με βασικά στοιχεία επισκέπτη.
- Αποστολή email επικοινωνίας προς τον διαχειριστή μέσω ενσωματωμένης φόρμας.
- Δυνατότητα χρήσης της εφαρμογής σε ελληνικά και αγγλικά, ανάλογα την προτίμηση του χρήστη.

Από την πλευρά του διαχειριστή, προσφέρονται βασικά εργαλεία διαχείρισης:

- Γίνεται επισκόπηση των αιτημάτων κράτησης και αξιολογούνται ανάλογα με την άποψη του διαχειριστή.
- Δίνεται η δυνατότητα να εγκριθούν, απορριφθούν ή να διαγραφούν εγγραφές.
- Δυναμική επισκόπηση των δεδομένων μέσω προστατευμένου dashboard.
- Ασφάλεια δεδομένων μέσω μηχανισμών authentication & authorization.

Η τεχνική υλοποίηση στηρίζεται σε τεχνολογίες όπως HTML, CSS και JavaScript για το frontend, καθώς και Node.js με Express.js για την υλοποίηση του backend. Για να επιτευχθεί με καλύτερο τρόπο η αποθήκευση των δεδομένων χρησιμοποιείται σχεσιακή βάση PostgreSQL, ενώ για την

ασφάλεια έχουν ενσωματωθεί μηχανισμοί, όπως bcrypt για την κρυπτογράφηση κωδικών και JWT για τη διαχείριση ταυτότητας χρηστών. Οι τεχνολογίες αυτές επιλέχθηκαν με κριτήριο το πόσο εύκολα χρησιμοποιούνται, το πόσο σταθερές είναι, την ευκολία στην επεκτασιμότητα και την κοινή χρήση στον πραγματικό κόσμο.

Τέλος, σημαντικός στόχος της εργασίας είναι να βοηθήσει στη συνεργασία μεταξύ των προγραμματιστών μέσα από την δημοσίευση του έργου ως open-source στο GitHub. Το γεγονός ότι ο κώδικα προσφέρεται για χρήση και συνοδεύεται από αναλυτικό τεχνικό και τεκμηριωτικό υλικό, δίνει την δυνατότητα σε άλλους προγραμματιστές καθώς και φοιτητές να μελετήσουν, να βελτιώσουν ή να επεκτείνουν την εφαρμογή ανάλογα με τις ανάγκες τους.

1.3 Κίνητρα και Ανάγκες που Οδήγησαν στην Ανάπτυξη

Στο πλαίσιο αυτής της πτυχιακής εργασίας αναπτύχθηκε ένας ιστότοπος για να υποστηρίξει την αυξανόμενη ζήτηση για επιχειρηματίες τουριστικών καταλυμάτων, παρέχοντας βασικά εργαλεία και πόρους ώστε να τους βοηθήσει να επιτύχουν και να επεκτείνουν αποτελεσματικά τις επιχειρήσεις τους. Τα μικρά καταλύματα σε αγροτικές ή νησιωτικές περιοχές αντιμετωπίζουν συχνά σημαντικές προκλήσεις τόσο στη διαχείριση κρατήσεων, όσο και στην προβολή του καταλύματός αλλά και την εξυπηρέτηση πελατών. Ο λόγος είναι ότι έχουν περιορισμένη πρόσβαση σε προηγμένα συστήματα κρατήσεων, πόρους μάρκετινγκ και αξιόπιστη σύνδεση στο διαδίκτυο. Αυτοί οι περιορισμοί εμποδίζουν την ικανότητά τους να προσελκύουν και να εξυπηρετούν αποτελεσματικά τους επισκέπτες.

Πολλοί ιδιοκτήτες ακινήτων βασίζονται σε μεγάλο βαθμό σε δημοφιλείς πλατφόρμες όπως η Booking.com και η Airbnb για να προσελκύσουν επισκέπτες και να αυξήσουν τις κρατήσεις τους. Ενώ αυτοί οι ιστότοποι παρέχουν σε πολύ σημαντικό βαθμό προβολή και δίνουν πρόσβαση σε ένα ευρύ κοινό, συχνά παρουσιάζουν σημαντικά μειονεκτήματα. Η προμήθεια την οποία καλούνται να πληρώσουν στις πλατφόρμες αυτές οι ιδιοκτήτες είναι αρκετά υψηλή, με αποτέλεσμα να μειώνονται τα κέρδη τους. Επίσης, αντιμετωπίζουν περιορισμούς στην προσαρμογή των καταχωρήσεων τους και στις αλληλεπιδράσεις με τους επισκέπτες. Επιπλέον, η χρήση αυτών των πλατφορμών μπορεί να μειώσει τον έλεγχο των ιδιοκτητών σχετικά με την επωνυμία τους καθώς και τη συνολική εμπειρία επισκεπτών, καθιστώντας δύσκολη τη δημιουργία μιας μοναδικής ταυτότητας και τη διατήρηση των προτύπων ποιότητας.

Ένα απλό σύστημα κρατήσεων ανοιχτού κώδικα, το οποίο είναι προσαρμόσιμο, έχει δημιουργηθεί ειδικά για τοπικούς και ανεξάρτητους οικοδεσπότες. Αυτή η καινοτόμος πλατφόρμα προσφέρει διαφανείς δυνατότητες διαχείρισης, μειώνοντας την ανάγκη για συμμετοχή τρίτων. Ο ανοιχτός χαρακτήρας του συστήματος αυτού επιτρέπει στους χρήστες και τους προγραμματιστές να προσαρμόζουν εύκολα και να επεκτείνουν τις δυνατότητές του, με το να προωθούν ένα περιβάλλον που στηρίζεται στην κοινότητα. Έτσι, δίνει τη δυνατότητα στους οικοδεσπότες να χειρίζονται αποτελεσματικά τις κρατήσεις τους διατηρώντας παράλληλα τον πλήρη έλεγχο των λειτουργιών του.

1.4 Μεθοδολογία

Αυτό το σύστημα κρατήσεων ανοιχτού κώδικα είναι ειδικά φτιαγμένο για τοπικούς οικοδεσπότες και προσαρμοσμένο στις ανάγκες τους. Αυτή η πλατφόρμα παρέχει διαφανείς λειτουργίες διαχείρισης που διασφαλίζουν ότι οι οικοδεσπότες μπορούν να βλέπουν και να ελέγχουν τις κρατήσεις τους με ευκολία. Η ευέλικτη σχεδίασή του βοηθάει στο να προσαρμοστεί με τέτοιο τρόπο ώστε να ταιριάζει στις ατομικές ανάγκες του κάθε οικοδεσπότη, ενώ η ενεργή υποστήριξη της κοινότητας προσφέρει συνεχή βοήθεια και κοινή γνώση. Δίνοντας τη δυνατότητα στους οικοδεσπότες να διαχειρίζονται αποτελεσματικά τις κρατήσεις τους ανεξάρτητα, αυτό το σύστημα βοηθάει και ενισχύει την τοπική φιλοξενία.

Για τη δημιουργία αυτης της πλατφόρμας συγκεντρώθηκαν οι απαραίτητες πληροφορίες ώστε να διασφαλιστεί η πλήρης κατανόηση των αναγκών που υπάρχουν. Χρειάστηκε να γίνει η ανάλυση του ρόλου που θα έχει η κάθε κατηγορία χρήστη, όπως για παράδειγμα είναι οι διαχειριστές και οι

πελάτες, ώστε να προσαρμόζονται ανάλογα οι λειτουργίες. Έπειτα σχεδιάστηκε τόσο μία ισχυρή βάση δεδομένων όσο και ένα σύστημα υποστήριξης, προκειμένου να δοθεί εμφαση στην ομαλή λειτουργία της πλατφόρμας και την σωστή διαχείριση του εκάστοτε τύπου δωματίου με στόχο την αποτελεσματική διαχείριση των κρατήσεων, τόσο από την πλευρά του πελάτη όσο και από την πλευρά του ίδιου του διαχειριστή.

Το backend δημιουργήθηκε χρησιμοποιώντας Node.js και Express.js. Έτσι παρέχεται ένα ισχυρό περιβάλλον διακομιστή. Επίσης χρησιμοποιήθηκε η PostgreSQL για αξιόπιστη αποθήκευση δεδομένων, χρησιμοποιώντας RESTful API με μορφή JSON για απρόσκοπτη επικοινωνία, ώστε αυτή να μην συναντά εμπόδια.. Η ασφάλεια διασφαλίστηκε μέσω του ελέγχου ταυτότητας JWT, ενώ το bcrypt χρησιμοποιήθηκε για την ασφαλή κατακερματισμό των κωδικών πρόσβασης χρηστών, ενισχύοντας τη συνολική ασφάλεια.

Το frontend κατασκευάστηκε χρησιμοποιώντας απλό HTML, CSS και JavaScript, δίνοντας έμφαση σε μια καθαρή και καλά οργανωμένη δομή. Αυτή η προσέγγιση εξασφάλισε έναν σαφή διαχωρισμό των ανησυχιών, καθιστώντας τον κώδικα ευκολότερο στη συντήρηση και την ενημέρωση. Επίσης, επέτρεψε την ανεμπόδιστη ενσωμάτωση του δυναμικού περιεχομένου, παρέχοντας στους χρήστες μια συναρπαστική εμπειρία.

Κατά τη διαδικασία ανάπτυξης, εφαρμόστηκε ολοκληρωμένη δοκιμή βάσει περιπτώσεων χρήσης για να διασφαλιστεί ότι η πλατφόρμα λειτουργεί αξιόπιστα, ότι τα δεδομένα παραμένουν ακριβή και ελαχιστοποιήθηκαν τα τρωτά σημεία ασφαλείας. Για τη διατήρηση της οργάνωσης και τη διευκόλυνση της συνεργασίας, ο έλεγχος έκδοσης έγινε με χρήση του Git και φιλοξενήθηκε στο GitHub. Έτσι, παρακολουθούνταν αποτελεσματικά οι αλλαγές και τα μέλη της ομάδας συνεργάζονταν εύκολα μεταξύ τους. Μετά την ολοκλήρωση, το έργο κυκλοφόρησε δημόσια ως πρωτοβουλία ανοιχτού κώδικα, προσκαλώντας τις συνεισφορές της κοινότητας και ενισχύοντας τη διαφάνεια. Αυτή η προσέγγιση εξασφάλισε ένα ισχυρό, ασφαλές και προσβάσιμο τελικό προϊόν.

1.5 Δομή της Εργασίας

Το Κεφάλαιο 2 αναλύονται οι βασικές θεωρητικές έννοιες και τα τεχνολογικά θεμέλια που είναι απαραίτητα για τη σύγχρονη ανάπτυξη Ιστού. Δίνει έμφαση σε βασικά εργαλεία, όπως το Node.js και το Express, για τη δημιουργία εφαρμογών στο backend, μαζί με το PostgreSQL για τη διαχείριση βάσεων δεδομένων. Το κεφάλαιο καλύπτει επίσης τεχνολογίες για το frontend όπως HTML, CSS και JavaScript, οι οποίες είναι ζωτικής σημασίας για τη δημιουργία διεπαφών χρήστη. Επιπλέον, αναλύονται εργαλεία ασφάλειας και ελέγχου ταυτότητας όπως το bcrypt για τους κωδικούς πρόσβασης, το JWT για τον έλεγχο ταυτότητας βάσει διακριτικών και το CORS για τη διαχείριση επικοινωνίας μεταξύ frontend και backend.

Το Κεφάλαιο 3 εξετάζονται διεξοδικά οι απαιτήσεις του συστήματος, ξεκινώντας με μια σαφή περιγραφή του βασικού προβλήματος που στοχεύει να αντιμετωπίσει το σύστημα. Προσδιορίζει βασικούς τύπους χρηστών, συγκεκριμένα πελάτες και διαχειριστές, επισημαίνοντας τον ξεχωριστό ρόλο που έχει ο καθένας καθώς και τις ανάγκες τους. Το κεφάλαιο περιγράφει, επίσης, τις λειτουργικές απαιτήσεις, όπως παραδείγματος χάρη τι πρέπει να κάνει το σύστημα, όσο και τις μη λειτουργικές απαιτήσεις, όπως η απόδοση και η ασφάλεια. Επιπλέον, χαρτογραφεί τις περιπτώσεις κύριας χρήσης, παρέχοντας μια ολοκληρωμένη επισκόπηση των βασικών λειτουργιών του συστήματος.

Το Κεφάλαιο 4 παρέχει μια ολοκληρωμένη ανάλυση του σχεδιασμού του συστήματος, διερευνώντας την αρχιτεκτονική της εφαρμογής, δηλαδή τη ροή δεδομένων μέσα στο σύστημα, απεικονίζοντας τον τρόπο με τον οποίο οι πληροφορίες μετακινούνται μεταξύ των στοιχείων. Το κεφάλαιο παρουσιάζει επίσης το Διάγραμμα Σχέσεων οντοτήτων (ERD) για την οπτικοποίηση των σχέσεων δεδομένων, περιγράφει τις διαδρομές υποστήριξης και τους ελεγκτές για τη διαχείριση αιτημάτων καθώς και την οργάνωση των αρχείων διεπαφής, διασφαλίζοντας μια σαφή κατανόηση της συνολικής δομής.

Το Κεφάλαιο 5 επικεντρώνεται κυρίως στην υλοποίηση του συστήματος. Στο συγκεκριμένο κεφάλαιο γίνεται περιγραφή του περιβάλλοντος στο οποίο αναπτύχθηκαν, καθώς και ανάλυση του τρόπου που

υλοποιήθηκαν βασικές λειτουργίες. Παρουσιάζεται δηλαδή η διαχείριση των κρατήσεων, η λειτουργικότητα εύρεσης διαθεσιμότητας, το σύστημα authentication (αυθεντικοποίησης) και οι επικοινωνίες με email μέσω της βιβλιοθήκης Nodemailer. Επιπλέον, περιλαμβάνονται πληροφορίες για τη δημοσίευση της εφαρμογής ως open-source έργο στο GitHub, καθώς και τη χρήση σχετικής άδειας (MIT).

Το Κεφάλαιο 6 αξιολογεί τις διαδικασίες δοκιμών του συστήματος, παρέχοντας περιγραφές διάφορων σεναρίων χρήσης για να εξασφαλιστεί ολοκληρωμένη κάλυψη. Εξηγεί επίσης τις λειτουργίες API, εντοπίζει πιθανά σφάλματα και προσφέρει βελτιώσεις που στοχεύουν στη βελτίωση της συνολικής αξιοπιστίας του συστήματος για καλύτερη απόδοση και ικανοποίηση των χρηστών.

Τέλος, στο Κεφάλαιο 7 συνοψίζονται τα κύρια ευρήματα και αποτελέσματα της εργασίας, επισημαίνοντας τα κύρια επιτεύγματα και την πρόοδο που σημειώθηκε σε όλο το έργο. Περιλαμβάνει τον απολογισμό των στόχων που επιτεύχθηκαν, αναγνωρίζει τους περιορισμούς και προσφέρει προτάσεις για το μέλλον, συμπεριλαμβανομένων βελτιώσεων όσον αφορά τα συστήματα πληρωμών και βελτιωμένη υποστήριξη φορητών συσκευών για να διασφαλιστεί μια καλύτερη εμπειρία χρήστη.

2 Θεωρητικό Υπόβαθρο & Τεχνολογίες

2.1 Τεχνολογίες Web (HTML/CSS/JS)

Την σημερινή εποχή οι web εφαρμογές χρησιμοποιούν ένα σύνολο θεμελιωδών τεχνολογιών που σκοπός τους είναι να εξασφαλίσουν τη σωστή δομή, την αισθητική και το πόσο διαδραστικές είναι οι ιστοσελίδες. Στην εφαρμογή που δημιουργήθηκε για τις ανάγκες της πτυχιακής, η ανάπτυξη του κώδικα για την υλοποίηση του frontend έχει σαν βάση τρεις τεχνολογίες: HTML5, CSS3 και JavaScript. Κάθε μία από αυτές έχει συγκεκριμένο ρόλο και βοηθάει στο πόσο καλά λειτουργεί η εφαρμογή και στην καλή εμπειρία του χρήστη [1],[2].

Η HTML5 (HyperText Markup Language έκδοση 5) είναι η γλώσσα σήμανσης που βάσει αυτής καθορίζεται η εμφάνιση, η δομή και το περιεχόμενο της ιστοσελίδας στην οποία χρησιμοποιείται. Μέσω της HTML, ορίζονται τα στοιχεία που συνθέτουν μια σελίδα, όπως κεφαλίδες, παράγραφοι, λίστες, πίνακες, εικόνες και φόρμες. Με την χρήση της HTML5 υπάρχουν νέες δυνατότητες, όπως η συμμετοχή πολυμέσων (βίντεο, ήχος) στην ιστοσελίδα χωρίς την ανάγκη πρόσθετων plugins, όπως επίσης υποστηρίζεται η τοπική αποθήκευση δεδομένων (local storage). Αυτά είναι κάποια απο τα χαρακτηριστικά που δίνουν την δυνατότητα στον προγραμματιστή να δημιουργήσει πιο σύγχρονες και πιο λειτουργικές προς τον χρήστη web εφαρμογές.

Το CSS3 (Cascading Style Sheets έκδοση 3) είναι η γλώσσα που είναι απαραίτητη για την καλή και συνεπή εμφάνιση της εφαρμογής που δημιουργήθηκε. Μέσω του CSS3, μπορεί να γίνει έλεγχος για τη διάταξη των στοιχείων, τα χρώματα της ιστοσελίδας, τις γραμματοσειρές και τα περιθώρια των κειμένων, τις σκιές, τις μεταβάσεις και τα animations που χρησιμοποιούνται, τα οποία βοηθούν στο να βελτιωθεί σημαντικά η αισθητική και το πόσο εύχρηστη είναι η εφαρμογή. Επιπλέον, το CSS3 έχει την δυνατότητα να χρησιμοποιεί τεχνικές responsive design, δηλαδή να μπορεί να προσαρμοστεί η εμφάνιση ανάλογα με το μέγεθος και τον τύπο της συσκευής (desktop, tablet, κινητό), κάτι που δεν μπορεί να λείπει απο μια σύγχρονη εφαρμογή.

Η JavaScript είναι μια γλώσσα προγραμματισμού που παρέχει τη δυνατότητα στην εφαρμογή να είναι δυναμική και διαδραστική όταν την χρησιμοποιεί ο πελάτης (client-side). Με τη χρήση της JavaScript, η ιστοσελίδα έχει την δυνατότητα να πραγματοποιεί τις ενέργειες του χρήστη που την χρησιμοποιεί, χωρίς να χρειάζεται να "φορτώνει" ξανά ολόκληρη η σελίδα. Αυτό περιλαμβάνει ενέργειες όπως η επικύρωση φορμών, η εμφάνιση ή απόκρυψη περιεχομένου, η δυναμική ενημέρωση δεδομένων, και η επικοινωνία με τον server μέσω τεχνολογιών όπως το AJAX (Asynchronous JavaScript and XML). Στην ιστοσελίδα που δημιουργήθηκε για την παρούσα πτυχιακή, η JavaScript δίνει την δυνατότητα στον χρήστη να βλέπει την ώρα εκείνη που την χρησιμοποιεί, τη διαθεσιμότητα των δωματίων και να αλλάζει τη γλώσσα του περιβάλλοντος, δίνοντάς του με τον τρόπο αυτό μια βελτιωμένη εμπειρία [3].

Στην δημιουργία web εφαρμογών, εκτός από τις βασικές τεχνολογίες που προαναφέρθηκαν, συχνά χρησιμοποιούνται και σύγχρονα frontend frameworks ή βιβλιοθήκες, όπως το React, Angular ή Vue, που κάνουν πιο εύκολη τη δημιουργία σύνθετων και διαχειρίσιμων διεπαφών χρήστη. Παρόλα αυτά, στην παρούσα εργασία έγινε χρήση «καθαρής» HTML, CSS και JavaScript, ώστε να παραμείνει απλή και να γίνει πλήρως κατανοητή η λειτουργία της κάθε τεχνολογίας, χωρίς την προσθήκη επιπλέον επιπέδων πολυπλοκότητας.

Σαν συμπέρασμα έχουμε οτι η σωστή χρήση και ο σωστός συνδυασμός των HTML5, CSS3 και JavaScript εξασφαλίζει τη δημιουργία μιας web εφαρμογής που είναι λειτουργική, εύκολη στη χρήση, διαδραστική και ανταποκρίνεται στις απαιτήσεις των χρηστών της σύγχρονης εποχής. Η εφαρμογή που δημιουργήθηκε εδώ, κάνει χρήση αυτών των τεχνολογιών ώστε να παρέχει σε όσους την χρησιμοποιούν εύκολη πρόσβαση σε πληροφορίες, διαδραστικότητα και ευελιξία στη χρήση. Τα στοιχεία κάνουν μια πλατφόρμα για ενοικιαζόμενα δωμάτια επιτυχημένη.

2.2 Node.js και Express

Σημαντικό κομμάτι για την σωστή λειτουργία της εφαρμογής έχει το backend μέρος της εφαρμογής, που είναι η ραχοκοκαλιά της εύρυθμης λειτουργίας , πάνω στην οποία πραγματοποιούνται όλες οι κρίσιμες διαδικασίες που δεν μπορεί να δει ο τελικός χρήστης αλλά σχετίζονται άμεσα με το πόσο σωστά λειτουργεί η πλατφόρμα. Στην εφαρμογή ενοικίασης δωματίων, που δημιουργήθηκε στο πλαίσιο της παρούσας εργασίας, το backend κομμάτι αναπτύχθηκε με την χρήση της πλατφόρμα Node.js και συνδυάστηκε με το web framework Express.js, τα χαρακτηριστικά των οποίων προσφέρουν ένα σύγχρονο, ευέλικτο και αποδοτικό περιβάλλον και βοηθάνε στην δημιουργία δυναμικών web υπηρεσιών.

Το **Node.js** είναι μια πλατφόρμα η οποία δίνει την δυνατότητα στον server να εκτελεί κώδικα JavaScript στο περιβάλλον του [4]. Βασίζεται στον V8 JavaScript engine της Google, που είναι γνωστός για την υψηλή ταχύτητα εκτέλεσης [5]. Ο Node.js χρησιμοποιείτα γιατί είναι ασύγχρονο και event-driven μοντέλο ώστε να μπορεί ο server να έχει τη δυνατότητα να διαχειρίζεται πολλά αιτήματα ταυτόχρονα, χωρίς να έχει πρόβλημα με μπλοκαρίσματα στην ροή εκτέλεσης. Αυτή η διαδικασία έχει ως αποτέλεσμα οι εφαρμογές να ανταποκρίνονται γρήγορα, χωρίς να υπάρχει μεγάλη καθυστέρηση και χωρίς χαμηλή κλιμακούμενη απόδοση.

Ο σκοπός της χρήσης JavaScript από τον προγραμματιστή, τόσο στο frontend όσο και στο backend όπως έγινε και στην παρούσα εργασία, είναι να δουλεύει με μόνο μια γλώσσα για την ολοκλήρωση της εφαρμογής. Έτσι μπορεί να γίνεται πιο εύκολα η ανάπτυξη και η συντήρηση του κώδικα. Για τη χρήση του node.js είναι πάρα πολύ σημαντικό το γεγονός ότι διαθέτη το npm (Node Package Manager), το οποίο αποτελείται από ένα τεράστιο οικοσύστημα πακέτων και βιβλιοθηκών, που κάνουν πιο εύκολη την προσθήκη λειτουργικοτήτων στην εφαρμογή.

Πάνω απο το Node.js τρέχει το **Express.js.** Το Express.js είναι ένα απλό και εύχρηστο εργαλείο για την δημιουργία ιστοσελίδων. Η χρήση του κάνει πιο απλή και κατανοητή την δομή για την ανάπτυξη RESTful API, που βοηθάει το προγραμματιστή να δημιουργήσει διαφορετικές σελίδες, να χειριστεί αιτήσεις HTTP και να διαχειριστεί πιο αποδοτικά το ενδιάμεσο λογισμικό (middleware) [6].

Χρησιμοποιώντας το Express.js, η εφαρμογή έχει τα endpoints που αντιστοιχούν σε λειτουργίες όπως:

- **Διαχείριση χρηστών:** Εγγραφή, σύνδεση, ενημέρωση προφίλ
- **Διαχείριση δωματίων:** Προβολή, προσθήκη, τροποποίηση και διαγραφή δωματίων
- **Κρατήσεις:** Δημιουργία, ακύρωση και ενημέρωση κρατήσεων
- **Ενημέρωση διαθεσιμότητας:** Ενημέρωση της διαθεσιμότητας σε πραγματικό χρόνο, ώστε ο χρήστης να βλέπει μόνο τα διαθέσιμα δωμάτια

Το Express διαθέτει επίσης και μηχανισμούς για την διαχείριση σφαλμάτων (error handling), που παίζουν σημαντικό ρόλο στο πόσο αξιόπιστη είναι η εφαρμογή, και προσφέρει εύκολη ενσωμάτωση middleware για την ασφάλεια, όπως authentication και logging.

Συνοψίζοντας, το backend της εφαρμογής χρησιμοποιεί διαφορετικά εργαλεία που λειτουργούν όλα μαζί σαν ομάδα. Αυτό βοηθά να καλλιεργηθεί η βεβαιότητα ότι η εφαρμογή είναι ασφαλής, λειτουργεί καλά και είναι εύκολη στη χρήση. Το Node.js προσφέρει την πλατφόρμα εκτέλεσης, το Express.js οργανώνει τη ροή και την επικοινωνία μέσω RESTful endpoints, ενώ η PostgreSQL κάνει αποθήκευση των δεδομένων.

Η χρήση των βιβλιοθηκών pg, bcryptjs και jsonwebtoken συμβάλλει στην ασφάλεια και την αξιοπιστία της εφαρμογής, εξασφαλίζοντας την προστασία των δεδομένων και την ομαλή αλληλεπίδραση με τους χρήστες.

Η αρχιτεκτονική αυτή επιτρέπει την κλιμάκωση της εφαρμογής. Επίσης την κάνει πιο εύκολη στις διορθώσεις και προσφέρει την δυνατότητα να γίνουν προσθήκες με νέες λειτουργίες και επεκτάσεις αργότερα.

2.3 PostgreSQL

Η σωστή επιλογή και η ρύθμιση της βάσης δεδομένων είναι πολύ σημαντική για να λειτουργεί καλά ένας ιστότοπος, ειδικά όταν βοηθά στην παρακολούθηση των κρατήσεων, των χρηστών και των διαθέσιμων δωματίων. Στο πλαίσιο της εργασίας αυτής, η επιλογή της PostgreSQL ως σύστημα διαχείρισης σχεσιακής βάσης δεδομένων (RDBMS) έγινε με βασικότερο κριτήριο την ύπαρξη ισορροπίας μεταξύ ευελιξίας, απόδοσης και αξιοπιστίας [3].

Η χρήση της PostgreSQL προσφέρει μια ολοκληρωμένη υποστήριξη ACID (Atomicity, Consistency, Isolation, Durability), η οποία είναι πολύ σημαντική για συστήματα κρατήσεων όπου τα δεδομένα πρέπει να παραμένουν ακέραια ακόμα και υπό συνθήκες πολλαπλών ταυτόχρονων αιτημάτων. Παράλληλα, βοηθά στην σύνδεση διαφορετικών πληροφοριών μεταξύ πινάκων με την βοήθεια πρωτευόντων και ξένων κλειδιών. Αυτό κάνει ευκολότερη την διαδικασία να διατηρούνται οργανωμένα και να "συνεργάζονται" με ομαλό τρόπο σύνθετα δεδομένα όπως τα δωμάτια, οι κρατήσεις και οι εικόνες [7], [8].

Η βάση δεδομένων σχεδιάστηκε με τέτοιο τρόπο ώστε να υποστηρίζει πλήρως τις λειτουργίες της εφαρμογής. Έχουν δημιουργηθεί, στην βάση, ξεχωριστοί πίνακες για τους χρήστες (users), τα δωμάτια (rooms), τις κρατήσεις (bookings) και τις εικόνες (room_images). Η σύνδεση μεταξύ των οντοτήτων βασίζεται σε σχέσεις τύπου 1:N. Για παράδειγμα κάθε δωμάτιο μπορεί να έχει πολλές εικόνες και πολλές κρατήσεις, αλλά κάθε κράτηση συνδέεται με ένα μόνο δωμάτιο. Αυτό το μοντέλο βοηθά να γίνεται γρήγορα η εύρεση μια κράτησης, να γίνει υποβολεί ή επεξεργασία αυτής της κράτησής όποτε είναι απαραίτητο.

Για να γίνει εφικτή η σύνδεση μεταξύ του backend της εφαρμογής (Node.js + Express) και της PostgreSQL, έγινε χρήση της βιβλιοθήκης pg (γνωστή και ως node-postgres) [9]. Αυτή η βιβλιοθήκη επιτρέπει την ασφαλή εκτέλεση ερωτημάτων SQL μέσω JavaScript, αξιοποιώντας ασύγχρονες κλήσεις και connection pooling. Με την βοήθεια της pg, η εφαρμογή μπορεί να ανακαλεί δεδομένα (π.χ. "ποια δωμάτια είναι διαθέσιμα για 2 άτομα από 10/7 έως 13/7"), να προσθέτει νέες κρατήσεις, να ενημερώνει καταστάσεις μια κράτησης (π.χ. approved, rejected) ή να διαγράφει τις εγγραφές που δεν είναι πλέον απαραίτητες.

Ένα απο τα πλεονεκτήματα της pg είναι η δυνατότητα χρήσης prepared statements, τα οποία εκτός του ότι βελτιώνουν την ταχύτητα ερωτημάτων που επαναλαμβάνονται, βοηθούν και στο να μειωθεί σημαντικά ο κίνδυνος επιθέσεων τύπου SQL injection, κάνοντας την ασφάλεια της εφαρμογής πιο δυνατή. Η χρήση prepared statements έγινε κυρίως στις λειτουργίες εισαγωγής νέας κράτησης και σύνδεσης διαχειριστών, όπου γίνεται επεξεργασία ευαίσθητων δεδομένων είναι τα ονόματα και οι κωδικοί πρόσβασης.

Η PostgreSQL υποστηρίζει επίσης συναλλαγές (transactions). Για παράδειγμα, όταν γίνεται κράτηση, το σύστημα πρώτα κάνει έλεγχο για το αν το δωμάτιο είναι διαθέσιμο για τις ημερομηνίες που επέλεξε ο χρήστης και στη συνέχεια καταχωρεί τα στοιχεία της κράτησης σε ένα ενιαίο block. Αν για οποιονδήποτε λόγο αποτύχει κάποια ενδιάμεση εντολή (π.χ. αποτυχία σύνδεσης ή έλλειψη πόρων), τότε γίνεται ακύρωση της όλης διαδικασίας και η βάση ξαναγυρνάει στην προηγούμενη σταθερή κατάσταση.

Στην πράξη, η χρήση της PostgreSQL κάνει πιο εύκολη την ανάπτυξη, την κατανόηση και την επεκτασιμότητα της εφαρμογής. Η δυνατότητα που έχει, να εκτελεί γρήγορα τα φίλτρα (π.χ. "δώσε όλα τα δωμάτια με χωρητικότητα ≥ 3 και τιμή < 90 ευρώ") παίζει σημαντικό ρόλο στη δημιουργία της δυναμικής αναζήτησης. Επίσης, η ενσωμάτωση χρονικών σημάνσεων (timestamps) στους πίνακες bookings και users βοήθησε στον εντοπισμό των πιο πρόσφατων ενεργειών και στην υλοποίηση προβολής ιστορικού.

Συνολικά, η επιλογή της PostgreSQL φαίνεται να είναι η πιο κατάλληλη για τις ανάγκες της εφαρμογής, καθώς προσφέρει τεκμηριωμένη και ισχυρή υποδομή που υποστηρίζει τόσο την λειτουργία της εφαρμογής όπως είναι όσο και επεκτάσεις που μπορεί να πραγματοποιηθούν στο μέλλον, όπως η εισαγωγή αξιολογήσεων πελατών ή η διασύνδεση με σύστημα πληρωμών.

2.4 RESTful API

Για να μπορέσουν να επικοινωνήσουν το frontend και το backend κομμάτι της εφαρμογής, χρησιμοποιήθηκε ένα RESTful API (Representational State Transfer) [10]. Τα API (Application Programming Interfaces) είναι χρήσιμα λόγω της δυνατότητας που έχουν να βοηθούν πολλά διαφορετικά συστήματα να επικοινωνούν μεταξύ τους, χρησιμοποιώντας συγκεκριμένους κανόνες και μορφές για την ανταλλαγή δεδομένων. Στην εφαρμογή αυτή, επιτρέπεται η αποστολή και η λήψη δεδομένων μέσω των HTTP μεθόδων, όπως GET, POST, PUT και DELETE, λόγω ενός συνόλου από endpoints που δίνεται από το RESTful API.

Για να πραγματοποιηθεί η επικοινωνία χρησιμοποιείται η μορφή JSON (JavaScript Object Notation), η οποία είναι ελαφριά, ευανάγνωστη και αναλύεται πιο εύκολα από εφαρμογές frontend. Αυτό σημαίνει ότι το μέρος που φαίνεται (το μπροστινό μέρος) και το μέρος που λειτουργεί στα παρασκήνια (το πίσω μέρος) δεν χρειάζεται να γνωρίζουν τα πάντα το ένα για το άλλο. Το frontend απλώς στέλνει και λαμβάνει πληροφορίες με την χρήση του JSON, χωρίς να χρειάζεται να γνωρίζει πώς ο διακομιστής διατηρεί ή χρησιμοποιεί τα δεδομένα [11].

Με το RESTful API γίνεται, για παράδειγμα, η ανάκτηση των διαθέσιμων δωματίων, η αποστολή νέας κράτησης, η εγγραφή και σύνδεση χρηστών ή η ενημέρωση των στοιχείων τους. Επιπρόσθετα, η αρχιτεκτονική REST βοηθά ώστε να έχει το σύστημα να επεκταθεί και να είναι εύκολο στη συντήρησή του, κάνοντας πιο εύκολες τις μελλοντικές προσθήκες λειτουργιών χωρίς να χρειάζεται να δημιουργηθεί απο την αρχή η εφαρμογή.

2.5 Τεχνολογίες Αυθεντικοποίησης (JWT, bcrypt)

Η ασφάλεια των χρηστών είναι μια απο τις βασικές προτεραιότητες για κάθε διαδικτυακή εφαρμογή, ειδικά όταν η εφαρμογή περιέχει ευαίσθητα προσωπικά δεδομένα, όπως στοιχεία σύνδεσης, αιτήματα κρατήσεων και πρόσβαση σε λειτουργίες διαχειριστή. Στο πλαίσιο της παρούσας εργασίας, προκειμένου να ενισχυθεί η προστασία των δεδομένων και να εξασφαλιστεί η ασφαλής πρόσβαση στις λειτουργίες του συστήματος, αξιοποιήθηκαν δύο τεχνολογίες αυθεντικοποίησης, οι οποίες είναι αρκετά γνωστές στο χώρο: το bcrypt για το hashing των κωδικών πρόσβασης και το JWT (JSON Web Tokens) για τη διαχείριση των συνδεδεμένων χρηστών [12],[13].

Χρήση της βιβλιοθήκης bcrypt γίνεται κυρίως κατά την εγγραφή και σύνδεση των διαχειριστών του συστήματος. Όταν ένας χρήστης της εφαρμογής δημιουργεί έναν νέο λογαριασμό ή τον προσθέτει ο προγραμματιστής, ο κωδικός πρόσβασης δεν αποθηκεύεται απευθείας στη βάση δεδομένων, αλλά θα υποστεί μια επεξεργασία μέσω του αλγορίθμου hashing που παρέχει το bcrypt. Συγκεκριμένα, προστίθεται ένα τυχαίο στοιχείο που ονομάζεται salt και το οποίο καθιστά κάθε hashed κωδικό μοναδικό, ακόμα και αν δύο χρήστες έχουν επιλέξει τον ίδιο κωδικό. Η διαδικασία αυτή δυσκολεύει σημαντικά την αποκρυπτογράφηση των κωδικών σε περίπτωση που η βάση δεδομένων παραβιαστεί, ενώ προστατεύει και από γνωστές μεθόδους επιθέσεων, όπως τα dictionary attacks ή τα rainbow tables.

Παράλληλα, για να γίνει διαχείριση των ενεργών χρηστών και να διατηρηθεί η αυθεντικοποίησή τους καθ' όλη τη διάρκεια της χρήσης της εφαρμογής, γίνεται εφαρμογή του πρότυπου JWT (JSON Web Token). Το JWT επιτρέπει στον διακομιστή να εκδώσει ένα κωδικοποιημένο ψηφιακό "διακριτικό" (token) όταν ένας χρήστης πραγματοποιεί επιτυχημένη σύνδεση. Το διακριτικό αυτό περιλαμβάνει βασικές πληροφορίες για τον χρήστη, όπως για παράδειγμα το αναγνωριστικό του (user ID) και τον ρόλο του (π.χ. admin), και στη συνέχεια δεσμεύεται με ένα μυστικό "κλειδί". Το token που έχει παραχθεί θα αποσταλεί στον client και μπορεί να γίνει αποθήκευσή του τοπικά, συνήθως στο localStorage ή σε cookie [14], [15], [16].

Κάθε φορά που ο χρήστης προσπαθεί να πραγματοποιήσει μια ενέργεια που απαιτεί εξουσιοδότηση (όπως η προβολή κρατήσεων ή η επεξεργασία δωματίων), ο browser στέλνει το token μαζί με το αίτημα στον διακομιστή. Ο διακομιστής κάνει επαλήθευση της εγκυρότητας του token και έχει τη δυνατότητα είτε να επιτρέπει είτε απορρίπτει την πρόσβαση ανάλογα με τα δεδομένα που αυτό περιέχει. Η προσέγγιση αυτή επιτρέπει στο backend να παραμένει stateless, δηλαδή δίνει τη

δυνατότητα να μη αποθηκεύει πληροφορίες σύνδεσης για κάθε χρήστη. Έτσι το σύστημα διατηρείται απλό και αποδοτικό.

Ο συνδυασμός bcrypt και JWT προσφέρει ένα ισχυρό και σύγχρονο μοντέλο ασφάλειας, το οποίο μπορεί να επεκταθεί. Από τη μία πλευρά, εξασφαλίζεται ότι οι κωδικοί πρόσβασης αποθηκεύονται με ασφαλή τρόπο, και από την άλλη, η εφαρμογή μπορεί να διαχειρίζεται χρήστες που παραμένουν συνδεδεμένοι για συγκεκριμένο χρονικό διάστημα χωρίς να απαιτείται επανασύνδεση σε κάθε νέα ενέργεια. Η λογική αυτή ενισχύει την εμπειρία χρήστη, αποτρέπει μη εξουσιοδοτημένη πρόσβαση και συμβάλλει καθοριστικά στη γενικότερη ασφάλεια της εφαρμογής.

Η εφαρμογή των παραπάνω τεχνολογιών επιτυγχάνεται μέσω της χρήσης των βιβλιοθηκών bcryptjs και jsonwebtoken στο περιβάλλον του Node.js. Η εφαρμογή τους αποδείχθηκε απλή στο πώς συντάσσεται και ιδιαίτερα αποτελεσματική στην πράξη, με το υπολογιστικό κόστος να είναι πολύ μικρό για τον διακομιστή. Επίσης, υπήρξαν σημαντικά πλεονεκτήματα ασφάλειας. Σημαντικό επίσης είναι το γεγονος ότι οι τεχνολογίες αυτές είναι open source πράγμα το οποίο επιτρέπει τη διαρκή υποστήριξη από την κοινότητα και τη συνεχή εξέλιξή τους.

Η αυθεντικοποίηση αποτελεί ένα από τα σημαντικότερα κομμάτια κάθε σύγχρονης διαδικτυακής εφαρμογής. Το να γίνει η επιλογή σωστού συνδυασμού εργαλείων και τεχνικών διασφαλίζει εκτός απο τη σωστή λειτουργία της εφαρμογής, και την προστασία των χρηστών, κάτι το οποίο είναι απαραίτητο, ειδικά σε εφαρμογές όπως το σύστημα κρατήσεων, όπου η διαχείριση και η πρόσβαση σε ευαίσθητα δεδομένα πρέπει να γίνεται με αυστηρό έλεγχο.

2.6 Open Source και Άδειες Χρήσης

Η ανάπτυξη της εφαρμογής υποστηρίχθηκε από την χρήση του Git, ενός συστήματος ελέγχου εκδόσεων (version control system) το οποίο έχει καθιερωθεί ως βασικό εργαλείο στην οργάνωση και διαχείριση λογισμικού. Το Git επιτρέπει στους προγραμματιστές να καταγράφουν κάθε αλλαγή στον πηγαίο κώδικα με λεπτομέρεια, να συνεργάζονται αποτελεσματικά σε ομάδες, καθώς και να διαχειρίζονται πολλαπλές εκδόσεις του έργου μέσω διακλαδώσεων (branches). Με αυτόν τον τρόπο, είναι λιγότερο πιθανό να συμβούν λάθη και είναι πιο εύκολο να γίνει επαναφορά σε προηγούμενη έκδοση εάν κάτι πάει στραβά. Αυτό βοηθά το λογισμικό να παραμένει σταθερό και να βελτιώνεται ομαλά [17].

Ο πηγαίος κώδικας της εφαρμογής φιλοξενείται δημόσια στο GitHub, η οποία είναι μια από τις μεγαλύτερες πλατφόρμες διαχείρισης αποθετηρίων Git. Το GitHub παρέχει εργαλεία για version control, για τη διαχείριση θεμάτων (issues), για αυτόματη κατασκευή (CI/CD) και συνεργασία προγραμματιστών σε πραγματικό χρόνο. Η κοινή χρήση του έργου στο GitHub σημαίνει ότι όχι μόνο οι άνθρωποι που το δημιούργησαν μπορούν να το δουν και να το χρησιμοποιήσουν, αλλά και πολλοί άλλοι προγραμματιστές και χρήστες από όλο τον κόσμο έχουν τη δυνατότητα να το δουν, να μάθουν από αυτό, ακόμη και να βοηθήσουν στη βελτίωσή του.

Ο λόγος που αποφασίστηκε να γίνει η εφαρμογή ανοιχτού κώδικα είναι για να μπορούν όλοι να δουν πώς λειτουργεί. Αυτό βοηθά τους ανθρώπους να το κατανοήσουν καλύτερα, να μοιράζονται ιδέες και να συνεργάζονται για να το κάνουν ακόμα καλύτερο. Όταν ο πηγαίος κώδικας είναι ανοιχτός σε όλους, οποιοσδήποτε μπορεί να εντοπίσει σφάλματα, να προτείνει βελτιώσεις ή να προσθέσει νέες δυνατότητες. Αυτή η ομαδική εργασία βοηθά το λογισμικό να βελτιώνεται και να αναπτύσσεται με την πάροδο του χρόνου.

Για τη νομική κάλυψη και καθορισμό των όρων χρήσης, η εφαρμογή διατίθεται υπό την άδεια MIT (Massachusetts Institute of Technology License), η οποία αποτελεί μία από τις πιο διαδεδομένες και ευέλικτες άδειες ανοιχτού κώδικα. Η άδεια MIT είναι σαν ένα σύνολο φιλικών κανόνων για την κοινή χρήση προγραμμάτων υπολογιστή και σύμφωνα με αυτή είναι δυνατόν να γίνει χρήση του προγράμματος με διάφορους τρόπους, όπως η αντιγραφή, η αλλαγή, το μοίρασμα σε τρίτους ή ακόμα και να γίνει εμπορική χρήση του, χωρίς παραπάνω άδεια. Το μόνο που είναι απαραίτητο να πραγματοποιηθεί είναι να υπάρχει η αρχική σημείωση που να αναφέρεται στο ποιος το έφτιαξε [18], [19].

Η άδεια MIT διευκολύνει τους προγραμματιστές και τις εταιρείες να χρησιμοποιούν και να μοιράζονται το λογισμικό στα δικά τους έργα και βοηθά περισσότερους ανθρώπους να χρησιμοποιούν και να βελτιώνουν τον κώδικα. Ταυτόχρονα, διασφαλίζει ότι τα άτομα που δημιούργησαν το λογισμικό δεν φέρουν ευθύνη εάν κάτι πάει στραβά όταν το χρησιμοποιούν άλλοι.

Επιπλέον, η φιλοσοφία του open source προάγει την διαφάνεια και την ανοιχτή επικοινωνία, μειώνοντας την εξάρτηση από κλειστό κώδικα και αυξάνοντας την εμπιστοσύνη των χρηστών προς την εφαρμογή. Η κοινή χρήση κώδικα με άλλους, ώστε να μπορούν να τον δουν και να βοηθήσουν στη βελτίωσή του, τον καθιστά ασφαλέστερο. Όταν περισσότεροι άνθρωποι ελέγχουν τον κώδικα, είναι πιο πιθανό να βρουν και να διορθώσουν τυχόν τρωτά σημεία.

Τέλος, η δημοσίευση του έργου στο GitHub και η επιλογή μιας ανοιχτής άδειας χρήσης υποστηρίζουν τη φιλοσοφία της συνεργασίας και της συλλογικής εξέλιξης του λογισμικού, η οποία αποτελεί θεμέλιο λίθο της σύγχρονης πληροφορικής και προγραμματιστικής κοινότητας.

3 Ανάλυση Απαιτήσεων

3.1 Περιγραφή προβλήματος

Τα τελευταία χρόνια, οι νέες ψηφιακές τεχνολογίες εξαπλώθηκαν πολύ γρήγορα και αυτό άλλαξε τον τρόπο με τον οποίο τα ξενοδοχεία, τα εστιατόρια και οι ταξιδιωτικές εταιρείες κάνουν τη δουλειά τους. Παρόλα αυτά, πολλές μικρές επιχειρήσεις, όπως τα ενοικιαζόμενα δωμάτια, εξακολουθούν να χρησιμοποιούν παλιομοδίτικες μεθόδους για την εξυπηρέτηση των πελατών τους, ειδικά όσον αφορά τις κρατήσεις. Στην περίπτωση του καταλύματος που ασχολείται η συγκεκριμένη εργασία ("Studio Liotrivi"), οι κρατήσεις πραγματοποιούνταν κυρίως μέσω τηλεφωνικής επικοινωνίας ή μέσω email, χωρίς να υπάρχει ένα αυτοματοποιημένο σύστημα διαχείρισης. Τα μειονεκτήματα από αυτή τη διαδικασία είναι πολλά και σημαντικά. Μερικά από αυτά είναι οι καθυστερήσεις στην απόκριση, η αδυναμία άμεσης ενημέρωσης για τη διαθεσιμότητα των δωματίων, η μεγάλη πιθανότητα να γίνουν λάθη και διπλοκρατήσεις, αλλά και η δυσκολία που έχει ο ιδιοκτήτης όσον αφορά την οργάνωση και στην αρχειοθέτηση των κρατήσεων.

Το συγκεκριμένο κεφάλαιο αναλύει τον τρόπο με τον οποίο προσδιορίστηκαν οι ανάγκες που έχουν οι χρήστες, οι απαιτήσεις του συστήματος και τα σενάρια για τη χρήση της εφαρμογής. Μέσα από την καταγραφή λειτουργικών και μη λειτουργικών απαιτήσεων καθώς και των περιπτώσεων χρήσης, καθορίζεται η λειτουργικότητα και ο τρόπος με τον οποίο αλληλεπιδρούν οι χρήστες με την πλατφόρμα.

Το ότι δεν υπάρχει ένα ψηφιακό σύστημα κρατήσεων δημιουργεί προβλήματα στην σωστή λειτουργία του καταλύματος, μερικά από τα οποία είναι η δημιουργία επιπλέον φόρτου εργασίας για τον ιδιοκτήτη ή τον υπεύθυνο κρατήσεων και δεν δίνει μια καλή εμπειρία στον χρήστη. Ο λόγος είναι ότι δεν δινόταν η δυνατότητα στον χρήστη να δει άμεσα τη διαθεσιμότητα των δωματίων, να επιλέξει ημερομηνίες ή να ολοκληρώσει μόνος του μια κράτηση σε πραγματικό χρόνο. Επίσης, δεν υπάρχει η δυνατότητα για συλλογή και ανάλυση δεδομένων, όπως για παράδειγμα η παρακολούθηση της πληρότητας, το ποσοστό ζήτησης ανάλογα με την εποχή, ή η δημιουργία στατιστικών στοιχείων που θα μπορούσαν να βοηθήσουν στη λήψη αποφάσεων για την βελτίωση της επιχείρησης.

Μέσα σε αυτό το πλαίσιο, προέκυψε η ανάγκη για ανάπτυξη μιας διαδικτυακής εφαρμογής κρατήσεων, η οποία θα προσφέρει στους χρήστες ένα εύχρηστο, εύκολα προσβάσιμο και λειτουργικό περιβάλλον για την αναζήτηση και την κράτηση δωματίων. Παράλληλα, θα δίνεται η δυνατότητα στον ιδιοκτήτη του καταλύματος να κάνει διαχείριση των κρατήσεων του καταλύματος με ευκολία, να παρακολουθεί τη διαθεσιμότητα, να δέχεται νέες αιτήσεις σε πραγματικό χρόνο και να κερδίζει χρόνο από διαδικασίες που είναι ίδιες για κάθε κράτηση. Η διαδικασία δημιουργίας μιας ιστοσελίδας κρατήσεων που μεταφέρει την επιχείρηση από την παραδοσιακή σε σύγχρονη ψηφιακή προσέγγιση, συμμετέχει σημαντικά στον εκσυγχρονισμό της επιχείρησης και την κάνει πιο ανταγωνιστική, ιδιαίτερα σε μια αγορά όπου οι περισσότεροι χρήστες πλέον αναζητούν και ολοκληρώνουν τις κρατήσεις τους αποκλειστικά μέσω διαδικτύου.

Η δημιουργία μιας σύγχρονης web εφαρμογής έρχεται λοιπόν να καλύψει αυτό το κενό, δίνοντας τη δυνατότητα στην επιχείρηση να ανταποκρίνεται στις απαιτήσεις της εποχής και επικεντρώνεται τόσο στην ευκολία του πελάτη όσο και στην αποτελεσματικότητα της επιχείρησης [4], [6].

3.2 Απαιτήσεις χρηστών

Για να γίνει εύχρηστη η ιστοσελίδα online κρατήσεων για το “Studio Liotrivi”, είναι σημαντικό να είναι αντιληπτό ποιος θα την χρησιμοποιήσει και ποιες είναι οι λειτουργικές ανάγκες που πρέπει να καλυφθούν. Η αναγνώριση αυτών των απαιτήσεων βοηθά να αποφασιστεί το πώς θα γίνει η δημιουργία του συστήματος, το πως θα επιλεγούν τα σωστά εργαλεία και το πως θα γίνει η ιστοσελίδα εύκολη στη χρήση της. Η εφαρμογή έχει δύο τύπους χρηστών με διαφορετικό ρόλο ο καθένας: τους επισκέπτες, δηλαδή τα άτομα που τη χρησιμοποιούν χωρίς λογαριασμό και τους διαχειριστές, οι οποίοι είναι υπεύθυνοι για τη διαχείριση και τον έλεγχο της εφαρμογής.

➤ **Επισκέπτης (Χρήστης χωρίς λογαριασμό)**

Ο επισκέπτης είναι το κύριο άτομο που χρησιμοποιεί την εφαρμογή. Είναι ο πελάτης που θέλει να κάνει αναζήτηση και κράτηση για ένα ή περισσότερα δωμάτια. Για να αποφευχθεί η δημιουργία λογαριασμού, η εμπειρία του χρήστη πρέπει να είναι όσο το δυνατόν πιο απλή, ώστε να γίνεται πιο εύκολα και γρήγορα η ολοκλήρωσης της κράτησης. Οι βασικές απαιτήσεις αυτού του ρόλου περιλαμβάνουν:

- **Προβολή διαθέσιμων δωματίων**: Μέσω αυτής της λειτουργίας ο πελάτης έχει την δυνατότητα να δει αναλυτικά τα δωμάτια που είναι διαθέσιμα για κράτηση, με όλες τις απαραίτητες πληροφορίες (τιμή ανά διανυκτέρευση, χωρητικότητα, τύπος δωματίου, περιγραφή δωματίου, καθώς και φωτογραφίες που απεικονίζουν τα δωμάτια και τον περιβάλλοντα χώρο.)
- **Αναζήτηση και φιλτράρισμα**: Το σύστημα προσφέρει τη δυνατότητα φιλτραρίσματος των δωματίων με βάση κάποιες συγκεκριμένες παραμέτρους, όπως είναι οι επιλογή ημερομηνιών check-in/check-out και ο αριθμός των ατόμων. Η χρήση του φίλτρου βοηθάει στο να προσδιοριστεί η διαθεσιμότητα και προβάλλει στον χρήστη μόνο τα δωμάτια που είναι διαθέσιμα με βάση τα κριτήριά του.
- **Υποβολή αίτησης κράτησης**: Ο επισκέπτης, για να κάνει κράτηση, πρέπει να συμπληρώσει μία σύντομη φόρμα, στην οποία δηλώνει τα προσωπικά του στοιχεία (π.χ. όνομα, email), τις ημερομηνίες διαμονής και το δωμάτιο στο οποίο θέλει να μείνει. Με την υποβολή αυτής της φόρμας δεν ολοκληρώνεται η κράτηση αυτόματα, αλλά γίνεται εμφανής με σχετικό αίτημα στον διαχειριστή για επιβεβαίωση ή απόρριψη.
- **Εναλλαγή γλώσσας περιβάλλοντος**: Λαμβάνοντας υπόψιν ότι οι υποψήφιοι πελάτες του καταλύματος μπορεί να προέρχονται από διαφορετικές χώρες, δίνεται η δυνατότητα εναλλαγής γλώσσας, τουλάχιστον μεταξύ ελληνικών και αγγλικών, ώστε να είναι πιο εύκολη η κατανόηση και η πλοήγηση για όλους.

Η διεπαφή του επισκέπτη πρέπει να είναι απλή, ξεκάθαρη, φιλική προς τον χρήστη και πλήρως responsive, ώστε να λειτουργεί εξίσου καλά σε κινητές συσκευές και υπολογιστές.

➤ **Διαχειριστής (Admin)**

Ο διαχειριστής της εφαρμογής είναι συνήθως ο ιδιοκτήτης ή το προσωπικό που έχει στην επίβλεψη του το κατάλυμα και του δίνεται η δυνατότητα να χρησιμοποιεί ένα σύνολο επεκταμένων λειτουργιών της εφαρμογής για να κάνει την διαχείριση. Σε αντιθέση με τον πελάτη, ο admin μπορεί να χρησιμοποιήσει τις λειτουργίες αυτές μόνο με πιστοποίηση (μέσω email και κωδικού πρόσβασης) και η χρήση του συστήματος γινεται για την καθημερινή λειτουργία και συντήρηση των δεδομένων της πλατφόρμας. Οι ανάγκες του περιλαμβάνουν:

- **Πλήρης προβολή κρατήσεων**: Ο διαχειριστής μπορεί να δει και να επεξεργαστεί όλες τις ενεργές και ολοκληρωμένες κρατήσεις. Τα δεδομένα αυτά εμφανίζονται οργανωμένα, με δυνατότητα ταξινόμησης ανά ημερομηνία, δωμάτιο ή πελάτη.
- **Έγκριση/απόρριψη κρατήσεων**: Όταν ένας απο τους επισκέπτες κάνει ένα αίτημα για μια κράτηση, αυτό αποθηκεύεται προσωρινά και εμφανίζεται στον διαχειριστή. Ο τελευταίος, αφού έχει συνδεθεί, έχει την υποχρέωση να εγκρίνει ή να απορρίψει την κράτηση, ανάλογα με την πληρότητα, τη χρονική περίοδο ή άλλες απαιτήσεις που μπορεί να έχει η επιχείρηση.
- **Επεξεργασία δωματίων**: Μέσα από τον πίνακα διαχείρισης, ο admin έχει τη δυνατότητα να προσθέτει νέα δωμάτια στην εφαρμογή. Η επεξεργασία υφιστάμενων δωματίων ή η προσωρινή απόκρυψή τους δεν υποστηρίζεται στην τρέχουσα έκδοση της εφαρμογής.

- **Ημερολογιακή προβολή**: Ο διαχειριστής έχει μέσω της ιστοσελίδας την δυνατότητα να δει τις κρατήσεις του μαζεμένες σε ένα ημερολόγιο κρατήσεων, όπου μπορεί να βλέπει συνοπτικά όλες τις κρατήσεις ανά δωμάτιο και ανά ημέρα. Αυτή το εργαλείο βοηθά σημαντικά στην δει ο admin συνολικά την πληρότητα και να φτιάξει ανάλογα τον προγραμματισμό της επιχείρησης.

Η διατήρηση της ασφάλειας των πληροφοριών, η διασφάλιση οτι η πρόσβαση γίνεται μόνο απο άτομα που έχουν την αυτή την δυνατότητα και το πόσο απλή είναι η χρήση της εφαρμογής αποτελούν βασικές προτεραιότητες για τη διαχειριστική διεπαφή της εφαρμογής, καθώς εξυπηρετεί καθημερινές ανάγκες λειτουργίας.

Συμπερασματικά, για να είναι επιτυχημένη η εφαρμογή πρέπει να γίνονται κατανοητοί οι ρόλοι και οι απαιτήσεις που έχουν οι χρήστες. Η διατήρηση μιας ξεκάθαρης διάκρισης μεταξύ επισκεπτών και των διαχειριστών βοηθάει να διατηρούνται όλες οι πληροφορίες ασφαλείς, ενώ εξασφαλίζεται μια ευχάριστη και αποτελεσματική εμπειρία για όλους τους εμπλεκόμενους.

3.3 Λειτουργικές και μη λειτουργικές απαιτήσεις

Η σωστή οργάνωση των απαιτήσεων του συστήματος αποτελεί προϋπόθεση για την επιτυχή ανάλυση, σχεδίαση και υλοποίηση οποιασδήποτε εφαρμογής. Οι απαιτήσεις της εφαρμογή χωρίζονται σε δύο βασικές κατηγορίες: τις λειτουργικές, που σκοπός τους είναι να δείξουν τι πρέπει να κάνει η εφαρμογή και τις μη λειτουργικές, που εξηγούν ποιους κανόνες και πρότυπα πρέπει να ακολουθεί το λογισμικό για να λειτουργεί καλά, να λειτουργεί σωστά και να μην έχει προβλήματα.

➢ Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν τη βασικές λειτουργίες της εφαρμογής και τις ενέργειες που πρέπει να είναι σε θέση να εκτελεί τόσο ο επισκέπτης όσο και ο διαχειριστής. Οι σημαντικότερες απο τις απαιτήσεις αυτές για την ιστοσελίδα που αναπτύχθηκε για την παρούσα πτυχιακή είναι οι εξής:

- **Κράτηση δωματίων με καθορισμένες ημερομηνίες**: Η εφαρμογή πρέπει να δίνει στον επισκέπτη τη δυνατότητα να επιλέξει ημερομηνίες check-in και check-out, να δηλώσει τον αριθμό των ατόμων και να υποβάλει αίτημα κράτησης για κάποιο από τα διαθέσιμα δωμάτια που υπάρχουν στην λίστα.
- **Προβολή μόνο διαθέσιμων δωματίων**: Κατά την αναζήτηση δωματίων, το σύστημα πρέπει να ελέγχει το ποια είναι διαθέσιμα στη βάση δεδομένων και να εμφανίζει στον χρήστη μόνο εκείνα που δεν έχουν κρατηθεί για τις συγκεκριμένες ημερομηνίες που έχει επιλέξει.
- **Διαχείριση κρατήσεων από τον admin**: Ο διαχειριστής πρέπει να έχει πρόσβαση στην λίστα κρατήσεων και να του δίνεται η δυνατότητα έγκρισης ή απόρριψης ανάλογα με τη διαθεσιμότητα ή κάποια άλλα κριτήρια. Η αλλαγή της κατάστασης μιας κράτησης, που κάνει ο διαχειριστής, πρέπει να ενημερώνει δυναμικά το σύστημα [9], [12].
- **Αποστολή email επικοινωνίας**: Η εφαρμογή πρέπει να επιτρέπει στους επισκέπτες να μπορούν να επικοινωνήσουν με τον διαχειριστή μέσω ειδικής φόρμας. Το μήνυμα αυτό πρέπει να αποστέλλεται μέσω email στον διαχειριστή της εφαρμογής του καταλύματος, περιλαμβάνοντας τα στοιχεία του αποστολέα και το περιεχόμενο της ερώτησης.

➤ **Μη Λειτουργικές Απαιτήσεις**

Οι μη λειτουργικές απαιτήσεις αφορούν τα τεχνικά χαρακτηριστικά και τους περιορισμούς του συστήματος, οι οποίοι δεν έχουν άμεση σχέση με τη λογική λειτουργία αλλά επηρεάζουν την εμπειρία χρήσης, το πόσο εύκολη είναι η συντήρηση του και την απόδοσή του.

- **Χρόνος απόκρισης**: Η εφαρμογή πρέπει να πραγματοποιεί τις λειτουργίες τού για κάθε αίτημα χρήστη (αναζήτηση, υποβολή φόρμας, προβολή δεδομένων) σε χρόνο μικρότερο των 2 δευτερολέπτων, με σκοπό να διασφαλίζεται η ταχύτητα και η ευκολία χρήσης.
- **Πολυγλωσσική υποστήριξη**: Για την κάλυψη επισκεπτών που είναι απο το εξωτερικό, η διεπαφή του χρήστη πρέπει να δίνεται τουλάχιστον σε δύο γλώσσες, στην προκειμένη περίπτωση τα ελληνικά και τα αγγλικά. Η εναλλαγή γλώσσας πρέπει να γίνεται εύκολα και να επηρεάζει όλη την πληροφορία που μπορεί να δει ο χρήστης.
- **Υποστήριξη πολλαπλών συσκευών**: Η εφαρμογή οφείλει να είναι fully responsive, δηλαδή να μπορεί να προσαρμόζεται η εμφάνισή της αυτόματα σε διαφορετικές αναλύσεις και συσκευές, όπως desktop, tablet και κινητά τηλέφωνα, χωρίς να χάνεται η λειτουργικότητα ή κάτι απο το περιεχόμενο.
- **Διαθεσιμότητα του πηγαίου κώδικα (open source)**: Η ανάπτυξη της εφαρμογής, που δημιουργήθηκε για τις ανάγκες της πτυχιακής, βασίζεται σε αρχές ανοικτού λογισμικού. Ο πηγαίος κώδικας δημοσιεύεται δημόσια στο GitHub υπό την άδεια MIT, επιτρέποντας σε άλλους προγραμματιστές να μελετήσουν, τροποποιήσουν ή επαναχρησιμοποιήσουν τον κώδικα με ελάχιστους περιορισμούς [16], [7].

Η κατανομή των απαιτήσεων σε διαφορετικές ομάδες βοηθά το έργο να παραμείνει οργανωμένο και σε καλό δρόμο. Δίνει επίσης ένα σαφές σχέδιο για τον έλεγχο του πόσο καλά λειτουργεί το τελικό προϊόν. Στον Πίνακα 3.1 παρουσιάζονται συνοπτικά οι απαιτήσεις της εφαρμογής και ο διαχωρισμός τους σε λειτουργικές ή μη.

Πίνακας 3.1: Λειτουργικές και Μη Λειτουργικές Απαιτήσεις της Εφαρμογής

Κατηγορία	Ρόλος	Περιγραφή Απαίτησης
Λειτουργική	Επισκέπτης	Αναζήτηση διαθέσιμων δωματίων με βάση ημερομηνίες και αριθμό ατόμων
	Επισκέπτης	Υποβολή αίτησης κράτησης
	Επισκέπτης	Αλλαγή γλώσσας διεπαφής (GR/EN)
	Επισκέπτης	Αποστολή μηνύματος επικοινωνίας μέσω φόρμας
	Διαχειριστής	Επισκόπηση όλων των αιτήσεων κράτησης
	Διαχειριστής	Έγκριση ή απόρριψη αιτήσεων
	Διαχειριστής	Διαχείριση δωματίων (τίτλος, τύπος, τιμή, εικόνες, χωρητικότητα)
	Διαχειριστής	Προβολή ημερολογίου κρατήσεων ανά δωμάτιο
Μη Λειτουργική	Σύστημα	Απόκριση σε όλα τα αιτήματα σε λιγότερο από 2 δευτερόλεπτα
	Σύστημα	Υποστήριξη γλωσσών: Ελληνικά και Αγγλικά

Κατηγορία	Ρόλος	Περιγραφή Απαίτησης
	Σύστημα	Συμβατότητα με κινητές συσκευές και υπολογιστές (responsive σχεδίαση)
	Σύστημα	Δημοσίευση κώδικα στο GitHub με άδεια χρήσης MIT (open source)

3.4 Περιπτώσεις Χρήσης (Use Cases)

Η περιγραφή των περιπτώσεων χρήσης (Use Cases) είναι ένας τρόπος για να φανεί τι μπορεί να κάνουν διαφορετικοί άνθρωποι με ένα πρόγραμμα υπολογιστή, καθώς επίσης κάνει κατανοητό το πως χρησιμοποιεί ο καθένας το σύστημα και ποιες ενέργειες μπορεί να κάνει. Η συγκεκριμένη εφαρμογή κράτησης δωματίων υποστηρίζει, όπως έχει ήδη αναφερθεί παραπάνω, δύο βασικούς ρόλους χρηστών: τον επισκέπτη (χωρίς λογαριασμό) και τον διαχειριστή (administrator). Ο σκοπός των Use Cases είναι να καθοριστεί όσο πιο καλά γίνεται η συμπεριφορά της εφαρμογής σε διάφορα σενάρια αλληλεπίδρασης, ώστε να μπορούν να χρησιμοποιηθούν ως παράδειγμα για την υλοποίηση, τη δοκιμή και τη μελλοντική επέκταση του συστήματος.

➤ **Επισκέπτης (Guest)**

Ο επισκέπτης της ιστοσελίδας, ο οποίος δεν χρειάζεται να έχει λογαριασμό, έχει στη διάθεσή του τις παρακάτω περιπτώσεις χρήσης:

- **UC1: Αναζήτηση διαθέσιμων δωματίων**

  Εισάγει τις ημερομηνίες check-in και check-out που επιθυμεί, καθώς και τον αριθμό ατόμων. Το σύστημα κάνει έλεγχο τη βάση δεδομένων και δείχνει μόνο τα δωμάτια που είναι διαθέσιμα για τη συγκεκριμένη περίοδο.

- **UC2: Προβολή πληροφοριών δωματίου**

  Ο επισκέπτης μπορεί να δει αναλυτικές πληροφορίες για κάθε δωμάτιο, όπως είναι η περιγραφή, η τιμή, η χωρητικότητα και φωτογραφίες.

- **UC3: Υποβολή αίτησης κράτησης**

  Από τη σελίδα ενός διαθέσιμου δωματίου, ο επισκέπτης μπορεί να συμπληρώσει μια φόρμα κράτησης (όνομα, email, σχόλια) και να στείλει αίτημα. Το αίτημα αποθηκεύεται ως "σε εκκρεμότητα" και ενημερώνεται ο διαχειριστής ώστε να εγκρίνει ή να απορρίψει το αίτημα.

- **UC4: Αλλαγή γλώσσας περιβάλλοντος**

  Η διεπαφή προσφέρει τη δυνατότητα αλλαγής γλώσσας μεταξύ Ελληνικών και Αγγλικών. Όλα τα στοιχεία του περιβάλλοντος προσαρμόζονται δυναμικά ανάλογα με την επιλογή.

- **UC5: Αποστολή μηνύματος επικοινωνίας**

  Ο επισκέπτης μπορεί να στείλει μήνυμα μέσω φόρμας επικοινωνίας. Το μήνυμα αποστέλλεται με email στον διαχειριστή για παραπάνω επικοινωνία.

➤ **Διαχειριστής (Admin)**

Ο διαχειριστής της εφαρμογής είναι υπεύθυνος για τη σωστή διαχείριση του καταλύματος και έχει πρόσβαση σε περισσότερες λειτουργίες μέσω ασφαλούς εισόδου:

- **UC6: Προβολή κρατήσεων**

Μπορεί να δει συγκεντρωτικά όλες τις αιτήσεις κράτησης, μαζί με τα στοιχεία των πελατών, τις ημερομηνίες και την κατάσταση της κράτησης.

- **UC7: Έγκριση/Απόρριψη κρατήσεων**

  Για κάθε κράτηση, πρέπει ο admin να κάνει αποδοχή ή απόρριψη. Η ενέργεια αυτή ενημερώνει τη βάση και ενεργοποιεί αποστολή ενημερωτικού email στον επισκέπτη.

- **UC8: Επεξεργασία δωματίων**

  Δεν μπορεί να επεξεργαστεί τα δεδομένα των δωματίων, δηλαδή το όνομα, την τιμή, τον τύπο και την χωρητικότητα ούτε μπορεί να προσθέσει ή να αφαιρέσει εικόνες σε αυτή την έκδοση της εφαρμογής.

- **UC9: Προβολή ημερολογίου κρατήσεων**

  Μέσω ενός ημερολογίου που περιέχει όλες τις κρατήσει, ο διαχειριστής μπορεί να δει ποιες ημέρες είναι κλεισμένες ανά δωμάτιο, διευκολύνοντας τον προγραμματισμό και την οργάνωση της διαχείρισης.

4 Σχεδιασμός Συστήματος

4.1 Αρχιτεκτονική εφαρμογής

Η εφαρμογή «Studio Liotrivi», που δημιουργήθηκε στο πλαίσιο της συγκεκριμένης εργασίας, ακολουθεί μια αρχιτεκτονική τύπου Client-Server, η οποία είναι απλή στην κατανόηση και μπορεί να επεκταθεί εύκολα. Στην αρχιτεκτονική αυτή διαχωρίζονται με σαφήνεια το frontend (επισκέπτης) και το backend (διακομιστής), καθώς "εκμεταλλεύεται" τεχνολογίες που θεωρούνται σύγχρονες τόσο σε επίπεδο παρουσίασης όσο και σε επίπεδο διαχείρισης δεδομένων και ασφάλειας. Σκοπός της επιλογής να χρησιμοποιηθεί η συγκεκριμένη αρχιτεκτονική ήταν να είναι εύκολη στη συντήρηση, να μπορεί να επεκταθεί και να μπορεί να συνεργάζεται με άλλες πλατφόρμες, ενώ παράλληλα με τα παραπάνω να είναι γρήγορη στην λειτουργία και εύκολη στην χρήση από τον τελικό χρήστη.

4.1.1 Τεχνολογική Στοίβα

Η τεχνολογική στοίβα της εφαρμογής αποτελείται από μια σειρά εργαλείων ανοιχτού κώδικα τα οποία προσφέρουν σταθερότητα και υποστήριξη μακροπρόθεσμα:

Frontend

- **HTML5**: Αποτελεί τη βασική γλώσσα της εφαρμογής η οποία χρησιμοποιείται προκειμένου να παρουσιαστεί η δομή του περιεχομένου της εφαρμογής. Μέσω του HTML5 ορίζονται τα στοιχεία με τα οποία έχει διεπαφή ο χρήστης (όπως φόρμες, κουμπιά, πίνακες και επικεφαλίδες), προσφέροντας τη δημιουργία ενός σαφούς και κατανοητού layout. Επιπλέον, παρέχει σύγχρονες δυνατότητες, όπως είναι η ενσωμάτωση πολυμέσων που ενισχύουν το πόσο προσβάσιμη και λειτουργική είναι η εφαρμογή [1], [2].
- **CSS3**: Γίνεται χρήση αυτής της τεχνολογίας για να βελτιωθεί η μορφοποίηση και η αισθητική παρουσίαση του περιβάλλοντος χρήστη. Υποστηρίζει responsive σχεδίαση για να μπορεί να προσαρμοστεί σε διαφορετικές αναλύσεις και συσκευές (desktop, tablet, mobile), προσφέροντας έτσι φιλική εμπειρία προς τον χρήστη.
- **JavaScript (vanilla):** Η χρήση της είναι απαραίτητη για τον δυναμικό χειρισμό στοιχείων της διεπαφής χρήστη, όπως για παράδειγμα το να μπορεί να υποβληθεί η φόρμα κράτησης, να γίνει προβολή διαθέσιμων δωματίων με βάση κάποια φίλτρα (ημερομηνίες, αριθμός ατόμων), ή να εναλλάσσονται εικόνες μέσα στη συλλογή εικόνων με slideshow. Επίσης είναι απαραίτητη και για την εναλλαγή γλώσσας (i18n) [1].
- **Bootstrap 5.3**: Ενσωματώνεται μέσω CDN (δηλαδή μέσω του διαδικτύου) στο index.html για την παροχή έτοιμων στοιχείων διεπαφής χρήστη (UI components) και ενός responsive grid συστήματος. Βοηθά στο να δημιουργούνται προσαρμόσιμα και καλαίσθητα layout, περιορίζοντας την χρήση του σύνθετου CSS [20].

Backend

- **Node.js**: Χρησιμοποιείται ως runtime περιβάλλον (δηλαδή το λογισμικό που εκτελεί ένα προγραμμα) στην πλευρά του διακομιστή, δίνοντας έτσι την δυνατότητα για εκτέλεση του JavaScript εκτός του προγράμματος περιήγησης. Κάνει πιο εύκολο τον χειρισμό των HTTP αιτημάτων και την επικοινωνία με τη βάση δεδομένων [3], [21].
- **Express.js**: Είναι ένα ελαφρύ και βοηθητικό εργαλείο λογισμικού (framework), το οποίο προσαρμόζεται εύκολα, για το Node.js. Μέσω αυτού δίνεται η δυνατότητα να δημιουργηθούν σημεία πρόσβασης (endpoints) στο API. Επίσης, καθορίζει τι θα γίνει με την δρομολόγηση (routing), διαχειρίζεται τις ενδιάμεσες λειτουργίες (middleware) και πραγματοποιείται η σύνδεση με τη βάση δεδομένων [5], [22].
- **JWT (jsonwebtoken)**: Χρησιμοποιείται για αυθεντικοποίηση των χρηστών. Τα tokens αποθηκεύονται και επιβεβαιώνονται από τον server [16], [14].

- **bcryptjs**: Βιβλιοθήκη για τον κρυπτογραφικό κατακερματισμό (hashing) των κωδικών πρόσβασης. Οι κωδικοί αποθηκεύονται σε μορφή hash στη βάση δεδομένων, ώστε να μην μπορούν να αναγνωστούν σε περίπτωση διαρροής. Έτσι, ενισχύεται η ασφάλεια του συστήματος [12], [13], [23] .
- **Nodemailer**: Χρησιμοποιείται για την αποστολή email επικοινωνίας μέσα από την ιστοσελίδα, όπως φαίνεται στο endpoint /contact [24].

Database

- **PostgreSQL**: Ισχυρό σύστημα σχεσιακής βάσης δεδομένων, κατάλληλο για πολύπλοκες εφαρμογές. Υποστηρίζει πολλαπλές οντότητες, όπως χρήστες (users), κρατήσεις (bookings), δωμάτια (rooms) και εικόνες (room_images) [3], [7], [8], [25].
- Η επικοινωνία με τη βάση δεδομένων γίνεται μέσω της βιβλιοθήκης driver pg, μέσω της χρήσης ερωτημάτων (queries) που εκτελούνται ασύγχρονα από τον Express server με τη χρήση async/await [9].

Λοιπά

- **CORS**: Ενεργοποιείται με την εντολή app.use(cors()) για να επιτρέπει αιτήματα από διαφορετικά origins, το οποίο είναι απαραίτητο για ανάπτυξη του frontend-backend που τρέχουν σε ξεχωριστές διευθύνσεις [3], [6].

4.1.2 Αρχιτεκτονικό Μοντέλο – Client/Server

Η εφαρμογή αποτελείται μια τριεπίπεδη αρχιτεκτονική (Three-Tier Architecture) [10] , διαχωρισμένη σε:

1. Παρουσίαση (Presentation Layer – Frontend)

Το frontend τρέχει στον browser του χρήστη και είναι υπεύθυνο για:

- την εμφάνιση δωματίων (μέσω της HTML/CSS)
- τη φόρμα κράτησης (με JavaScript χειρισμό)
- τη λειτουργία εναλλαγής γλώσσας (διεθνοποίηση)
- την αποστολή δεδομένων με AJAX (fetch προς backend API)
- την προβολή εικόνων μέσω gallery/lightbox.

Ο χρήστης αλληλεπιδρά με κουμπιά, φόρμες και λίστες, π.χ. επιλογή ημερομηνιών → fetch → προβολή διαθέσιμων δωματίων.

2. Λογική Εφαρμογής (Business Logic Layer – Backend)

Ο server είναι υπεύθυνος για:

- την επεξεργασία των αιτημάτων (GET /rooms, POST /bookings, POST /login)
- την αυθεντικοποίηση με JWT tokens
- τον έλεγχο των credentials των χρηστών μέσω bcrypt
- την αποστολή email επικοινωνίας μέσω SMTP (nodemailer)
- την απόκριση με JSON προς το frontend.

Ο βασικός server (server.js) περιλαμβάνει routes για:

- /rooms – λίστα δωματίων
- /rooms/:id/images – εικόνες συγκεκριμένου δωματίου

- /login και /register – σύνδεση / εγγραφή
- /bookings – διαχείριση κρατήσεων
- /contact – αποστολή email από φόρμα επικοινωνίας

3. Επίπεδο Δεδομένων (Data Layer – PostgreSQL)

Η βάση PostgreSQL περιλαμβάνει πίνακες όπως:

- users → email, hashed password, role
- rooms → όνομα, τιμή, τύπος, χωρητικότητα, περιγραφή
- bookings → ημερομηνίες, στοιχεία πελάτη, status
- room_images → image URLs ανά δωμάτιο

Όλα τα δεδομένα φυλάσσονται και ανακτώνται μέσω SQL queries από τον Express server.

4.1.3 Λειτουργική Ροή (Request Flow)

Η επικοινωνία χρήστη–συστήματος γίνεται μέσω HTTP αιτημάτων και JSON αποκρίσεων.

Παράδειγμα Ροής Κράτησης:

1. Ο επισκέπτης επιλέγει ημερομηνίες check-in / check-out και αριθμό ατόμων στο frontend (HTML + JavaScript).
2. Η JavaScript στέλνει GET αίτημα σε /bookings/available-rooms με query parameters.
3. Ο server επεξεργάζεται το αίτημα, φιλτράρει τα δωμάτια στη βάση δεδομένων και επιστρέφει/προτείνει μόνο τα διαθέσιμα σε μορφή JSON.
4. Η JavaScript λαμβάνει την απάντηση και γεμίζει δυναμικά το rooms-container με τα διαθέσιμα δωμάτια.
5. Ο χρήστης επιλέγει ένα δωμάτιο, συμπληρώνει φόρμα κράτησης και υποβάλλει με POST στο /bookings.
6. Η κράτηση αποθηκεύεται στη βάση με status: pending (σε εκκρεμότητα), μέχρι ο διαχειριστής να την εγκρίνει.

Παράδειγμα Επικοινωνίας:

- Ο χρήστης γράφει μήνυμα στη φόρμα επικοινωνίας → POST /contact
- Ο server χρησιμοποιεί nodemailer για να αποστείλει email στον διαχειριστή.
- Επιστρέφει success: true αν το email αποσταλεί επιτυχώς.

4.1.4 Ασφάλεια & Βελτιστοποίηση

Για να ενισχυθεί η ασφάλεια και να γίνει πιο αξιόπιστο το σύστημα, εφαρμόστηκαν διάφορες τεχνικές. Για αρχή, οι κωδικοί πρόσβασης των χρηστών δεν αποθηκεύονται ποτέ απευθείας, όπως έχει ήδη προαναφερθεί. Πρώτα γίνεται η μετατροπή τους σε hashed μορφή με χρήση της βιβλιοθήκης bcrypt, παρέχοντας έτσι περισσότερη προστασία ακόμη και σε καταστάσεις που έχει παραβιαστεί η βάση δεδομένων. Επιπλέον, κατά τη διαδικασία της ταυτοποίησης, οι χρήστες παίρνουν ένα JSON Web Tokens (JWT) που ενσωματώνουν το userId και το role. Αυτό έχει ισχύ για μία ώρα, και έτσι εξασφαλίζεται η προσωρινή και ασφαλής πρόσβαση σε προστατευόμενες λειτουργίες. Για τη διασύνδεση μεταξύ του frontend και του backend, το σύστημα ενεργοποιεί την πολιτική Cross-Origin Resource Sharing (CORS), επιτρέποντας νόμιμα αιτήματα από διαφορετικές πηγές. Αυτό είναι ιδιαίτερα χρήσιμο κατά την ανάπτυξη ή την απομακρυσμένη φιλοξενία των επιμέρους μερών της

εφαρμογής. Η διαχείριση σφαλμάτων γίνεται οργανωμένα μέσω δομής try/catch σε κάθε route, ώστε σε περίπτωση προβλήματος να εμφανίζεται στον χρήστη ένα κατάλληλο μήνυμα και ο κωδικός κατάστασης 500. Με τον τρόπο αυτό βελτιώνεται η ανθεκτικότητα της εφαρμογής. Τέλος, ιδιαίτερη έμφαση δόθηκε στο να γίνει βελτιστοποίηση των αποκρίσεων, με τις βάσεις δεδομένων να αξιοποιούν μηχανισμούς connection pooling και ασύγχρονες κλήσεις, καταφέρνοντας να πετύχουν χρόνους απόκρισης κάτω των 2 δευτερολέπτων ακόμη και σε απαιτητικές λειτουργίες.

4.1.5 Πλεονεκτήματα της Αρχιτεκτονικής

Η αρχιτεκτονική της εφαρμογής προσφέρει πολλά πλεονεκτήματα που την κάνουν να είναι ευέλικτη και σύγχρονη. Αρχικά, δίνει τη δυνατότητα για πολύ μεγάλη επεκτασιμότητα, καθώς δίνει την δυνατότητα να προστεθεί ένα νέο frontend, όπως για παράδειγμα μια mobile εφαρμογή, αξιοποιώντας τα ίδια API endpoints. Η επαναχρησιμοποίηση αυτών των endpoints βοηθά στην εξυπηρέτηση διαφορετικών περιβαλλόντων με ενιαίο backend. Παράλληλα, η ασφάλεια αυξάνεται σε μεγάλο βαθμό μέσω της χρήσης tokens JWT για έλεγχο ταυτότητας και της τεχνικής hashing προκειμένου να αποθηκευτούν οι κωδικοί πρόσβασης. Με τον τρόπο αυτό περιορίζονται κοινά προβλήματα όπως το session hijacking και η αποθήκευση plaintext κωδικών. Η προσβασιμότητα της εφαρμογής είναι επίσης πολύ μεγάλη, καθώς η εφαρμογή είναι διαθέσιμη σε πολλαπλές συσκευές και γλώσσες. Τέλος, το γεγονός ότι πρόκειται για open source έργο, με τον πηγαίο κώδικα διαθέσιμο στο GitHub και την εφαρμογή να δημοσιεύεται με την άδεια MIT, προσφέρει στους χρήστες και προγραμματιστές πλήρη ελευθερία τροποποίησης, μελέτης και επαναχρησιμοποίησης της εφαρμογής.

Η αρχιτεκτονική της εφαρμογής είναι ταυτόχρονα απλή και έχει ισχυρές τεχνολογίες, καθώς βασίζεται σε ήδη δοκιμασμένα πρότυπα ανάπτυξης διαδικτυακών εφαρμογών. Υπάρχει διαχωρισμός ανάμεσα σε frontend, backend και βάση δεδομένων, διαθέτει σταθερότητα, ευκολία στην συντήρηση και δίνει την δυνατότητα για μελλοντική επέκταση. Ο τεχνολογικός συνδυασμός Node.js + Express + PostgreSQL με υποστήριξη JWT, bcrypt και nodemailer κάνει την εφαρμογή λειτουργική, ασφαλή και προσαρμόσιμη στις ανάγκες ενός σύγχρονου καταλύματος.

4.2 Διάγραμμα ροής δεδομένων

Η κατανόηση της ροής δεδομένων μεταξύ των χρηστών και του συστήματος είναι κρίσιμη για τη σχεδίαση και τεκμηρίωση της εφαρμογής. Το Διάγραμμα Ροής Δεδομένων επιπέδου 0 (DFD Level 0) αποτελεί ένα από τα πιο βασικά και ταυτόχρονα χρήσιμα εργαλεία για την αποτύπωση των κύριων λειτουργιών και της διακίνησης πληροφοριών εντός ενός πληροφοριακού συστήματος. Το παρόν διάγραμμα περιγράφει το πώς οι χρήστες αλληλεπιδρούν με το σύστημα κρατήσεων και πώς τα δεδομένα καταλήγουν ή πως προέρχονται από τη σχεσιακή βάση δεδομένων (PostgreSQL) [26].

4.2.1 Οντότητες του DFD

Το Διάγραμμα Ροής Δεδομένων (Εικ. 4.1) αποτελείται απο τέσσερις βασικές οντότητες, καθεμία απο τις οποίες καθεμία έχει διαφορετικό ρόλο στο πλαίσιο της λειτουργίας της εφαρμογής. Ο Επισκέπτης (Visitor) είναι ο τελικός χρήστης της εφαρμογής, ο οποίος πραγματοποιεί αναζητήσεις και υποβάλλει αιτήματα κρατήσεων για τα διαθέσιμα δωμάτια. Από την άλλη πλευρά, ο Διαχειριστής (Admin), π.χ. ο ιδιοκτήτης του καταλύματος, έχει πρόσβαση σε περισσότερες λειτουργίες ώστε να μπορεί να διαχειρίζεται την πλατφόρμα, όπως είναι για παράδειγμα η έγκριση ή η απόρριψη κρατήσεων, η παρακολούθηση του ημερολογίου και η επεξεργασία των δεδομένων των δωματίων. Το Σύστημα Κρατήσεων αναφέρεται στο backend μέρος της εφαρμογής, το οποίο έχει υλοποιηθεί με χρήση Node.js και Express. Το σύστημα αυτό διαχειρίζεται όλα τα αιτήματα των χρηστών, επεξεργάζεται τις σχετικές πληροφορίες και λειτουργεί με τέτοιο τρόπο ώστε να συνδέονται οι χρήστες και η βάση δεδομένων. Τέλος, η Βάση Δεδομένων (PostgreSQL) είναι το αποθετήριο όλων των βασικών και κρίσιμων πληροφοριών της εφαρμογής, περιλαμβάνοντας δεδομένα που σχετίζονται με τους χρήστες, με τα δωμάτια, με τις κρατήσεις, με τις εικόνες αλλά και με τις παραμετροποιήσεις του συστήματος.

4.2.2 Περιγραφή Ροών Δεδομένων

Στο πλαίσιο της λειτουργίας της εφαρμογής, οι κύριες ενέργειες διακρίνονται σε ενέργειες επισκέπτη και διαχειριστή, με κάθε ρόλο να έχει τις δικές του αρμοδιότητες και αυτοί συνεργάζονται με το σύστημα με συγκεκριμένο τρόπο. Ο επισκέπτης πληκτρολογεί τις απλές λεπτομέρειες σχετικά με την κράτησή του στην αρχική σελίδα της εφαρμογής, όπως την ημερομηνία άφιξης (check-in), την ημερομηνία αναχώρησης (check-out) και τον αριθμό των ατόμων για τα οποία ενδιαφέρεται να κάνει κράτηση. Τα δεδομένα αυτά αποστέλλονται στο Σύστημα Κρατήσεων, το οποίο επεξεργάζεται το αίτημα και επιστρέφει μια λίστα με δωμάτια από τα οποία ο επισκέπτης έχει τη δυνατότητα να επιλέξει με βάση αυτό που θέλει ο ίδιος. Εφόσον ο χρήστης επιλέξει να προχωρήσει σε κράτηση, του ζητείται να γράψει το όνομά του και τη διεύθυνση email που διαθέτει και στη συνέχεια αυτές οι πληροφορίες αποθηκεύονται στην βάση ως μια νέα εγγραφή. Η εφαρμογή εμφανίζει ένα μήνυμα μέσω του οποίου ενημερώνεται άμεσα ο επισκέπτης σχετικά με την επιτυχία ή την αποτυχία του αιτήματος που υπέβαλε. Για παράδειγμα σε περιπτώσεις στις οποίες δεν υπάρχει διαθεσιμότητα, το αίτημα ια κράτηση κρίνεται ανεπιτυχές.

Από την άλλη πλευρά, ο διαχειριστής εισέρχεται στην εφαρμογή αφού ελεγχθεί ότι είναι αυτός μέσω token JWT και έτσι έχει πρόσβαση σε κάποιες διαφορετικές σελίδες της εφαρμογής, όπως ο Πίνακας Κρατήσεων και το Ημερολόγιο Δωματίων. Από εκεί, μπορεί να εκτελέσει μια σειρά λειτουργιών, όπως η προβολή όλων των κρατήσεων (είτε αυτών που εκκρεμούν είτε αυτών που έχουν εγκριθεί), να εγκρίνει ή να απορρίψει νέα αιτήματα, να τροποποιεί ή να διαγράφει υπάρχουσες κρατήσεις, καθώς και να ενημερώσει το ημερολόγιο με το να προσθέσει δωμάτια με τα αντίστοιχα στοιχεία, εικόνες και τιμές ή να αφαιρέσει αντίστοιχα. Όλες οι παραπάνω ενέργειες μετατρέπονται σε HTTP αιτήματα προς το backend της εφαρμογής, το οποίο με τη σειρά του επικοινωνεί με τη βάση δεδομένων για την αντίστοιχη ενημέρωση.

Ο ρόλος της βάσης δεδομένων, η οποία έχει υλοποιηθεί με PostgreSQL, είναι κρίσιμος για την λειτουργία της εφαρμογής, καθώς αποτελεί τον βασικό αποθηκευτικό μηχανισμό της. Εκεί βρίσκονται όλα τα δωμάτια που είναι διαθέσιμα, οι κρατήσεις που έχουν γίνει από πελάτες έως τώρα, το ποιοι είναι οι χρήστες και οι δυνατότητές τους, καθώς και οι φωτογραφίες των δωματίων. Κάθε φορά που ο επισκέπτης ή ο διαχειριστής κάνει κάτι καινούργιο, το σύστημα εκτελεί ειδικές εντολές που ονομάζονται ερωτήματα SQL. Αυτές οι εντολές διασφαλίζουν ότι τα δεδομένα (πληροφορίες) είναι σωστά, πλήρη και ενημερωμένα.

4.2.3 Οπτική Αναπαράσταση – Διάγραμμα Επιπέδου 0

Η οπτική αναπαράσταση της ροής δεδομένων στην εφαρμογή αποτυπώνεται μέσω του Διαγράμματος Ροής Δεδομένων (Data Flow Diagram - DFD) επιπέδου 0 (σχ. 4.1). Αυτή η εικόνα δείχνει πώς τα διάφορα μέρη του συστήματος συνομιλούν μεταξύ τους και περιλαμβάνει τον επισκέπτη, τον διαχειριστή, το backend και τη βάση δεδομένων. Μέσω αυτής της απλοποιημένης αναπαράστασης, γίνεται κατανοητός ο τρόπος με τον οποίο τα δεδομένα διακινούνται στο σύστημα, από την αρχική εισαγωγή τους μέχρι την αποθήκευσή τους.

Το μοντέλο DFD επιπέδου 0 αναδεικνύει τη βασική λειτουργική ροή της εφαρμογής. Τονίζει τον κομβικό ρόλο του backend με το να λειτουργεί σαν βοηθός που συνδέει αυτά που εμφανίζονται στην οθόνη με το χώρο αποθήκευσης, όπου φυλάσσονται όλες οι πληροφορίες. Αυτή η ρύθμιση βοηθά στο να διατηρούνται οι πληροφορίες ξεκάθαρες και ασφαλείς. Έτσι εξασφαλίζεται το ότι μόνο τα "σωστά" άτομα μπορούν να δουν ή να κάνουν πράγματα μέσα στην εφαρμογή, διατηρώντας τα πάντα ασφαλή και εύκολα στη διαχείριση. Παράλληλα, κάνει εφικτή και εύκολη την επεκτασιμότητα της αρχιτεκτονικής, επιτρέποντας την προσθήκη νέων λειτουργιών ή τη βελτίωση ήδη υπάρχοντων χωρίς να απαιτείται ριζική αναδιάρθρωση του συστήματος [27].

Συνολικά, η χρήση του DFD ως εργαλείο μοντελοποίησης βοηθάει ουσιαστικά όχι μόνο στη φάση της σχεδίασης αλλά και στην τεκμηρίωση της εφαρμογής. Παράλληλα, είναι ένας σημαντικός "οδηγός" που βοηθά τους ανθρώπους να καταλάβουν τον τρόπο που λειτουργεί το σύστημα πληροφοριών.

Αυτό διευκολύνει τους προγραμματιστές, τους αναλυτές και άλλους χρήστες να το διατηρήσουν ομαλό ή να το βελτιώσουν στο μέλλον.

Σχήμα 4.1: DFD Level 0

4.3 Δομή και Σχέσεις της Βάσης Δεδομένων

Είναι πολύ σημαντικό το να υπάρχει ένα καλά μελετημένο σχέδιο για τον τρόπο οργάνωσης και διατήρησης των πληροφοριών καθώς αυτό βοηθά το σύστημα να λειτουργεί ομαλά, να παραμένει ισχυρό και να είναι πιο εύκολο να γίνουν διορθώσεις ή βελτιώσεις στο μέλλον. Στην περίπτωση της εφαρμογής “Studio Liotrivi”, η βάση δεδομένων δημιουργήθηκε με το ισχυρό και αξιόπιστο σύστημα διαχείρισης σχεσιακών βάσεων PostgreSQL. Η δομή σχεδιάστηκε με τέτοιο τρόπο ώστε να εξυπηρετεί τις ανάγκες όλων των εμπλεκόμενων οντοτήτων – του επισκέπτη, του διαχειριστή και του backend συστήματος – διασφαλίζοντας ότι οι πληροφορίες μεταφέρονται ομαλά και αποδοτικά.

Η βασική αρχιτεκτονική της βάσης δεδομένων στηρίζεται σε τέσσερις κύριους πίνακες: rooms, bookings, users και room_images. Κάθε πίνακας έχει έναν καλά καθορισμένο ρόλο και έχει σχεδιαστεί με σεβασμό στις αρχές της κανονικοποίησης (normalization), ώστε να διασφαλίζεται ότι δεν υπάρχει επανάληψη συμβάντων και ότι όλες οι πληροφορίες ταιριάζουν. Παρακάτω παρουσιάζεται η αναλυτική περιγραφή των πινάκων, οι μεταξύ τους σχέσεις και η αποτύπωσή τους στο αντίστοιχο Entity Relationship διάγραμμα (σχ. 4.1).

Στο επίκεντρο της δομής βρίσκεται ο πίνακας rooms, ο οποίος έχει όλες τις σημαντικές λεπτομέρειες για κάθε δωμάτιο του καταλύματος. Στον πίνακα καταγράφονται τα χαρακτηριστικά του κάθε δωματίου όπως το όνομα, ο τύπος (μονόκλινο, δίκλινο, οικογενειακό), η τιμή ανά διανυκτέρευση, η μέγιστη χωρητικότητα και υπάρχει και ο σύνδεσμος που οδηγεί στην κύρια φωτογραφία. Ο συγκεκριμένος πίνακας συνδέεται με άλλους δύο βασικούς πίνακες: τον πίνακα bookings, ο οποίος αφορά την καταγραφή των κρατήσεων, και τον πίνακα room_images, που αφορά την αποθήκευση περισσότερων φωτογραφιών που σχετίζονται με κάθε δωμάτιο [28].

Ο πίνακας bookings κάνει καταγραφή των κρατήσεων που γίνονται από τους επισκέπτες. Περιλαμβάνει, μεταξύ άλλων, το αναγνωριστικό του δωματίου (room_id), το ονοματεπώνυμο και το email του πελάτη, τις ημερομηνίες check-in και check-out, την κατάσταση της κράτησης (π.χ. σε αναμονή, εγκεκριμένη, απορριφθείσα), καθώς και ένα χρονικό αποτύπωμα της δημιουργίας της κράτησης. Η σχέση του με τον πίνακα rooms διευκολύνει την εύρεση και το φιλτράρισμα ανά δωμάτιο και την υποστήριξη στατιστικών ή λειτουργικών ερωτημάτων.

Ο πίνακας users διαχειρίζεται τους λογαριασμούς των ατόμων που ελέγχουν την εφαρμογή, καθώς οι επισκέπτες δεν χρειάζεται να δημιουργήσουν τον δικό τους λογαριασμό. Κάθε χρήστης χαρακτηρίζεται από ένα μοναδικό id, το email σύνδεσης, τον κωδικό πρόσβασης (ο οποίος αποθηκεύεται κρυπτογραφημένος με χρήση bcrypt) και τον ρόλο του στο σύστημα. Η ύπαρξη του

πεδίου role καθιστά δυνατή την παροχή διαφορετικών ειδών πρόσβασης σε διαφορετικούς ανθρώπους στο μέλλον.

Για την ενίσχυση της παρουσίασης των δωματίων σε οπτικό πλαίσιο, χρησιμοποιείται ο πίνακας room_images, με τον οποίο δίνεται η δυνατότητα να προστεθούν πολλές εικόνες για κάθε δωμάτιο. Ο κάθε σύνδεσμος εικόνας (image_url) συσχετίζεται με το κατάλληλο room_id, σχηματίζοντας μία σχέση τύπου 1:N. Ο σχεδιασμός του πίνακα βοηθά αφενός στην δυναμική εμφάνιση των εικόνων στον ιστότοπο, αφετέρου διευκολύνει την προσθήκη περισσότερων εικόνων στο μέλλον.

Η σχεσιακή δομή της βάσης δεδομένων ακολουθεί απλές, λειτουργικές και αποδοτικές σχέσεις. Συγκεκριμένα:

- Ένα δωμάτιο (rooms) μπορεί να έχει πολλές κρατήσεις (bookings) – σχέση 1:N.
- Ένα δωμάτιο μπορεί να έχει πολλές εικόνες (room_images) – επίσης 1:N.
- Οι χρήστες (users) δεν χρειάζεται να έχουν ειδικό λογαριασμό για να κάνουν κράτηση. Αντίθετα, τα στοιχεία της κράτησής τους αποθηκεύονται ακριβώς για τον χρόνο που χρειάζονται, χωρίς να δημιουργηθεί μόνιμος λογαριασμός.

Η δομή μπορεί να αλλάξει εύκολα αν χρειαστεί να γίνει προσθήκη περισσότερων δεδομένων αργότερα, όπως η προσθήκη νέων πινάκων για αξιολογήσεις (reviews), μηνύματα επικοινωνίας, ή σύνθετοι κανόνες διαθεσιμότητας (availability rules), χωρίς να χρειάζονται ριζικές αλλαγές στη βάση [29].

Η συνολική σχεδίαση προσφέρει:

- Γρήγορη και αποτελεσματική ανάκτηση δεδομένων,
- Εύκολη ενσωμάτωση εργαλείων ORM όπως Sequelize ή Prisma για Node.js περιβάλλον,
- Ασφαλή αποθήκευση των credentials των διαχειριστών με χρήση hashing,
- Υψηλή συμβατότητα για πιθανή μελλοντική migration σε άλλα RDBMS με ελάχιστες τροποποιήσεις.

Η συνδυασμένη χρήση της PostgreSQL με καλά δομημένα μοντέλα δεδομένων επιτρέπει την εφαρμογή να παραμείνει αξιόπιστη, να επεκταθεί εάν χρειαστεί και να αλλάξει εύκολα στο μέλλον, είτε όσον αφορά το πώς λειτουργεί στα παρασκήνια είτε όσον αφορά τον τρόπο που είναι κατασκευασμένη.

4.4 Δομή backend

Το backend της εφαρμογής "Studio Liotrivi" έχει κατασκευαστεί με τη χρήση της τεχνολογίας Node.js σε συνδυασμό με το framework Express.js, προσφέροντας έναν ελαφρύ, γρήγορο και επεκτάσιμο τρόπο δημιουργίας RESTful API. Η αρχιτεκτονική βασίζεται στον διαχωρισμό ευθυνών (separation of concerns), διατηρώντας τα αρχεία οργανωμένα σε φακέλους, ώστε να είναι εύκολη η κατανόηση του κώδικα, η διόρθωσή του εάν χρειάζεται και η προσθήκη νέων λειτουργιών αργότερα [4],[6].

4.4.1 Δομή Φακέλων και Αρχείων

Η δομή των φακέλων και αρχείων του backend της εφαρμογής είναι διαμορφωμένη με τρόπο που δείχνει ποια έρχονται πρώτα και πώς συνδέονται, κάνοντας πιο εύκολη την κατανόηση και την εργασία με αυτά αργότερα. Στον βασικό κατάλογο της εφαρμογής, με όνομα hotel-booking-app/, περιλαμβάνονται, όπως φαίνεται την εικόνα 4.1, διάφοροι υποφάκελοι και αρχεία, το καθένα με διακριτό ρόλο. Ο φάκελος config/ περιέχει το αρχείο db.js, το οποίο είναι υπεύθυνο για τη διαχείριση της σύνδεσης με τη βάση δεδομένων PostgreSQL. Ο φάκελος node_modules/ φιλοξενεί τις

εξαρτήσεις που εγκαθίστανται μέσω npm, οι οποίες είναι απαραίτητες για τη λειτουργία της εφαρμογής.

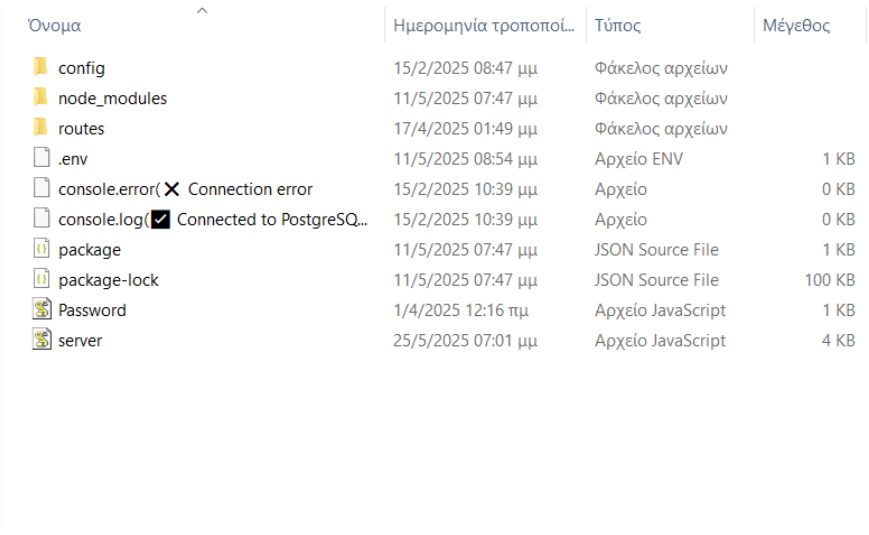
Στον φάκελο routes/ βρίσκονται τα αρχεία bookings.js και rooms.js, τα οποία περιέχουν τις διαδρομές (routes) που σχετίζονται αντίστοιχα με τις κρατήσεις και τα δωμάτια. Το αρχείο .env περιέχει τις περιβαλλοντικές μεταβλητές, όπως τα στοιχεία σύνδεσης με τη βάση δεδομένων και το κρυφό κλειδί για τα JWT tokens. Το package.json περιλαμβάνει τις εξαρτήσεις του έργου και τα αντίστοιχα scripts, ενώ το server.js αποτελεί το κύριο αρχείο εκκίνησης του Express server, στο οποίο γίνεται η αρχικοποίηση του διακομιστή, η φόρτωση των routes και η εφαρμογή των απαραίτητων middleware.

Κάθε στοιχείο διαδραματίζει έναν ξεχωριστό ρόλο στην εφαρμογή:

- server.js: Το βασικό αρχείο που αρχικοποιεί τον Express server, φορτώνει τα routes, εφαρμόζει middleware (CORS, JSON parsing), και εκκινεί τον server στη θύρα που δηλώνεται στο .env.
- config/db.js: Περιλαμβάνει τον κώδικα σύνδεσης με τη βάση PostgreSQL μέσω του πακέτου pg. Εδώ δηλώνονται οι μεταβλητές περιβάλλοντος, ενώ περιέχονται και μηνύματα ελέγχου επιτυχίας/σφάλματος (π.χ. Connected to PostgreSQL, Connection error) [9].
- routes/: Περιέχει όλα τα αρχεία που καθορίζουν τα endpoints του API. Αυτά τα αρχεία περιλαμβάνουν τη λογική για επεξεργασία αιτημάτων CRUD (Create, Read, Update, Delete) [30].
- .env: Αποθηκεύει ευαίσθητες ρυθμίσεις, όπως το connection string για τη βάση δεδομένων, το secret key για τα JWT tokens, καθώς και το port του server [31].

  Παράδειγμα:

```
ini
ΑντιγραφήΕπεξεργασία
PORT=5000
DB_HOST=localhost
DB_USER=postgres
DB_PASS=****
DB_NAME=hotelbooking
JWT_SECRET=*****
```



Εικόνα 4.1: Δομή φακέλων του backend της εφαρμογής "Studio Liotrivi".

4.4.2 Βασικά Routes και Λειτουργίες

Η εφαρμογή χρησιμοποιεί ένα σύνολο RESTful routes, που είναι οργανωμένα σε αρχεία bookings.js και rooms.js. Αυτά τα endpoints επιτρέπουν την αλληλεπίδραση frontend-backend μέσω HTTP αιτημάτων με ανταλλαγή δεδομένων σε μορφή JSON [6].

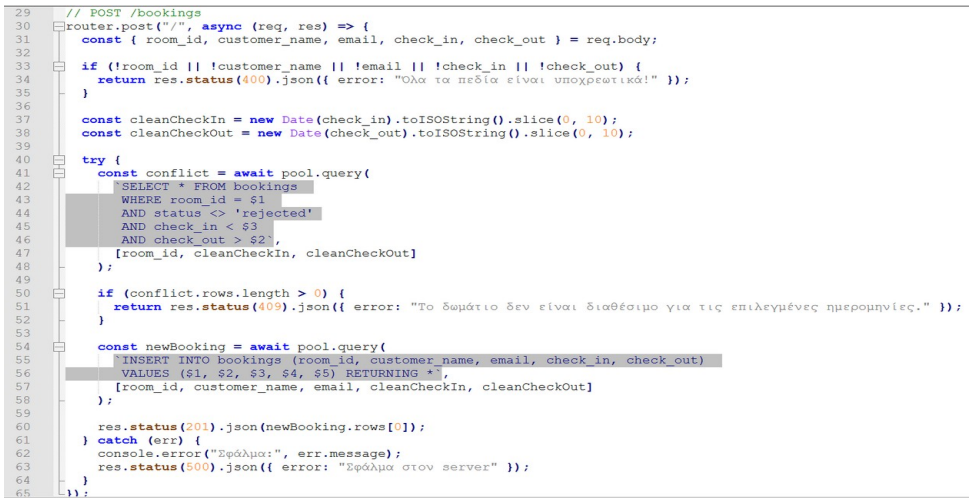
➤ **Bookings Routes (routes/bookings.js)**

Τα routes που σχετίζονται με τις κρατήσεις επιτρέπουν την πλήρη διαχείριση της σχετικής λειτουργικότητας μέσω HTTP αιτημάτων. Ο χρήστης μπορεί να δει όλες τις κρατήσεις που έχουν αποθηκευτεί, αποστέλλοντας αίτημα τύπου GET προς το endpoint /bookings. (Εικ.4.2). Για τη δημιουργία νέας κράτησης, χρησιμοποιείται αίτημα POST στο ίδιο endpoint, όπου ο πελάτης υποβάλλει τα απαραίτητα στοιχεία (όνομα, email, ημερομηνίες διαμονής). Εφόσον υπάρχει καταχωρημένη κράτηση, ο χρήστης ή ο διαχειριστής μπορεί να την επεξεργαστεί με τη μέθοδο PUT στο endpoint /bookings/:id, όπου το :id αντιστοιχεί στο μοναδικό αναγνωριστικό της κράτησης. Αντίστοιχα, η διαγραφή μιας κράτησης πραγματοποιείται με DELETE στο ίδιο endpoint.(Εικ. 4.3).

Επιπλέον, παρέχεται η δυνατότητα ελέγχου της διαθεσιμότητας των δωματίων μέσω του endpoint /bookings/available-rooms, όπου με GET αίτημα και κατάλληλα παραμετροποιημένες ημερομηνίες επιστροφής και αναχώρησης, το backend επαρέχει μια λίστα με δωμάτια τα οποία είναι διαθέσιμα για τις ημερομηνίες που πρόκειται να γίνει η κράτηση. Τα δεδομένα αποθηκεύονται στον πίνακα bookings, και κάθε αίτημα σχετίζεται με έναν room_id. Η λογική ελέγχει τη διαθεσιμότητα συγκρίνοντας τις ημερομηνίες check_in και check_out με ήδη καταχωρημένες κρατήσεις.

```
6   // GET /bookings
7   router.get("/", async (req, res) => {
8     try {
9       const result = await pool.query(`
10        SELECT
11          id,
12          room_id,
13          customer_name,
14          email,
15          TO_CHAR(check_in, 'YYYY-MM-DD')   AS check_in,
16          TO_CHAR(check_out, 'YYYY-MM-DD')  AS check_out,
17          status
18        FROM bookings
19      `);
20      res.json(result.rows);
21    } catch (err) {
22      console.error("Σφάλμα:", err.message);
23      res.status(500).json({ error: "Σφάλμα στον server" });
24    }
25  });
```

Εικόνα 4.2: Κώδικας για τις διαδρομές GET του endpoint /bookings.



```
// POST /bookings
router.post("/", async (req, res) => {
  const { room_id, customer_name, email, check_in, check_out } = req.body;

  if (!room_id || !customer_name || !email || !check_in || !check_out) {
    return res.status(400).json({ error: "Όλα τα πεδία είναι υποχρεωτικά!" });
  }

  const cleanCheckIn = new Date(check_in).toISOString().slice(0, 10);
  const cleanCheckOut = new Date(check_out).toISOString().slice(0, 10);

  try {
    const conflict = await pool.query(
      `SELECT * FROM bookings
       WHERE room_id = $1
       AND status <> 'rejected'
       AND check_in < $3
       AND check_out > $2`,
      [room_id, cleanCheckIn, cleanCheckOut]
    );

    if (conflict.rows.length > 0) {
      return res.status(409).json({ error: "Το δωμάτιο δεν είναι διαθέσιμο για τις επιλεγμένες ημερομηνίες." });
    }

    const newBooking = await pool.query(
      `INSERT INTO bookings (room_id, customer_name, email, check_in, check_out)
       VALUES ($1, $2, $3, $4, $5) RETURNING *`,
      [room_id, customer_name, email, cleanCheckIn, cleanCheckOut]
    );

    res.status(201).json(newBooking.rows[0]);
  } catch (err) {
    console.error("Σφάλμα:", err.message);
    res.status(500).json({ error: "Σφάλμα στον server" });
  }
});
```

Εικόνα 4.3: Κώδικας για τις διαδρομές POST του endpoint /bookings.

➤ Rooms Routes (`routes/rooms.js`)

Τα routes που σχετίζονται με τα δωμάτια της εφαρμογής είναι υπεύθυνα για την ανάκτηση πληροφοριών σχετικά με τη διαθεσιμότητα, τα χαρακτηριστικά και τις εικόνες κάθε δωματίου. Μέσω της μεθόδου GET στο endpoint /rooms, το backend παρέχει μία λίστα με όλα τα διαθέσιμα δωμάτια, συμπεριλαμβανομένων και των βασικών τους πεδίων, όπως το όνομα, τον τύπο, την τιμή και τη χωρητικότητα. (Εικ 4.4)

Επιπλέον, με GET αίτημα στο endpoint /rooms/:id/images, όπου το :id αντιστοιχεί στο αναγνωριστικό κάθε δωματίου, είναι δυνατή η ανάκτηση όλων των εικόνων που σχετίζονται με το συγκεκριμένο δωμάτιο. Οι πληροφορίες αυτές αντλούνται από τους πίνακες rooms και room_images, και παρέχονται στο frontend σε μορφή JSON, επιτρέποντας την οπτική απεικόνιση του καταλύματος με τρόπο λειτουργικό και φιλικό προς τον χρήστη.

```
router.get("/available-rooms", async (req, res) => {
  console.log("🔍 RAW QUERY PARAMS:", req.query);
  const { check_in, check_out, people } = req.query;
  const numberOfPeople = parseInt(people, 10);

  console.log("➡️ check_in:", check_in);
  console.log("➡️ check_out:", check_out);
  console.log("➡️ numberOfPeople:", numberOfPeople);

  if (!check_in || !check_out || isNaN(numberOfPeople)) {
    return res.status(400).json({ error: "Λείπουν ή είναι άκυρες οι παράμετροι." });
  }

  try {
    const query = `
      SELECT *
      FROM rooms
      WHERE capacity >= $1
        AND id NOT IN (
          SELECT room_id
          FROM bookings
          WHERE status <> 'rejected'
            AND check_in < $3
            AND check_out > $2
        )
    `;

    const values = [numberOfPeople, check_in, check_out];

    console.log("📄 Εκτελώ SQL με:", values);

    const result = await pool.query(query, values);

    res.json(result.rows);
  } catch (err) {
    console.error("❌ Σφάλμα κατά την αναζήτηση δωματίων:", err.message);
    res.status(500).json({ error: "Σφάλμα στον server" });
  }
});
```

Εικόνα 4.4: Route που επιστρέφει τα διαθέσιμα δωμάτια με βάση ημερομηνίες και αριθμό ατόμων.

4.4.3 Middleware και Ασφάλεια

Στο αρχείο server.js του backend χρησιμοποιούνται διάφορα middleware που ενισχύουν τη λειτουργικότητα και την ασφάλεια της εφαρμογής. Το express.json() χρησιμοποιείται για την επεξεργασία των εισερχόμενων αιτημάτων σε μορφή JSON, επιτρέποντας στο διακομιστή να διαβάζει σωστά τα δεδομένα που αποστέλλονται από το frontend. Το cors() ενεργοποιεί την πολιτική CORS (Cross-Origin Resource Sharing), επιτρέποντας την επικοινωνία μεταξύ του frontend και του backend ακόμη και όταν αυτά φιλοξενούνται σε διαφορετικά origins.(εικ.4.5)

Επιπλέον, με τη χρήση του dotenv.config(), η εφαρμογή μπορεί να διαβάζει περιβαλλοντικές μεταβλητές από το αρχείο .env, εξασφαλίζοντας διαχωρισμό του κώδικα από ευαίσθητα δεδομένα, όπως είναι τα στοιχεία σύνδεσης με τη βάση δεδομένων ή μυστικά κλειδιά (π.χ. για authentication).

Η διαχείριση της ασφάλειας των χρηστών πραγματοποιείται με τη χρήση JWT (JSON Web Tokens) για την αυθεντικοποίηση των διαχειριστών. Οι κωδικοί πρόσβασης αποθηκεύονται με ασφάλεια, εφαρμόζοντας hashing μέσω της βιβλιοθήκης bcryptjs, ώστε να διασφαλίζεται η προστασία των διαπιστευτηρίων ακόμη και σε περίπτωση παραβίασης της βάσης δεδομένων [12], [23].

```
1   require("dotenv").config();
2   const express = require("express");
3   const cors = require("cors");
4   const bcrypt = require("bcryptjs");
5   const jwt = require("jsonwebtoken");
6   const pool = require("./config/db");
7
8   const app = express();
9   app.use(cors());
10  app.use(express.json());
```

Εικόνα 4.5: Φόρτωση middleware και παραμέτρων περιβάλλοντος στην κεντρική εφαρμογή.

4.4.4 Συνοπτικός Πίνακας Routes

Ο παρακάτω πίνακας (4.1) συνοψίζει όλα τα βασικά endpoints που προσφέρει το backend της εφαρμογής. Τα endpoints είναι οργανωμένα ανά κατηγορία λειτουργιών (κρατήσεις, δωμάτια, αυθεντικοποίηση και επικοινωνία) και περιλαμβάνουν τη μέθοδο HTTP που χρησιμοποιείται, καθώς και μία σύντομη περιγραφή της λειτουργίας τους. Αυτή η παρουσίαση επιτρέπει μια άμεση κατανόηση του τρόπου με τον οποίο το frontend επικοινωνεί με το backend, διευκολύνοντας τόσο την ανάπτυξη όσο και τη μελλοντική συντήρηση ή επέκταση του API.

- **Κατηγορία Bookings**: Περιλαμβάνει endpoints για τη δημιουργία, ανάκτηση, ενημέρωση και διαγραφή κρατήσεων, καθώς και για τον έλεγχο της διαθεσιμότητας δωματίων.
- **Κατηγορία Rooms**: Αφορά την προβολή των διαθέσιμων δωματίων και των σχετικών εικόνων τους.
- **Κατηγορία Auth**: Διαχειρίζεται τη διαδικασία σύνδεσης και εγγραφής νέων διαχειριστών στο σύστημα.
- **Κατηγορία Επικοινωνίας**: Προσφέρει endpoint για την αποστολή μηνυμάτων μέσω φόρμας επικοινωνίας που συνδέεται με το backend.

Ο πίνακας λειτουργεί ως οδηγός για τους προγραμματιστές ή τους διαχειριστές συστημάτων που θέλουν να κατανοήσουν γρήγορα τις διαθέσιμες λειτουργίες του API, χωρίς να χρειάζεται να εξερευνήσουν ολόκληρη την υλοποίηση.

Πίνακας 4.1: Συνοπτική παρουσίαση των βασικών RESTful endpoints του backend της εφαρμογής

Κατηγορία	Endpoint	Μέθοδος	Περιγραφή
Bookings	`/bookings`	GET	Λήψη όλων των κρατήσεων
	`/bookings`	POST	Υποβολή νέας κράτησης
	`/bookings/:id`	PUT	Ενημέρωση κράτησης
	`/bookings/:id`	DELETE	Διαγραφή κράτησης
	`/bookings/available-rooms`	GET	Λήψη διαθέσιμων δωματίων για συγκεκριμένες ημερομηνίες
Rooms	`/rooms`	GET	Λίστα δωματίων
	`/rooms/:id/images`	GET	Εικόνες δωματίου
Auth	`/login`	POST	Είσοδος χρήστη
	`/register`	POST	Εγγραφή νέου διαχειριστή (προαιρετικό)
Επικοινωνία	`/contact`	POST	Αποστολή email μέσω φόρμας επικοινωνίας

Με βάση τον παραπάνω πίνακα (routes), γίνεται ξεκάθαρο ότι η δομή του backend ακολουθεί τα βασικά πρότυπα ανάπτυξης RESTful APIs με Express.js, αξιοποιώντας modular προσέγγιση και απλό routing. Χάρη σε αυτόν τον σχεδιασμό, η μελλοντική επέκταση του συστήματος είναι ιδιαίτερα εύκολη, καθώς μπορούν να προστεθούν νέες λειτουργίες (π.χ. αξιολογήσεις, αρχειοθέτηση) ή να μεταβεί το έργο σε πιο σύνθετη αρχιτεκτονική (όπως MVC ή service-based). Ο διαχωρισμός των ευθυνών (routing – configuration – server logic) διευκολύνει τη συντήρηση, τη διόρθωση προβλημάτων και επιτρέπει την ομαλή συνεργασία πολλών προγραμματιστών στην ίδια βάση κώδικα.

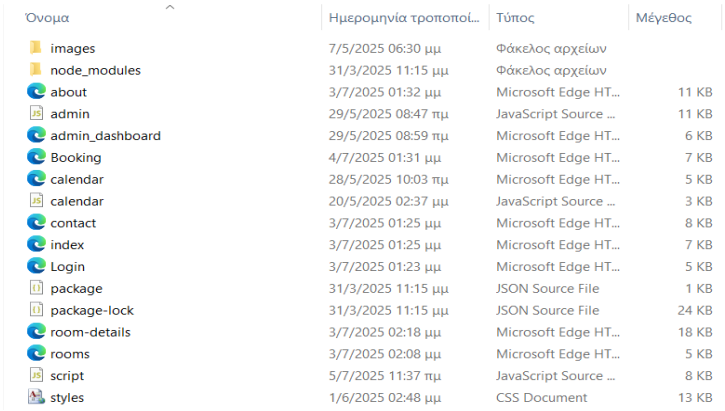
4.5 Δομή Frontend (σελίδες, αρχεία)

Το frontend της εφαρμογής αποτελεί το τμήμα που αλληλεπιδρά άμεσα με τον χρήστη, είτε πρόκειται για τον επισκέπτη που πραγματοποιεί κράτηση, είτε για τον διαχειριστή του καταλύματος. Το περιβάλλον εργασίας σχεδιάστηκε ώστε να είναι εύχρηστο, ξεκαθαρό, λειτουργικό και συμβατό με διαφορετικούς τύπους συσκευών (responsive σχεδιασμός). Δημιουργήθηκε και ολοκληρώθηκε με τη χρήση HTML5, CSS3 και JavaScript, χωρίς τη χρήση κάποιου frontend framework (π.χ. React, Vue), ώστε να παραμείνει η εφαρμογή ελαφριά και ευέλικτη.

➤ Δομή Αρχείων

Η βασική δομή του frontend ακολουθεί μια οργανωμένη διάταξη φακέλων και αρχείων. Περιλαμβάνει επιμέρους αρχεία HTML για κάθε μία σελίδα της εφαρμογής, όπως τα: index.html, rooms.html, booking.html, about.html, contact.html, login.html, admin_dashboard.html, calendar.html και room-details.html. Παράλληλα, περιλαμβάνονται τα αρχεία styles.css για την μορφοποίηση, καθώς και τα script.js, admin.js και calendar.js για τις διάφορες JavaScript λειτουργίες της εφαρμογής. Τέλος, υπάρχει και ο φάκελος images/, στον οποίο αποθηκεύονται οι εικόνες των δωματίων, τα λογότυπα και τα στοιχεία του carousel. (Εικ.4.6)

Η χρήση ξεχωριστών HTML αρχείων για κάθε ενότητα της ιστοσελίδας επιτρέπει σαφή διαχωρισμό των λειτουργιών, ενώ τα κοινά αρχεία CSS και JavaScript συμβάλλουν στην επαναχρησιμοποίηση του κώδικα και διευκολύνουν τη συντήρηση της εφαρμογής.



Όνομα	Ημερομηνία τροποποί...	Τύπος	Μέγεθος
images	7/5/2025 06:30 μμ	Φάκελος αρχείων	
node_modules	31/3/2025 11:15 μμ	Φάκελος αρχείων	
about	3/7/2025 01:32 μμ	Microsoft Edge HT...	11 KB
admin	29/5/2025 08:47 πμ	JavaScript Source ...	11 KB
admin_dashboard	29/5/2025 08:59 πμ	Microsoft Edge HT...	6 KB
Booking	4/7/2025 01:31 μμ	Microsoft Edge HT...	7 KB
calendar	28/5/2025 10:03 πμ	Microsoft Edge HT...	5 KB
calendar	20/5/2025 02:37 μμ	JavaScript Source ...	3 KB
contact	3/7/2025 01:25 μμ	Microsoft Edge HT...	8 KB
index	3/7/2025 01:25 μμ	Microsoft Edge HT...	7 KB
Login	3/7/2025 01:23 μμ	Microsoft Edge HT...	5 KB
package	31/3/2025 11:15 μμ	JSON Source File	1 KB
package-lock	31/3/2025 11:15 μμ	JSON Source File	24 KB
room-details	3/7/2025 02:18 μμ	Microsoft Edge HT...	18 KB
rooms	3/7/2025 02:08 μμ	Microsoft Edge HT...	5 KB
script	5/7/2025 11:37 πμ	JavaScript Source ...	8 KB
styles	1/6/2025 02:48 μμ	CSS Document	13 KB

Εικόνα 4.6: Δομή φακέλων του frontend της εφαρμογής "Studio Liotrivi".

➤ **Περιγραφή Κύριων Σελίδων**

- **index.html**: Η αρχική σελίδα της εφαρμογής λειτουργεί ως ψηφιακή "βιτρίνα" του καταλύματος. Περιλαμβάνει δυναμικό carousel εικόνων από τα δωμάτια και σύντομα μηνύματα καλωσορίσματος. Ο χρήστης μπορεί άμεσα να μεταβεί στη σελίδα διαθεσιμότητας δωματίων μέσω κουμπιού δράσης.
- **rooms.html**: Σε αυτήν τη σελίδα πραγματοποιείται η αναζήτηση διαθεσιμότητας δωματίων (Εικ.4.7). Ο χρήστης συμπληρώνει πεδία check-in, check-out καθώς και τον αριθμό των ατόμων για τα οποία ενδιαφέρεται να κάνει κράτηση. Στη συνέχεια, με χρήση των JavaScript και fetch API, αποστέλλεται αίτημα προς τον server και εμφανίζονται τα διαθέσιμα δωμάτια. Κάθε δωμάτιο περιέχει τις εικόνες, την τιμή, τη χωρητικότητα και το κουμπί «Κάντε Κράτηση».

```
33      <label for="check-in" data-i18n="rooms.checkin">Άφιξη:</label>
34      <input type="date" id="check-in" name="check-in" required>
35
36      <label for="check-out" data-i18n="rooms.checkout">Αναχώρηση:</label>
37      <input type="date" id="check-out" name="check-out" required>
38
39      <label for="guests" data-i18n="rooms.guests">Άτομα:</label>
40      <input type="number" id="guests" name="guests" min="1" required>
41
42      <button id="search-btn" type="button" data-i18n="rooms.search">🔍 Αναζήτηση Διαθεσιμότητας</button>
43
44  <div id="rooms-container">
```

Εικόνα 4.7: Φόρμα αναζήτησης διαθεσιμότητας δωματίων από τον επισκέπτη.

- **booking.html**: Αποτελεί τη σελίδα ολοκλήρωσης της κράτησης. Ο χρήστης συμπληρώνει τα προσωπικά του στοιχεία (ονοματεπώνυμο, email), καθώς και τις ημερομηνίες κράτησης (Εικ. 4.8). Η κράτηση υποβάλλεται μέσω POST αιτήματος στον server, και ο χρήστης λαμβάνει κατάλληλα μηνύματα επιτυχίας ή αποτυχίας ανάλογα με τη διαθεσιμότητα.

```
<form id="booking-form" class="booking-layout">
  <div class="form-left">
    <label for="room_id" data-i18n="booking.roomType">Τύπος Δωματίου:</label>
    <select id="room_id" name="room_id" required></select>

    <label for="name" data-i18n="booking.name">Ονοματεπώνυμο:</label>
    <input type="text" id="name" name="name" required>

    <label for="email" data-i18n="booking.email">Email:</label>
    <input type="email" id="email" name="email" required>
  </div>

  <div class="form-right">
    <div class="dates">
      <div>
        <label for="checkin_date" data-i18n="booking.checkin">Ημερομηνία Άφιξης:</label>
        <input type="date" id="checkin_date" name="checkin_date" required>
      </div>
      <div>
        <label for="checkout_date" data-i18n="booking.checkout">Ημερομηνία Αναχώρησης:</label>
        <input type="date" id="checkout_date" name="checkout_date" required>
      </div>
    </div>
    <div class="submit-wrapper">
      <button type="submit" data-i18n="booking.submit">Κάντε Κράτηση</button>
    </div>
  </div>
</form>
```

Εικόνα 4.8: Φόρμα κράτησης με πεδία για όνομα, email, ημερομηνίες.

- **room-details.html**: Σελίδα με λεπτομέρειες δωματίου. Εδώ παρουσιάζονται επιπλέον πληροφορίες για κάθε δωμάτιο, δυναμικά φορτωμένες από τη βάση δεδομένων (τιμή, περιγραφή, εικόνες). Υπάρχει δυνατότητα μεγέθυνσης φωτογραφιών μέσω lightbox και απευθείας σύνδεση με τη φόρμα κράτησης.
- **about.html**: Παρέχει πληροφορίες για το κατάλυμα Studio Liotrivi, την ιστορία του, τις παροχές (π.χ. Wi-Fi, ιδιωτικό πάρκινγκ) και συχνές ερωτήσεις (FAQ). Περιλαμβάνει slider εικόνων για αισθητική παρουσίαση και ενίσχυση της εμπειρίας χρήστη.
- **contact.html**: Σελίδα επικοινωνίας όπου οι χρήστες μπορούν να αποστείλουν μήνυμα μέσω φόρμας, η οποία υποβάλλεται στον server μέσω POST αιτήματος και email (χρησιμοποιώντας nodemailer στο backend). Παράλληλα, εμφανίζεται χάρτης Google Maps με την ακριβή τοποθεσία του καταλύματος.
- **login.html**: Σελίδα σύνδεσης αποκλειστικά για διαχειριστές. Μετά τη συμπλήρωση email και κωδικού, πραγματοποιείται έλεγχος στο backend και δημιουργείται JWT token, το οποίο αποθηκεύεται στο localStorage για την αυθεντικοποίηση του χρήστη στις προστατευμένες σελίδες.

➤ **Σελίδες Διαχείρισης**

Οι παρακάτω σελίδες είναι προσβάσιμες μόνο από τον διαχειριστή του καταλύματος, ο οποίος έχει κάνει login.

- **admin_dashboard.html**: Παρέχει πίνακα κρατήσεων, με δυνατότητα έγκρισης ή απόρριψης αιτήσεων, επεξεργασίας ή διαγραφής (Εικ 4.9). Τα δεδομένα αντλούνται από τον backend με χρήση JavaScript και fetch API. Επίσης, εμφανίζεται λίστα με διαθέσιμα δωμάτια και δυνατότητα προσθήκης νέου δωματίου.

```
58    <table id="pending-table">
59      <thead>
60        <tr>
61          <th>Όνομα</th>
62          <th>Email</th>
63          <th>Δωμάτιο</th>
64          <th>Άφιξη</th>
65          <th>Αναχώρηση</th>
66          <th>Ενέργεια</th>
67        </tr>
68      </thead>
69      <tbody></tbody>
70    </table>
```

Εικόνα 4.9: Πίνακας εκκρεμών αιτήσεων στην περιοχή του διαχειριστή.

- **calendar.html**: Οπτική απεικόνιση κρατήσεων ανά δωμάτιο με χρήση της βιβλιοθήκης FullCalendar (Εικ. 4.10). Ο διαχειριστής μπορεί να επιλέξει δωμάτιο και να δει όλες τις εγκεκριμένες κρατήσεις στο ημερολόγιο.

```
79        const calendar = new FullCalendar.Calendar(calendarEl, {
80          initialView: "dayGridMonth",
81          height: 500,
82          events: [],
83          displayEventTime: false,
84          eventTimeFormat: {
85            hour: "2-digit",
86            minute: "2-digit",
87            hour12: false
88          }
89        });
90
```

Εικόνα 4.10: Ενσωμάτωση του ημερολογίου κρατήσεων στο frontend με FullCalendar.

Ο τρόπος που λειτουργεί το frontend υποστηρίζεται σε μεγάλο βαθμό από την JavaScript, η οποία εφαρμόζεται σε όλα τα HTML αρχεία της εφαρμογής. Μέσω των αντίστοιχων scripts, υλοποιείται η διαχείριση μεταφράσεων με χρήση αντικειμένων τύπου translations, προσφέροντας έτσι την υποστήριξη για ελληνικά και αγγλικά. Επιπλέον, πραγματοποιούνται αιτήματα fetch προς το backend για την δυναμική φόρτωση δεδομένων, όπως είναι τα δωμάτια, οι κρατήσεις και οι εικόνες. Η JavaScript επαληθεύει επίσης τα δεδομένα εισόδου πριν την αποστολή (validation) και φροντίζει για την αποθήκευση βασικών στοιχείων στο localStorage, όπως το JWT token και η επιλεγμένη γλώσσα του χρήστη. Παράλληλα, ενεργοποιούνται λειτουργίες όπως lightbox, slide galleries και responsive carousel στις σχετικές σελίδες (π.χ. index.html και about.html).

Ο σχεδιασμός του frontend έγινε με επίκεντρο τη βελτιστοποίηση της εμπειρίας του χρήστη σε διάφορες συσκευές, όπως κινητά, tablets και επιτραπέζιους υπολογιστές, μέσω της χρήσης viewport

ρυθμίσεων και CSS media queries. Είναι εμφανής ο διαχωρισμός μεταξύ των ρόλων του επισκέπτη και του διαχειριστή, προσπαθώντας έτσι να βελτιωθεί η ασφάλεια και η απλότητα στη χρήση. Το περιβάλλον διεπαφής (UI) διατηρείται καθαρό και φιλικό προς όλους τους χρήστες, ανεξαρτήτως τεχνικού επιπέδου.

Εν κατακλείδι, η frontend δομή του Studio Liotrivi προσφέρει μια ολοκληρωμένη εμπειρία κράτησης, η οποία είναι βασισμένη σε έναν απλό αλλά λειτουργικό σχεδιασμό. Ο χρήστης οδηγείται βήμα-βήμα από το αν θέλει να κάνει κράτηση μέχρι την επιβεβαίωσή της, ενώ ο διαχειριστής μπορεί να δει τα πάντα και να διαχειριστεί το σύστημα εύκολα χρησιμοποιώντας ειδικά εργαλεία.

5 Υλοποίηση της εφαρμογής

5.1 Περιβάλλον Ανάπτυξης

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε εξ ολοκλήρου σε τοπικό περιβάλλον, χρησιμοποιώντας μια σειρά από δωρεάν και χρήσιμα εργαλεία ανοιχτού κώδικα, τα οποία είναι κατάλληλα για την υλοποίηση ενός δυναμικού συστήματος διαχείρισης κρατήσεων. Το λειτουργικό σύστημα που χρησιμοποιήθηκε είναι τα Windows 10, σε συνδυασμό με την πλατφόρμα XAMPP, η οποία επέτρεψε την τοπική φιλοξενία και εξυπηρέτηση του frontend μέρους της εφαρμογής, που υλοποιήθηκε με HTML, CSS και JavaScript.

Η συγγραφή και επεξεργασία του κώδικα έγινε μέσω του Visual Studio Code (VS Code), ενός σύγχρονου επεξεργαστή που παρέχει υποστήριξη σε διαφορετικές γλώσσες προγραμματισμού και επεκτάσεις, προσφέροντας λειτουργίες όπως syntax, highlighting, linting και debugging. Το εργαλείο αυτό βοήθησε στο να γίνει πιο γρήγορα η ανάπτυξη και η οργάνωση του έργου [32].

Το κομμάτι του backend δημιουργήθηκε με την πλατφόρμα Node.js, στην έκδοση 18.x.x, αξιοποιώντας το framework Express.js για την κατασκευή RESTful API endpoints, μέσω των οποίων επιτυγχάνεται η επικοινωνία μεταξύ του frontend και της βάσης δεδομένων. Για τη σύνδεση με τη βάση χρησιμοποιήθηκε η βιβλιοθήκη `pg`, η οποία παρέχει ένα απλό και αποδοτικό API για αλληλεπίδραση με συστήματα PostgreSQL, όπως έχει αναφερθεί και πιο πάνω.

Η βάση δεδομένων της εφαρμογής αναπτύχθηκε με το σύστημα PostgreSQL, έκδοσης 15.x, το οποίο επιλέχθηκε λόγω της αξιοπιστίας του και της παροχής ολοκληρωμένων λύσεων με πολλές δυνατότητες. Η αποθήκευση των δεδομένων αφορά πίνακες σχετικούς με χρήστες, δωμάτια, κρατήσεις και εικόνες. Η διαχείριση της βάσης, καθώς και η εμφάνιση των δεδομένων, πραγματοποιήθηκαν μέσω του εργαλείου pgAdmin, το οποίο προσφέρει εικόνες και εργαλεία για να γίνουν κατανοητά τα δεδομένα, οι κανόνες και ο τρόπος με τον οποίο συνδέονται τα πάντα μεταξύ τους [35].

Για τον έλεγχο εκδόσεων και την αποθήκευση του έργου χρησιμοποιήθηκε το Git, ενώ ο κώδικας του έργου φιλοξενήθηκε στο GitHub. Ο φάκελος του έργου έχει όλα τα απαραίτητα μέρη και είναι ανοιχτός για να τον δουν, να τον χρησιμοποιήσουν και να τον επεξεργαστούν όλοι. Περιέχει αρχείο τεκμηρίωσης README, άδεια χρήσης τύπου MIT και τα αρχεία που χρησιμοποιήθηκαν οργανωμένα σε ξεχωριστούς φακέλους για backend και frontend [18],[19].

Κατά τη διάρκεια την δημιουργίας του κώδικα χρησιμοποιήθηκαν και ειδικά εργαλεία για δοκιμές και έλεγχο ποιότητας. Το Postman αξιοποιήθηκε για τη δοκιμή των API endpoints του backend (μέθοδοι GET, POST, PUT, DELETE), ενώ τα εργαλεία ανάπτυξης του προγράμματος περιήγησης (Developer Tools) σε Chrome και Edge συνέβαλαν στον εντοπισμό σφαλμάτων CSS και στην ανάλυση του DOM. Παράλληλα, το pgAdmin χρησιμοποιήθηκε σε συνεχή βάση ώστε να γίνει ο έλεγχος της ακεραιότητας των δεδομένων στους πίνακες.

Η δομή της εφαρμογής, που αναπτύχθηκε για την δημιουργία αυτής της πτυχιακής, χωρίζεται σε δύο κύρια μέρη. Το backend, το οποίο είναι ανεπτυγμένο σε Node.js, περιλαμβάνει βασικά αρχεία και φακέλους όπως το server.js, τους υποφακέλους routes/, config/, controllers/ και το αρχείο db.sql που περιέχει τα ερωτήματα δημιουργίας της βάσης. Το frontend μέρος φιλοξενείται στον φάκελο htdocs/Template, και περιλαμβάνει τα αρχεία HTML της διεπαφής (όπως index.html, rooms.html, about.html), τα οποία συνοδεύονται από τα αντίστοιχα αρχεία CSS (styles.css), JavaScript (script.js) και υποφακέλους όπως images/.

Το περιβάλλον αυτό βοήθησε στο να δημιουργηθεί, να δοκιμαστεί και να μοιραστεί η εφαρμογή με ασφάλεια και ευκολία. Είναι φτιαγμένο έτσι ώστε ο καθένας να μπορεί να το δει και να το χρησιμοποιήσει δωρεάν. Στο μέλλον, υπάρχει η δυνατότητα να επεκταθεί και να βελτιωθεί.

5.2 Υλοποίηση Backend

Το backend της εφαρμογής αποτελεί το κύριο μέρος του συστήματος που βοηθά στο χειρισμό και τη διαχείριση των αιτημάτων, των κρατήσεων, τον έλεγχο των χρηστών και τη συντήρηση των δεδομένων του καταλύματος. Η υλοποίηση πραγματοποιήθηκε σε περιβάλλον Node.js, με τη χρήση της βιβλιοθήκης Express.js, η οποία διαθέτει ένα ελαφρύ αλλά ισχυρό πλαίσιο για την κατασκευή RESTful API endpoints. Η αρχιτεκτονική του backend σχεδιάστηκε έχοντας ως στόχο τη δυνατότητα του κώδικα να επεκταθεί, να μπορεί να είναι ευνόητος και να διατηρεί καθαρή και διαχωρισμένη λογική ανά μονάδα λειτουργίας [10].

Βασικό ρόλο στην εκκίνηση και στην λειτουργία της εφαρμογής διαδραματ Μέσα από αυτό γίνεται η εκκίνηση της εφαρμογής Express, η σύνδεση με τη βάση δεδομένων PostgreSQL, εισάγονται οι μεταβλητές περιβάλλοντος από αρχείο .env, καθώς και ενεργοποιούνται τα απαραίτητα middleware. Η λειτουργία CORS ενεργοποιείται για την αποδοχή αιτημάτων από διαφορετικά origins, δίνοντας έτσι την δυνατότητα να επικοινωνούν ομαλά το frontend και το backend. Παράλληλα, το middleware express.json() ενεργοποιείται ώστε να γίνεται σωστό parsing των εισερχόμενων JSON δεδομένων, ενώ ο server εκκινεί τελικά με τη μέθοδο app.listen(port) (Εικ. 5.1).

```
8   const app = express();
9   app.use(cors());
10  app.use(express.json());
11
12  // ✅ Εισαγωγή routes
13  const bookingRoutes = require("./routes/bookings");
14  app.use("/bookings", bookingRoutes);  // routes για GET, POST, DELETE κρατήσεων
```

Εικόνα 5.1: Εισαγωγή CORS και express.json

Για να ενισχυθεί η ασφάλεια, η αυθεντικοποίηση και η διαχείριση δεδομένων,γίνεται χρήση διαφορετικών middleware και βιβλιοθηκών. Το bcryptjs είναι σημαντικό ώστε να γίνεται σωστά και με ασφάλεια η κρυπτογράφηση των κωδικών πρόσβασης των διαχειριστών, διασφαλίζοντας ότι ακόμη και σε περίπτωση παραβίασης της βάσης, οι κωδικοί δεν μπορούν να αναγνωστούν. Παράλληλα, γίνεται χρήση των tokens JWT με την βοήθεια το jsonwebtoken που επιτρέπει τη δημιουργία και την επαλήθευση τους, τα οποία χρησιμοποιούνται ώστε να επαληθευτούν οι χρήστες μετά τη σύνδεσή τους. Επιπλέον, η βιβλιοθήκη dotenv διαχειρίζεται μεταβλητές περιβάλλοντος, εξασφαλίζοντας ότι ευαίσθητες πληροφορίες, όπως στοιχεία σύνδεσης με τη βάση ή μυστικά JWT, δεν είναι σκληροκωδικοποιημένες στον πηγαίο κώδικα [31],[33].

Τα routes της εφαρμογής διαχωρίζονται θεματικά, γεγονός που κάνει πιο εύκολη τη συντήρηση και τη δομική καθαρότητα του backend. Το αρχείο routes/rooms.js είναι υπεύθυνο για όλα τα αιτήματα που σχετίζονται με τα δωμάτια. Μέσω της μεθόδου `GET` στο endpoint /rooms, επιστρέφεται λίστα με όλα τα δωμάτια που έχουν καταχωρηθεί στη βάση. Επιπλέον, το endpoint /rooms/:id/images δίνει άδεια ώστε να ανακτηθούν οι εικόνες που συνδέονται με το κάθε δωμάτιο, προσφέροντας στο frontend το απαραίτητο υλικό για την οπτική απεικόνιση του καταλύματος.

Η διαχείριση των κρατήσεων υλοποιείται μέσω του αρχείου routes/bookings.js. Μέσα από τα κατάλληλα endpoints, ο διαχειριστής μπορεί να ανακτήσει λίστα όλων των κρατήσεων με GET /bookings, να δημιουργήσει νέα κράτηση μέσω POST /bookings, να ενημερώσει υπάρχουσα κράτηση με PUT /bookings/:id, να διαγράψει μια κράτηση μέσω DELETE /bookings/:id, καθώς και να τροποποιήσει την κατάστασή της (π.χ. από "pending" σε "approved") μέσω του endpoint PUT /bookings/:id/status. Επίσης, προσφέρεται η δυνατότητα να γίνεται έλεγχος διαθεσιμότητας με βάση συγκεκριμένες ημερομηνίες και τον αριθμό ατόμων που επιλέγει ο χρήστης, μέσω του endpoint GET /bookings/available-rooms, το οποίο κάνει έναν έλεγχο συγκρίνοντας τις ημερομηνίες εισόδου και εξόδου με κρατήσεις που υπάρχουν ήδη στην βάση.

Η επαλήθευση της ταυτότητας των χρηστών γίνεται μέσω των endpoints /login και /register. Ο διαχειριστής μπορεί να συνδεθεί αποστέλλοντας POST αίτημα στο /login, και εφόσον τα διαπιστευτήρια είναι σωστά, λαμβάνει JWT token για την ασφαλή πρόσβαση στο backend. Η

δημιουργία νέου διαχειριστή πραγματοποιείται με αίτημα POST στο /register, με τον κωδικό να αποθηκεύεται κρυπτογραφημένος μέσω bcrypt για μέγιστη ασφάλεια [33].

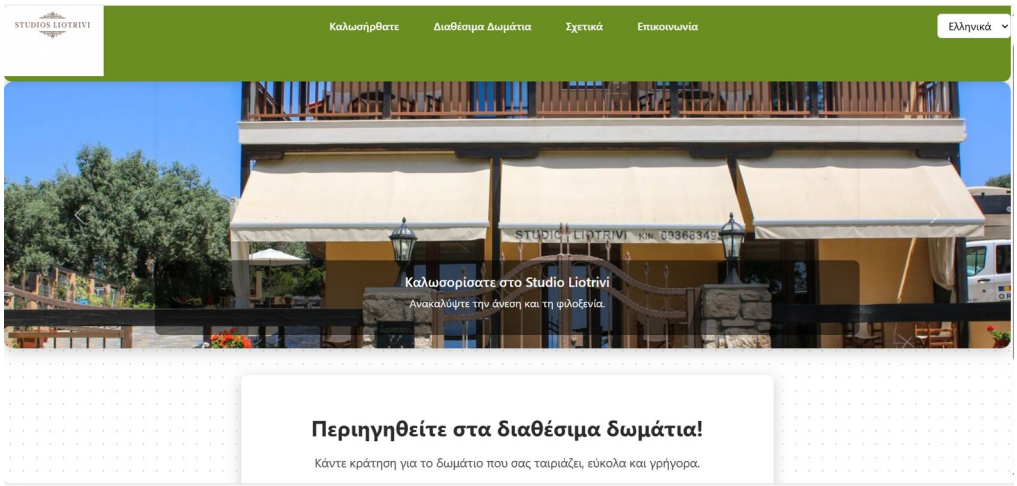
Τέλος, η εφαρμογή περιλαμβάνει και λειτουργικότητα επικοινωνίας μέσω του endpoint /contact, το οποίο υποστηρίζει αποστολή email μέσω POST αιτήματος. Η αποστολή γίνεται με χρήση της βιβλιοθήκης nodemailer, επιτρέποντας στον χρήστη να στείλει μήνυμα προς τον διαχειριστή του καταλύματος, το οποίο φτάνει ως email στην διεύθυνση που έχει οριστεί σαν email διαχειριστή.

Η παραπάνω δομή backend χαρακτηρίζεται από modular προσέγγιση, από υψηλό βαθμό επεκτασιμότητας και χρήση βέλτιστων πρακτικών για την ανάπτυξη σύγχρονων RESTful APIs, παρέχοντας ταυτόχρονα ασφάλεια, ευκολία συντήρησης και δυνατότητα μελλοντικών επεκτάσεων.

5.3 Υλοποίηση Frontend

Η διεπαφή της εφαρμογής δημιουργήθηκε με σκοπό να προσφέρει στον τελικό χρήστη μια απλή, με σαφή και εύκολη εμπειρία πλοήγησης. Για την ανάπτυξή της χρησιμοποιήθηκαν τεχνολογίες HTML, CSS και JavaScript, έχοντας ως γνώμονα την κατανόηση, φροντίζοντας να λειτουργεί καλά και ταυτόχρονα να φαίνεται όμορφη. Το σύνολο του frontend είναι στατικό HTML project και φιλοξενείται τοπικά με χρήση XAMPP, επιτρέποντας την απρόσκοπτη φόρτωση των σελίδων και τη σύνδεσή τους με το backend μέσω HTTP αιτημάτων [34].

Η δομή του frontend περιλαμβάνει βασικές σελίδες που καλύπτουν τις κύριες λειτουργίες του ιστότοπου. Η αρχική σελίδα, index.html, αποτελεί την είσοδο του χρήστη στην εφαρμογή και περιλαμβάνει μερικές απλές πληροφορίες για καλωσόρισμα και ένα μεγάλο κουμπί που βοηθά τον χρήστη να βρεί ένα δωμάτιο (Εικ. 5.2). Η σελίδα rooms.html περιλαμβάνει μια δυναμική φόρμα, μέσω της οποίας ο χρήστης μπορεί να εισαγάγει ημερομηνία άφιξης, αναχώρησης και αριθμό επισκεπτών, ώστε να αναζητήσει διαθέσιμα δωμάτια τα οποία πληρούν τα κριτήρια του (Εικ. 5.3). Στη συνέχεια, η σελίδα booking.html δίνει τη δυνατότητα στον χρήστη να ολοκληρώσει την κράτηση που θέλει, βάζοντας τα απαραίτητα στοιχεία (όνομα, email κ.λπ.) (Εικ. 5.4). Η about.html προσφέρει πληροφορίες για το κατάλυμα, τις παροχές του και συχνές ερωτήσεις, ενώ η contact.html περιλαμβάνει φόρμα επικοινωνίας για την αποστολή μηνύματος προς τον διαχειριστή, η οποία διασυνδέεται με το backend μέσω του endpoint /contact. Η σελίδα room-details.html παρέχει αναλυτικές πληροφορίες για κάθε δωμάτιο του καταλύματος. Η προβολή των δεδομένων γίνεται δυναμικά με χρήση JavaScript, αντλώντας περιγραφές, τιμές, εικόνες και χαρακτηριστικά από το backend, ανάλογα με το δωμάτιο που έχει επιλέξει ο χρήστης από τη σελίδα rooms.html.



Εικόνα 5.2: Αρχική οθόνη της ιστοσελίδας.

Η εφαρμογή λειτουργεί και μεταβάλλεται με τη χρήση του αρχείου script.js, το οποίο πραγματοποιεί κρίσιμες λειτουργίες. Συγκεκριμένα, χειρίζεται την αποστολή αιτημάτων στο endpoint

/bookings/available-rooms, αναλύοντας τα αποτελέσματα και παρουσιάζει στον χρήστη τα δωμάτια που είναι διαθέσιμα με βάση τα κριτήρια που έβαλε. Επίσης, διαθέτει μια λειτουργία για δυναμική gallery ανά δωμάτιο, από την οποία φορτώνονται εικόνες και υπάρχει η δυνατότητα προβολής των εικόνων μέσω lightbox. Η υποβολή των κρατήσεων γίνεται με POST αίτημα στο backend, ενώ προστίθεται και υποστήριξη αλλαγής γλώσσας μέσω dropdown menu. Η επιλογή της γλώσσας αποθηκεύεται τοπικά (με χρήση localStorage) και το περιεχόμενο αντικαθίσταται με βάση προκαθορισμένο λεξικό, προσφέροντας δυο γλώσσες ως εμπειρία χρήσης.

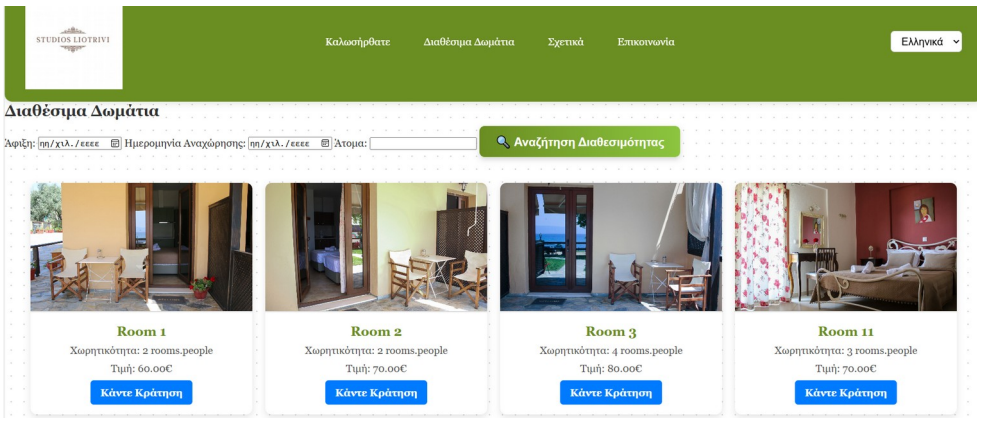
Το αρχείο styles.css φροντίζει ώστε να υπάρχει οπτικά μια συνοχή και να ανταποκρίνεται η διεπαφή σε διαφορετικούς τύπους συσκευών. Όλες οι σελίδες έχουν σχεδιαστεί με responsive layout, προσαρμόζοντας τη δομή τους ανάλογα με την συσκευή, δηλαδή σε οθόνες κινητών και tablets. Υπάρχει επίσης gallery με αυτόματη εναλλαγή εικόνων, κυρίως στη σελίδα about.html, ενισχύοντας τη συνολική εμπειρία από αισθητικής άποψης. Η επιλογή χρωμάτων, γραμματοσειρών και στοιχείων πλοήγησης έχει γίνει με στόχο την ομοιομορφία και την ευκολία στη χρήση, ενώ η πλοήγηση ανάμεσα στις σελίδες είναι άμεση και ελαφριά, με χρήση CSS transitions και minimal σχεδιαστικά πρότυπα.

Η εμπειρία χρήστη (UI/UX) έχει σχεδιαστεί με ιδιαίτερη προσοχή στη λειτουργικότητα και την ευκολία πλοήγησης, ακόμη και από μη εξοικειωμένους χρήστες. Κάθε μια από τις σελίδες έχει συγκεκριμένο σκοπό και δείχνει μόνο τα σημαντικά μέρη που χρειάζονται για τη δουλειά του χρήστη, προκειμένου να μην προκαλεί σύγχυση. Τα στοιχεία που χρησιμοποιούνται τακτικά, όπως το navigation bar και το footer, έχουν κατασκευαστεί με τρόπο που υποστηρίζει συνέπεια και επεκτασιμότητα. Επιπλέον, οι φόρμες και τα dropdown menus έχουν σχεδιαστεί με σαφή τρόπο, ώστε να ενισχύουν τη διαδραστικότητα χωρίς να κάνουν πιο δύσκολη την εμπειρία του χρήστη.

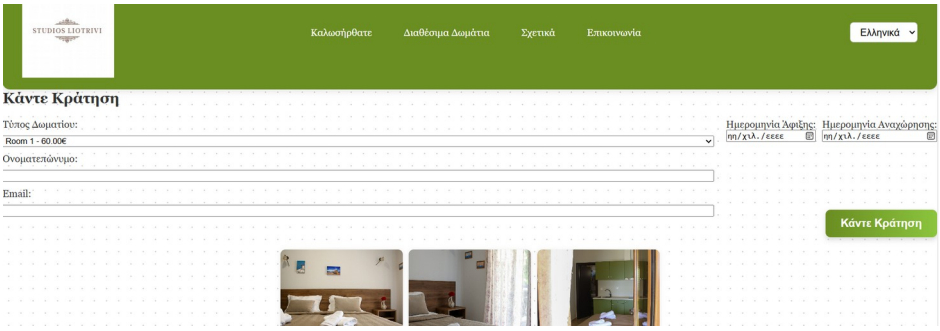
5.4 CRUD Κρατήσεων και Διαχείρισης

Στο πλαίσιο της ανάπτυξης της εφαρμογής, πραγματοποιήθηκαν οι βασικές λειτουργίες CRUD (Create, Read, Update, Delete) στο σύνολό του. Αυτές αποτελούν τον πυρήνα της διαχείρισης των κρατήσεων δωματίων και επιτρέπουν την πλήρη αλληλεπίδραση χρηστών και διαχειριστών με το σύστημα, προσφέροντας με τον τρόπο αυτό μια ολοκληρωμένη εμπειρία λειτουργίας και ελέγχου. Η επικοινωνία μεταξύ frontend και backend πραγματοποιείται μέσω RESTful API endpoints, τα οποία είναι κατάλληλα δομημένα ώστε να εξυπηρετούν διαφορετικές ανάγκες ανάλογα με τον ρόλο του χρήστη [10].

Η λειτουργία για την δημιουργία μιας κράτησης (Create) προσφέρει τη δυνατότητα στον χρήστη να πραγματοποιήσει κράτηση με σχετικά απλό και κατανοητό τρόπο. Μέσα από τη φόρμα που εμφανίζεται στη σελίδα booking.html, ο χρήστης πρέπει να επιλέξει τις ημερομηνίες άφιξης και αναχώρησης, να δηλώσει τον αριθμό των ατόμων για τα οποία θέλει να κάνει την κράτηση και να επιλέξει ένα από τα διαθέσιμα δωμάτια (Εικ.5.3). Στη συνέχεια, συμπληρώνει προσωπικά στοιχεία όπως το ονοματεπώνυμο και το email επικοινωνίας (5.4). Όλα τα δεδομένα αποστέλλονται στο backend με τη μορφή HTTP POST αιτήματος προς το endpoint /bookings, και η κράτηση καταχωρείται στη βάση δεδομένων με προεπιλεγμένη κατάσταση “pending” (σε αναμονή/εκκρεμότητα). Αυτή η διαδικασία διασφαλίζει ότι κάθε νέα κράτηση που έρχεται στο σύστημα, ελέγχεται πρώτα από τον διαχειριστή πριν πραγματοποιηθεί, εξασφαλίζοντας έλεγχο και ασφάλεια [30].

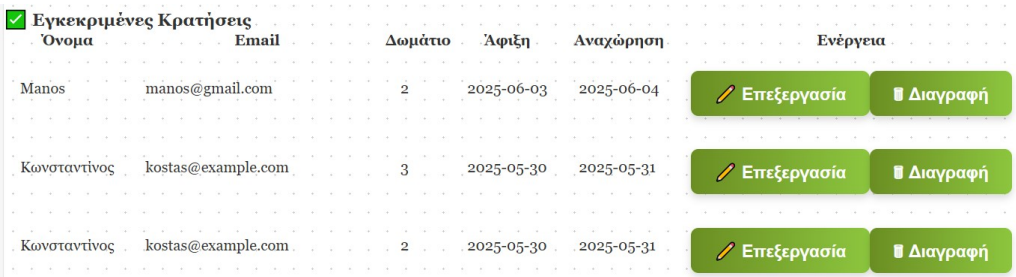


Εικόνα 5.3: Φόρμα αναζήτησης διαθέσιμων δωματίων.



Εικόνα 5.4: Φόρμα κράτησης.

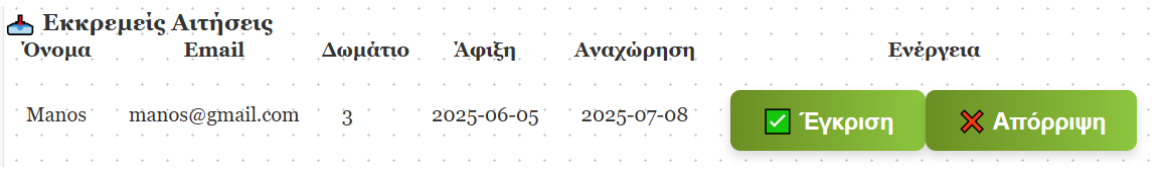
Η λειτουργία ανάγνωσης κρατήσεων (Read) είναι διαθέσιμη μέσω του dashboard του διαχειριστή, που του δίνει την δυνατότητα να δει όλες τις κρατήσεις που έχουν κάνει οι χρήστες στο σύστημα (Εικ 5.5). Με τη βοήθεια ενός HTTP GET αιτήματος προς το endpoint /bookings, εμφανίζονται πληροφορίες όπως το όνομα του πελάτη, η ηλεκτρονική διεύθυνση, οι ημερομηνίες διαμονής, το επιλεγμένο δωμάτιο, καθώς και η τρέχουσα κατάσταση της κράτησης. Η παρουσίαση γίνεται σε μορφή λίστας, διευκολύνοντας την ταχεία επισκόπηση και διαχείριση του ιστορικού των αιτήσεων.



Εικόνα 5.5: Παράδειγμα λίστας κρατήσεων.

Η ενημέρωση μιας κράτησης (Update) αφορά κυρίως το να αλλάξει η κατάσταση της κράτησης από τον διαχειριστή, όταν αυτή πρέπει να εγκριθεί ή να απορριφθεί (Εικ. 5.6). Η συγκεκριμένη διαδικασία πραγματοποιείται μέσω PUT αιτήματος στο endpoint /bookings/:id/status, όπου το :id αντιστοιχεί στο μοναδικό αναγνωριστικό της κράτησης. Ο διαχειριστής έχει την δυνατότητα να επιλέξει μεταξύ της κατάστασης “approved” (εγκεκριμένη) και “rejected” (απορριφθείσα), ανάλογα με τη διαθεσιμότητα των δωματίων, την εγκυρότητα των στοιχείων ή τους εσωτερικούς κανόνες του καταλύματος. Η

δυνατότητα αυτή αποτελεί κρίσιμο εργαλείο για την αξιοπιστία του συστήματος, αφού εξασφαλίζει ότι καμία κράτηση δεν επιβεβαιώνεται αυτόματα χωρίς επανεξέταση.



Εικόνα 5.6: Παράδειγμα εκκρεμών Αιτήσεων.

Η λειτουργία διαγραφής κράτησης (Delete) δίνει την δυνατότητα για οριστική αφαίρεση μιας κράτησης από τη βάση δεδομένων. Ο διαχειριστής μπορεί να διαγράψει εγγραφές που θεωρούνται μη έγκυρες, ακυρώθηκαν από τον πελάτη ή εισήχθησαν κατά λάθος. Η διαδικασία υλοποιείται με HTTP DELETE αίτημα στο endpoint /bookings/:id και εμφανίζει ένα μήνυμα στην οθόνη για να είναι σίγουρο ότι θα γίνει πραγματικά αυτή η αλλαγή, ώστε να μην χαθούν κατά λάθος δεδομένα [1].

Οι τέσσερις αυτές λειτουργίες, μαζί, σχηματίζουν τον πλήρη κύκλο ζωής μιας κράτησης: από τη δημιουργία της με υποβολή στοιχείων, την προβολή και επανεξέταση, την ενδεχόμενη επεξεργασία της κατάστασης και, εφόσον απαιτείται, τη διαγραφή της. Η παρουσία τους βοηθάει ώστε τα πάντα να ελέγχονται εύκολα και διασφαλίζει ότι τόσο οι επισκέπτες όσο και ο διαχειριστής είναι ευχαριστημένοι. Κάνει επίσης ολόκληρο το σύστημα να λειτουργεί καλά και να είναι αξιόπιστο.

5.5 Δυναμική Αναζήτηση Δωματίων με Φίλτρα

Η λειτουργία αναζήτησης δωματίου είναι πολύ σημαντική επειδή βοηθά τους επισκέπτες να βρίσκουν δωμάτια που είναι ανοιχτά, διαθέσιμα για τις ημερομηνίες που επιθυμούν και κατάλληλα για τον αριθμό των ατόμων που έχουν επιλέξει. Αντί να εμφανίζεται απλώς μια σταθερή λίστα πραγμάτων, η αναζήτηση μπορεί να αλλάξει και να βρει αυτό που ψάχνει ο χρήστης με βάση αυτό που έχει επιλέξει. Λειτουργεί άμεσα, καθιστώντας το πιο εύκολο στη χρήση.

διαδικασία ξεκινάει από μια διαδραστική φόρμα στην ιστοσελίδα, μέσω της οποίας ο επισκέπτης εισάγει την ημερομηνία άφιξης (check-in), την ημερομηνία αναχώρησης (check-out) και τον αριθμό των ατόμων που θέλουν να πάνε στο κατάλυμα (Εικ.5.7). Με την υποβολή της φόρμας, αποστέλλεται αίτημα τύπου GET στο backend, στο endpoint /bookings/available-rooms, μαζί με τις τιμές που εισήγαγε ο χρήστης ως query parameters. Ένα τυπικό παράδειγμα αιτήματος είναι το εξής:

/bookings/available-rooms?check_in=2025-06-01&check_out=2025-06-03&people=2

Το backend λαμβάνει τα δεδομένα και πραγματοποιεί SQL ερώτημα προς τη βάση δεδομένων, λαμβάνοντας υπόψη δύο πολύ σημαντικά κριτήρια. Αρχικά, ελέγχεται αν η χωρητικότητα (capacity) κάθε δωματίου είναι αρκετή για να φιλοξενήσει τον αριθμό επισκεπτών που έχει επιλέξει ο χρήστης. Στην συνέχεια, γίνεται έλεγχος για πιθανές κρατήσεις του δωματίου τις ίδιες ημερομηνίες που έχουν ήδη καταχωρηθεί για το συγκεκριμένο χρονικό διάστημα και δεν έχουν απορριφθεί.

```
31  <section id="rooms" class="rooms-section">
32    <h2 data-i18n="rooms.title">Διαθέσιμα Δωμάτια</h2>
33      <label for="check-in" data-i18n="rooms.checkin">Άφιξη:</label>
34      <input type="date" id="check-in" name="check-in" required>
35
36      <label for="check-out" data-i18n="rooms.checkout">Αναχώρηση:</label>
37      <input type="date" id="check-out" name="check-out" required>
38
39      <label for="guests" data-i18n="rooms.guests">Άτομα:</label>
40      <input type="number" id="guests" name="guests" min="1" required>
41
42      <button id="search-btn" type="button" data-i18n="rooms.search">🔍 Αναζήτηση Διαθεσιμότητας</button>
```

Εικόνα 5.7: Φόρμα φίλτρων αναζήτησης δωματίων

Εικόνα 5.8: SQL κώδικας

```
228       SELECT *
229       FROM rooms
230       WHERE capacity >= $1
231         AND id NOT IN (
232           SELECT room_id
233           FROM bookings
234           WHERE status <> 'rejected'
235             AND check_in < $3
236             AND check_out > $2
237         )
238     `;
```

Στην εικόνα 5.8 παρουσιάζεται η SQL εντολή που εκτελείται στο backend. Στο συγκεκριμένο ερώτημα, το `$1` αντιπροσωπεύει τον αριθμό ατόμων που έχει δηλώσει ο χρήστης, ενώ τα `$2` και `$3` αντιστοιχούν στην ημερομηνία άφιξης και αναχώρησης αντίστοιχα. Η λογική του ερωτήματος εξασφαλίζει ότι εξαιρούνται από τα αποτελέσματα όσα δωμάτια είναι ήδη κρατημένα για την επιλεγμένη περίοδο.

Η απόκριση του backend αποστέλλεται στο frontend σε μορφή JSON, περιλαμβάνοντας τα δωμάτια που πληρούν τα κριτήρια αναζήτησης. Η JavaScript που εκτελείται στην πλευρά του πελάτη αναλαμβάνει να εμφανίσει δυναμικά τα αποτελέσματα, χωρίς να χρειάζεται να γίνει επαναφόρτωση της σελίδας. Η ενημέρωση της σελίδας γίνεται αμέσως, καθιστώντας εύκολη και διασκεδαστική την εξερεύνηση και τη χρήση της εφαρμογής.

Στις περιπτώσεις όπου δεν υπάρχουν διαθέσιμα δωμάτια για τα συγκεκριμένα φίλτρα που επιλέχθηκαν, το σύστημα εμφανίζει κατάλληλο μήνυμα ειδοποίησης, όπως "Δεν υπάρχουν διαθέσιμα δωμάτια", στην προσπάθεια να διατηρηθεί η διαφάνεια και η άμεση ενημέρωση προς τον χρήστη. Αυτή η δυνατότητα κάνει την εφαρμογή πιο εύκολη στη χρήση και βοηθά τους επισκέπτες να κάνουν κράτηση ενός δωματίου γρήγορα και με σιγουριά.

5.6 Σύστημα Authentication και Authorization

Για να είναι τα στοιχεία όλων ασφαλή και να είναι επιβεβαιωμένο ότι μόνο οι κατάλληλοι άνθρωποι μπορούν να συνδεθούν, η εφαρμογή η εφαρμογή ενσωματώνει σύγχρονους και ενημερωμένους μηχανισμούς αυθεντικοποίησης (authentication) και εξουσιοδότησης (authorization). Αυτοί οι μηχανισμοί συνεργάζονται έτσι ώστε οι άνθρωποι να μπορούν να εισέλθουν με ασφάλεια στο σύστημα χωρίς ανησυχίες. Υπάρχει διαχωρισμός ρόλων (όπως επισκέπτης και διαχειριστής), και περιορίζεται η πρόσβαση σε ευαίσθητα σημεία του backend, ανάλογα με τα δικαιώματα που διαθέτει κάθε χρήστης.

Όταν γίνεται εγγραφή ενός νέου διαχειριστή, ο κωδικός πρόσβασης που επιλέγει, δεν αποθηκεύεται σε απλή μορφή στη βάση δεδομένων. Αντίθετα, χρησιμοποιείται η βιβλιοθήκη bcrypt, η οποία εφαρμόζει αλγόριθμο hashing με χρήση «αλατιού» (salt), δηλαδή τυχαίων δεδομένων που αυξάνουν την πολυπλοκότητα του κωδικού. Αυτό σημαίνει ότι ακόμα κι αν κάποιος παραβιάσει τη λίστα κωδικών πρόσβασης, δεν μπορεί να βρει εύκολα τους πραγματικούς κωδικούς πρόσβασης. Έτσι διατηρούνται όλα τα στοιχεία πιο ασφαλή. Η εντολή hashing εφαρμόζεται πριν από την αποθήκευση και διασφαλίζει ότι κάθε κωδικός είναι κωδικοποιημένος με μοναδικό τρόπο, ακόμη και αν δύο χρήστες επιλέξουν τον ίδιο κωδικό [23].

Μετά από επιτυχή είσοδο ενός διαχειριστή στο σύστημα (login), ο διακομιστής δημιουργεί ένα JWT (JSON Web Token), το οποίο έχει απλές πληροφορίες όπως το ειδικό αναγνωριστικό του χρήστη και ποια εργασία ή τμήμα έχει στο σύστημα. Το token αυτό υπογράφεται με μυστικό κλειδί (JWT_SECRET) και αποστέλλεται πίσω στον client, όπου αποθηκεύεται τοπικά - συνήθως στο localStorage ή στο sessionStorage. Από εκεί και πέρα, το token συνοδεύει κάθε επόμενο αίτημα προς το backend, κυρίως σε προστατευμένα endpoints, πράγμα το οποίο βοηθά τον διακομιστή να ελέγξει αμέσως εάν επιτρέπεται σε κάποιον να συνδεθεί και αν του επιτρέπεται να κάνει ορισμένες λειτουργίες.

Αν και το βασικό σύστημα δημιουργίας, υπογραφής και αποθήκευσης των tokens έχει ήδη ολοκληρωθεί, για την πλήρη προστασία συγκεκριμένων διαδρομών (όπως η πρόσβαση στον πίνακα κρατήσεων ή η έγκριση αιτημάτων) απαιτείται η υλοποίηση middleware. Αυτό το ενδιάμεσο λογισμικό αναλαμβάνει να ελέγχει την εγκυρότητα του token πριν δοθεί η άδεια για πρόσβαση σε δεδομένα που προορίζονται αποκλειστικά για εξουσιοδοτημένους χρήστες ή χρήστες συγκεκριμένου ρόλου (π.χ. μόνο διαχειριστές). Αυτήν τη στιγμή, ο τρόπος κατασκευής της εφαρμογής διευκολύνει την προσθήκη ενός ειδικού role-based middleware που μπορεί να δώσει σε διαφορετικούς ανθρώπους διαφορετικά επίπεδα άδειας σε μεταγενέστερο χρόνο.

Η χρήση JWT προσφέρει σημαντικά πλεονεκτήματα, καθώς καθιστά την εφαρμογή stateless - δεν απαιτεί διατήρηση session state στον διακομιστή - και δίνει την δυνατότητα στον χρήστη να περιηγείται στην εφαρμογή χωρίς να απαιτείται συνεχής επιβεβαίωση ταυτότητας. Επιπλέον, η διαχείριση ρόλων που περιλαμβάνεται στο token δημιουργεί τη δυνατότητα επέκτασης της εφαρμογής σε ένα πλήρες admin panel, όπου θα μπορούν να εφαρμοστούν διαφορετικά επίπεδα πρόσβασης και διαβαθμισμένα δικαιώματα ανά χρήστη ή ομάδα χρηστών.

Αυτή η ρύθμιση ασφαλείας κάνει την εφαρμογή πιο αξιόπιστη και ασφαλή. Ακολουθεί σημαντικούς κανόνες για να διατηρεί τα δεδομένα ασφαλή. Πράγμα το οποίο την κάνει αρκετά καλή ώστε να μπορεί να χρησιμοποιηθεί όχι μόνο για ακαδημαϊκή παρουσίαση αλλά και σε πραγματικά συστήματα αργότερα.

5.7 Επικοινωνία μέσω Email με χρήση Nodemailer

Στο πλαίσιο της εφαρμογής δημιουργήθηκε μία φόρμα επικοινωνίας, η οποία επιτρέπει σε κάθε επισκέπτη του ιστότοπου να στείλει απευθείας μήνυμα στον διαχειριστή/ ιδιοκτήτη του καταλύματος. Με τη λειτουργία αυτή καλύπτεται η ανάγκη για άμεση και εύκολη επικοινωνία μεταξύ πελάτη και επιχείρησης, χωρίς να είναι απαραίτητη η τηλεφωνική επαφή ή η χρήση εξωτερικών εφαρμογών αποστολής email. Με την προσθήκη μιας εύχρηστης φόρμας επικοινωνίας, ο ιστότοπος προσθέτει έναν χρήσιμο τρόπο ώστε να πραγματοποιείται η επικοινωνία, να μοιράζονται τα μηνύματα και να διατηρούνται όλα μέσα στον ιστότοπο.

Η τεχνική υλοποίηση βασίζεται στη χρήση της βιβλιοθήκης Nodemailer, η οποία αποτελεί μία από τις πιο διαδεδομένες λύσεις για αποστολή email από περιβάλλον Node.js. Η βιβλιοθήκη παρέχει διεπαφές για τη διαμόρφωση SMTP συνδέσεων και την αποστολή email με διάφορους τρόπους. Στην παρούσα υλοποίηση χρησιμοποιείται ο SMTP server της υπηρεσίας Gmail, με τη σύνδεση να πραγματοποιείται μέσω προσωρινού app password, σύμφωνα με τα πρόσφατα πρότυπα ασφαλείας της Google που καταργούν την άμεση χρήση username/password για εφαρμογές τρίτων [24],[36].

Η διαδικασία αποστολής ενός email περιλαμβάνει διάφορα βήματα. Αρχικά, ο χρήστης μεταβαίνει στη σελίδα "Επικοινωνία", όπου του εμφανίζεται μία φόρμα με πεδία για το ονοματεπώνυμο, τη διεύθυνση email και το μήνυμα που επιθυμεί να αποστείλει. Η φόρμα είναι απλή και εύκολη στη χρήση, ώστε όλοι να μπορούν να την κατανοήσουν και να τη συμπληρώσουν εύκολα, ακόμα και άτομα λιγότερο εξοικειωμένα..

Με την υποβολή της φόρμας, η JavaScript στον client αποστέλλει ένα HTTP POST αίτημα προς το backend, και συγκεκριμένα στο endpoint /contact. Το αίτημα περιλαμβάνει τα στοιχεία του χρήστη σε μορφή JSON, τα οποία μεταβιβάζονται στον server για περαιτέρω επεξεργασία.

Στην πλευρά του backend, έχει δημιουργηθεί ειδική συνάρτηση που αξιοποιεί το nodemailer.createTransport() για τη διαμόρφωση ενός SMTP transporter. Στη συνέχεια, δημιουργείται το περιεχόμενο του email σε HTML μορφή, ώστε το μήνυμα να αποστέλλεται με δομημένο και ευανάγνωστο τρόπο. Τα στοιχεία του αποστολέα εμφανίζονται ξεκάθαρα, ενώ το κείμενο του μηνύματος προσαρμόζεται αυτόματα στο σώμα του email.

Το email αποστέλλεται στη διεύθυνση ηλεκτρονικού ταχυδρομείου του διαχειριστή, η οποία έχει οριστεί ως προορισμός στο backend. Εφόσον η αποστολή είναι επιτυχής, ο διακομιστής επιστρέφει ένα μήνυμα επιτυχίας προς το frontend (success: true), ενώ σε περίπτωση αποτυχίας ενεργοποιείται μηχανισμός διαχείρισης σφαλμάτων και εμφανίζεται σχετικό μήνυμα στον χρήστη.

Ένα τυπικό παράδειγμα του email που λαμβάνει ο διαχειριστής θα μπορούσε να έχει την εξής μορφή:

> **Αποστολέας:** Γιάννης Παπαδόπουλος (giannis@example.com)
> Καλησπέρα, θα ήθελα πληροφορίες για διαμονή από 10 έως 15 Αυγούστου.

Η χρήση της μορφοποίησης HTML βοηθά στο να φαίνεται πιο ωραίο και πιο ευανάγνωστο το μήνυμα. Βοηθά επίσης τον διαχειρστή να το καταλάβει καλύτερα, ακόμα κι αν λαμβάνει πολλά μηνύματα κάθε μέρα. Επιπλέον, προσφέρει τη δυνατότητα στους πελάτες να επικοινωνούν με τον ιδιοκτήτη πιο εύκολα, ποιο άμεσα και κάνει την εφαρμογή να φαίνεται πιο επαγγελματική και πιο εύχρηστη.

5.8 Open Source Δομή και Δημοσίευση στο GitHub

Το τελευταίο βήμα που αφορά στο λογισμικό που αναπτύχθηκε στην εργασία αυτή, ήταν ο δωρεάν διαμοιρασμός του στο διαδίκτυο. Αυτό σημαίνει ότι ο καθένας έχει τη δυνατότητα να δει πώς λειτουργεί η εφαρμογή, να τη χρησιμοποιήσει, να την τροποποιήσει ελεύθερα και να την αξιοποιήσει για μάθηση ή βελτίωση, χωρίς να απαιτείται καμία οικονομική επιβάρυνση. Η επιλογή του GitHub ως πλατφόρμα φιλοξενίας κρίθηκε ως η πλέον κατάλληλη, λόγω της δημοφιλίας της, της ευκολίας χρήσης της αλλά και της δυνατότητας τεκμηρίωσης, ελέγχου εκδόσεων και διαχείρισης συνεργασιών.

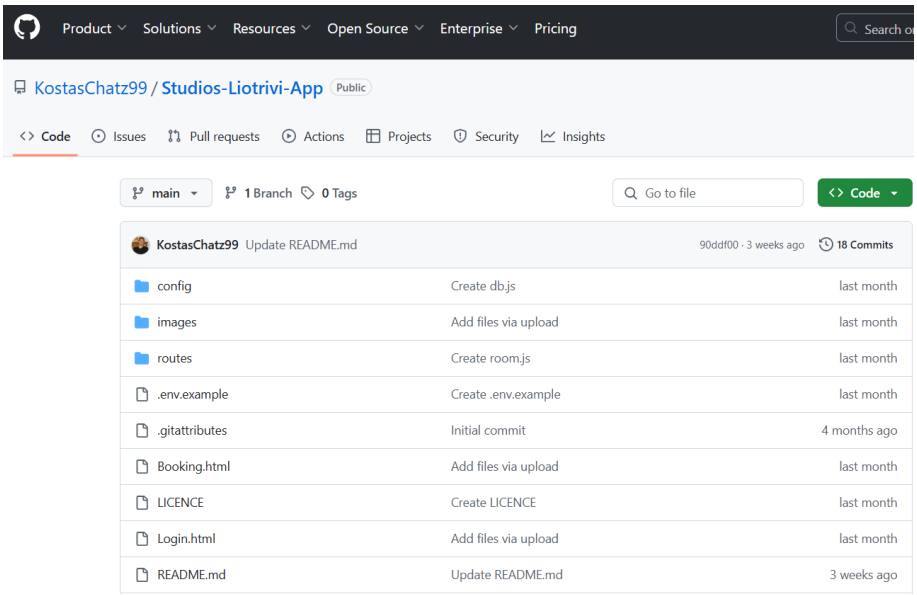
το GitHub δημιουργήθηκε δημόσιο αποθετήριο (repository), στο οποίο είναι ανεβασμένος ο πλήρης πηγαίος κώδικας της εφαρμογής — τόσο του backend όσο και του frontend. Η δομή του αποθετηρίου σχεδιάστηκε με βάση τις βέλτιστες πρακτικές του ανοικτού λογισμικού, ώστε να είναι κατανοητό, καθαρό και εύκολο για άλλα άτομα να κάνουν αλλαγές αν το επιθυμούν.

Κεντρικό ρόλο στην παρουσίαση της εφαρμογής εντός του repository διαδραματίζει το αρχείο README.md, το οποίο περιλαμβάνει λεπτομερείς οδηγίες για την εγκατάσταση της σε τοπικό περιβάλλον, την τεχνική περιγραφή των βασικών χαρακτηριστικών, τις τεχνολογίες που χρησιμοποιούνται (Node.js, Express, PostgreSQL, HTML/CSS/JS) και οδηγίες για πιθανή συμβολή (contribution) από άλλους χρήστες ή προγραμματιστές. Η δομή του README είναι φτιαγμένη με τέτοιο τρόπο ώστε ακόμα και κάποιος που δεν έχει ξανασχοληθεί να μπορεί να το εγκαταστήσει εύκολα ακολουθώντας μερικά απλά βήματα [17],[37].

Για την αδειοδότηση του προγράμματος επιλέχθηκε η άδεια MIT (MIT License), που σημαίνει ότι δίνεται η δυνατότητα να χρησιμοποιηθεί, να αντιγραφεί, να γίνουν αλλαγές, ακόμη και να πουληθεί το λογισμικό εάν το επιθυμεί ο χρήστης, αρκεί να γίνει αναφορά στο ποιος το δημιούργησε αρχικά. Η συγκεκριμένη άδεια επιλέχθηκε λόγω της ευρείας αποδοχής της από την κοινότητα ανοιχτού λογισμικού και της ευελιξίας που προσφέρει στους μελλοντικούς χρήστες [18].

Στο repository περιλαμβάνεται και το αρχείο .gitignore, το οποίο έχει διαμορφωθεί κατάλληλα ώστε να εξαιρούνται αρχεία και φάκελοι που δεν είναι απαραίτητο να ανέβουν στο GitHub. Ανάμεσα σε αυτά περιλαμβάνονται φάκελοι εξαρτήσεων (node_modules/), αρχεία περιβάλλοντος (.env) που περιέχουν ευαίσθητα δεδομένα όπως κωδικοί και κλειδιά API, καθώς και προσωρινά αρχεία που δημιουργούνται από περιβάλλοντα ανάπτυξης ή το λειτουργικό σύστημα (π.χ. .DS_Store, αρχεία log κ.λπ.).

Ο κώδικας είναι ανοιχτός, όπως φαίνεται και στην εικόνα 5.9, ώστε ένας χρήστης να μπορεί να τον δει, να τον χρησιμοποιήσει, να τον αλλάξει ή να τον συμπεριλάβει στη δική του εργασία αν είναι απαραίτητο. Αυτός ο τρόπος εργασίας βοηθά όλους να μοιράζονται εύκολα αυτά που γνωρίζουν, διευκολύνει τους φοιτητές και τους προγραμματιστές να συνεργάζονται και δίνει στην κοινότητα μια ισχυρή βάση για να συνεχίσει να βελτιώνει το πρόγραμμα. Η διεύθυνση του αποθετηρίου στο GitHub είναι:
https://github.com/KostasChatz99/Studios-Liotrivi-App



Εικόνα 5.9: Github repository.

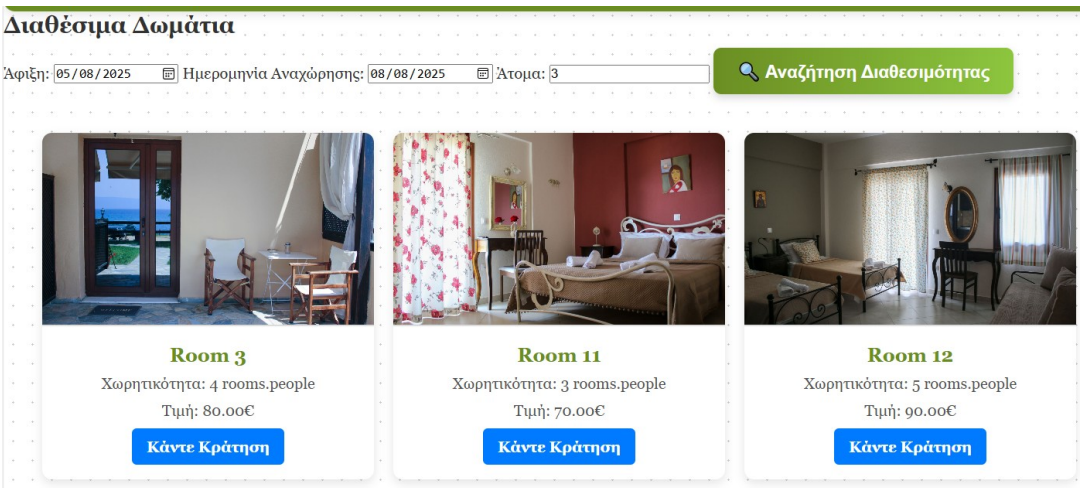
6 Δοκιμές & Αξιολόγηση

Για να γίνει σωστή αξιολόγηση του τρόπου λειτουργίας της εφαρμογής έγιναν δοκιμές με βάση γεγονότα που είναι πιθανό να συμβούν, καλύπτοντας τόσο την εμπειρία του επισκέπτη όσο και τις δυνατότητες που έχει ο διαχειριστής. Μέσα από αυτά τα σενάρια επιβεβαιώθηκε ότι η εφαρμογή είναι σταθερή, εύκολη στην χρήση και λειτουργεί σωστά, με τον τρόπο που σχεδιάστηκε και υλοποιήθηκε.

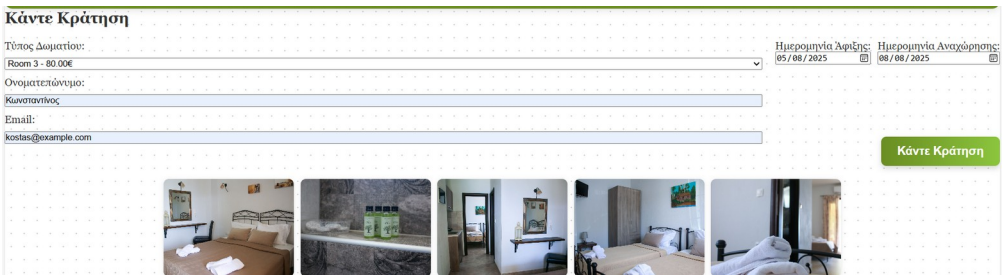
➤ Περίπτωση 1: Κράτηση από επισκέπτη

Ένα από τα πιο βασικά σενάρια αφορά την κράτηση ενός δωματίου από έναν απλό χρήστη. Στο συγκεκριμένο παράδειγμα, ένας επισκέπτης επιθυμεί να κάνει κράτηση για δύο άτομα, για το διάστημα από 5 έως 8 Αυγούστου. Η αλληλεπίδραση με την εφαρμογή εξελίσσεται ως εξής:

- Ο χρήστης μεταβαίνει στη σελίδα «Διαθέσιμα Δωμάτια».
- Συμπληρώνει τη φόρμα αναζήτησης (Εικ. 6.1), επιλέγοντας:
    - ημερομηνία άφιξης (check-in),
    - ημερομηνία αναχώρησης (check-out),
    - και αριθμό ατόμων.
- Η εφαρμογή αποστέλλει HTTP GET αίτημα στο endpoint /bookings/available-rooms [38].
- Το backend επεξεργάζεται το αίτημα και επιστρέφει λίστα με δωμάτια που είναι διαθέσιμα για τα κριτήρια που τέθηκαν.
- Τα δωμάτια εμφανίζονται δυναμικά στη σελίδα, με την προβολή φωτογραφιών, τιμών καθώς και ένα κουμπί επιλογής για κράτηση.
- Ο επισκέπτης επιλέγει ένα δωμάτιο και συμπληρώνει τα προσωπικά του στοιχεία στη φόρμα κράτησης (Εικ.6.2).
- Η φόρμα υποβάλλεται και αποστέλλεται αίτημα POST στο backend.
- Το σύστημα καταχωρεί τη νέα κράτηση με αρχική κατάσταση pending και εμφανίζει μήνυμα επιτυχίας.



Εικόνα 6.1: Αναζήτηση διαθέσιμων δωματίων από επισκέπτη



Εικόνα 6.2: Συμπλήρωση και υποβολή φόρμας κράτησης

➤ Περίπτωση 2: Επεξεργασία κράτησης από διαχειριστή

Η δεύτερη περίπτωση αφορά τη διαχείριση κρατήσεων από την πλευρά του υπεύθυνου διαχειριστή του καταλύματος. Ο διαχειριστής, μέσα από το περιβάλλον διαχείρισης, έχει πρόσβαση στα αιτήματα και έχει τη δυνατότητα να ελέγχει την κατάσταση κάθε κράτησης.

- Αρχικά, πραγματοποιεί είσοδο στο σύστημα μέσω της φόρμας σύνδεσης (login).
- Μεταβαίνει στην ενότητα «Πίνακας Κρατήσεων».
- Εκεί εμφανίζεται λίστα με όλες τις εκκρεμείς κρατήσεις (status: pending)(Εικ.6.3).
- Ο διαχειριστής επιλέγει αν θα εγκρίνει ή θα απορρίψει κάθε κράτηση, πατώντας το αντίστοιχο κουμπί.
- Η επιλογή ενημερώνει αυτόματα τη βάση δεδομένων με το νέο status της κράτησης.
- Το σύστημα επιστρέφει άμεσο feedback για την ενέργεια που πραγματοποιήθηκε.

Εικόνα 6.3: Επεξεργασία κατάστασης κράτησης (έγκριση ή απόρριψη)

Οι παραπάνω περιπτώσεις επιβεβαιώνουν τη σωστή ροή των δεδομένων, τη συνεργασία frontend και backend, καθώς και τη διάκριση των ρόλων στην εφαρμογή. Το σύστημα λειτουργεί με ομαλό και γρήγορο τρόπο ακόμα και όταν χρησιμοποιείται σε πραγματικό χρόνο, δείχνοντας ότι είναι ισχυρό και αξιόπιστο.

6.1 Έλεγχος Κρατήσεων και Συγκρούσεων

Κατά τη φάση υλοποίησης του backend, σημαντικό ρόλο έπαιξε το να είναι βέβαιο ότι σε κανέναν δεν θα επιτρεπόταν να κάνει κράτηση κατά λάθος για το ίδιο δωμάτιο δύο φορές ή να γίνει διπλοκράτηση από δύο διαφορετικά άτομα για το ίδιο δωμάτιο κατά τις ίδιες ημερομηνίες. Το σύστημα έπρεπε να εξασφαλίζει ότι για κάθε αίτηση κράτησης, η διαθεσιμότητα του εκάστοτε δωματίου θα ελέγχεται με αυστηρότητα, ώστε να αποφεύγονται φαινόμενα που μπορεί να προκαλέσουν την δυσαρέσκεια των πελατών αλλά και τη μη σωστή λειτουργία του καταλύματος.

➤ Σκοπός του Ελέγχου

Ο μηχανισμός επαλήθευσης κρατήσεων σχεδιάστηκε ώστε να διασφαλίζει τα εξής:

- Καμία κράτηση δεν μπορεί να πραγματοποιηθεί για ημερομηνίες που επικαλύπτονται με ήδη υπάρχουσα κράτηση η οποία έχει status “approved” ή “pending”.

- Οι κρατήσεις με status “rejected” αγνοούνται κατά τον έλεγχο διαθεσιμότητας, επιτρέποντας τη δημιουργία νέας κράτησης για το ίδιο χρονικό διάστημα.

➤ **Διαδικασία Δοκιμής και Περιπτώσεις**

Για να ελεγχθεί αν λειτουργεί σωστά ο μηχανισμός, σχεδιάστηκαν ρεαλιστικά σενάρια χρήσης:

Περίπτωση 1 – Απόπειρα κράτησης σε ήδη δεσμευμένο διάστημα

Ένας επισκέπτης θέλει να κάνει κράτηση για ένα δωμάτιο, επιλέγοντας ημερομηνίες που συμπίπτουν με προηγούμενη κράτηση που έχει status: approved ή pending.
Αναμενόμενο αποτέλεσμα: Το σύστημα απορρίπτει την αίτηση και εμφανίζει μήνυμα όπως: *«Το δωμάτιο δεν είναι διαθέσιμο για τις επιλεγμένες ημερομηνίες.»*

Περίπτωση 2 – Κράτηση σε διάστημα για το οποίο έχει γίνει αίτημα κράτησης το οποίο έχει απορριφθεί

Ο χρήστης επιχειρεί να κάνει κράτηση για ημερομηνίες που συμπίπτουν με κράτηση που έχει ήδη απορριφθεί (status: rejected).
Αναμενόμενο αποτέλεσμα: Το σύστημα επιτρέπει την καταχώρηση της νέας κράτησης, καθώς δεν θεωρεί τη συγκεκριμένη εγγραφή ενεργή.

Περίπτωση 3 – Επικαλυπτόμενες κρατήσεις σε εγκεκριμένο διάστημα

Εάν δύο αιτήματα κράτησης επικαλύπτονται με κράτηση που έχει γίνει αποδεκτή στο ίδιο δωμάτιο, ένα από τα δύο αιτήματα απορρίπτεται έχοντας ως βάση το ποιο πραγματοποιήθηκε πρώτο.

➤ **Υλοποίηση Ελέγχου**

Η επαλήθευση των ημερομηνιών στο backend υλοποιείται μέσω SQL ερωτήματος, το οποίο ελέγχει αν υπάρχει “σύγκρουση” ημερομηνιών με βάση τη λογική:

```
sql
Αντιγραφή Επεξεργασία
check_in < νέα_ημερομηνία_check_out AND check_out > νέα_ημερομηνία_check_in
```

Η συνθήκη αυτή επιβεβαιώνει ότι η νέα κράτηση που πραγματοποιήθηκε, επικαλύπτεται με τουλάχιστον μία ήδη υπάρχουσα, αν και μόνο αν η ημερομηνία άφιξης της νέας κράτησης είναι μικρότερη από την ημερομηνία αναχώρησης της παλιάς, και η ημερομηνία αναχώρησης της νέας κράτησης είναι μεγαλύτερη από την ημερομηνία άφιξης της παλιάς. Το ερώτημα αυτό εφαρμόζεται σε συνδυασμό με έλεγχο κατάστασης (status <> 'rejected'), ώστε να εξαιρούνται απορριφθείσες εγγραφές από τον μηχανισμό απόρριψης.

Ο έλεγχος “σύγκρουσης” κρατήσεων αποδείχθηκε αποτελεσματικός και αξιόπιστος κατά τις δοκιμές. Η εφαρμογή ανταποκρίνεται με ακρίβεια σε αιτήματα που θα μπορούσαν να δημιουργήσουν αλληλοεπικαλύψεις. Αυτό βοηθά στην αποφυγή σύγχυσης για τον υπεύθυνο του καταλύματος και διασφαλίζει ότι οι χρήστες λαμβάνουν τις σωστές απαντήσεις. Ο μηχανισμός ενσωματώνεται ομαλά στη λογική του backend, ενώ είναι σχεδιασμένος ώστε να υποστηρίξει επεκτασιμότητα στο μέλλον - για παράδειγμα, υπολογισμό χρόνων check-in/check-out σε επίπεδο ωρών, ή υποστήριξη δυναμικών περιορισμών κρατήσεων ανά τύπο δωματίου. Η σωστή λειτουργία του τμήματος αυτού είναι πολύ σημαντική γιατί βοηθά να διασφαλιστεί ότι ολόκληρο το σύστημα κρατήσεων παρέχει σωστές και αξιόπιστες πληροφορίες.

6.2 Έλεγχος Ορθής Λειτουργίας API

Για τη διασφάλιση της σωστής λειτουργίας του backend και των API endpoints της εφαρμογής, πραγματοποιήθηκαν στοχευμένες δοκιμές μέσω του εργαλείου Postman. Το Postman επιτρέπει την αποστολή αιτημάτων HTTP προς τον διακομιστή και την καταγραφή των αντίστοιχων αποκρίσεων,

γεγονός που το καθιστά πολύτιμο εργαλείο για την αξιολόγηση της συμπεριφοράς μιας εφαρμογής σε περιβάλλον ανάπτυξης.

Κατά τις δοκιμές, ελέγχθηκαν οι βασικές διαδρομές που σχετίζονται με δωμάτια και κρατήσεις. Αρχικά, με τη διαδρομή GET /rooms επιβεβαιώθηκε ότι το σύστημα επιστρέφει όλα τα διαθέσιμα δωμάτια σε έγκυρη JSON μορφή, η οποία περιλαμβάνει όλες τις σημαντικές λεπτομέρειες για κάθε δωμάτιο. Παράλληλα, η διαδρομή GET /bookings επέστρεψε τις κρατήσεις με σωστά οργανωμένα πεδία, όπως για παράδειγμα τις ημερομηνίες check-in/check-out και την αντίστοιχη κατάσταση της εκάστοτε κράτησης (status).

Η διαδρομή POST /bookings δοκιμάστηκε τόσο με έγκυρα όσο και με σκόπιμα εσφαλμένα δεδομένα, προκειμένου να διαπιστωθεί η δυνατότητα του συστήματος να ανιχνεύει και να αποτρέπει λανθασμένες εισαγωγές. Αντίστοιχα, μέσω του PUT /bookings/:id/status, δοκιμάστηκε η ενημέρωση της κατάστασης κράτησης (π.χ. από "pending" σε "approved" ή "rejected") και επιβεβαιώθηκε ότι το σύστημα ανταποκρίνεται κατάλληλα.

Ιδιαίτερη έμφαση δόθηκε και στη λειτουργία δυναμικής αναζήτησης δωματίων, μέσω του endpoint GET /bookings/available-rooms, όπου οι παράμετροι ημερομηνίας και αριθμού ατόμων αποστέλλονται ως query parameters. Το σύστημα παρουσίασε μόνο τα δωμάτια που ταίριαζαν με αυτό που ήταν το ζητούμενο, και το πέτυχε, ταιριάζοντας ακριβώς τις πληροφορίες από τη βάση δεδομένων.

Κατά τη διάρκεια των δοκιμών, καταγράφηκαν τα εξής:

- Όλα τα δεδομένα επέστρεφαν σε σωστή JSON μορφή, με δομή που ανταποκρίνεται πλήρως στους πίνακες της βάσης.
- Οι ημερομηνίες κρατήσεων αποθηκεύονταν και εμφανίζονταν στη μορφή `YYYY-MM-DD`, χωρίς ασυνέπειες.
- Το σύστημα ανίχνευε επικάλυψη κρατήσεων και, σε τέτοιες περιπτώσεις, επέστρεφε HTTP status 409 Conflict, όπως προβλέπεται.
- Σε περιπτώσεις όπου τα αιτήματα περιείχαν ελλιπή ή μη έγκυρα δεδομένα, το backend ανταποκρινόταν με κατάλληλα HTTP status codes (400 Bad Request ή 500 Internal Server Error) και επεξηγηματικά μηνύματα σφάλματος, συμβάλλοντας στη διάγνωση του προβλήματος.

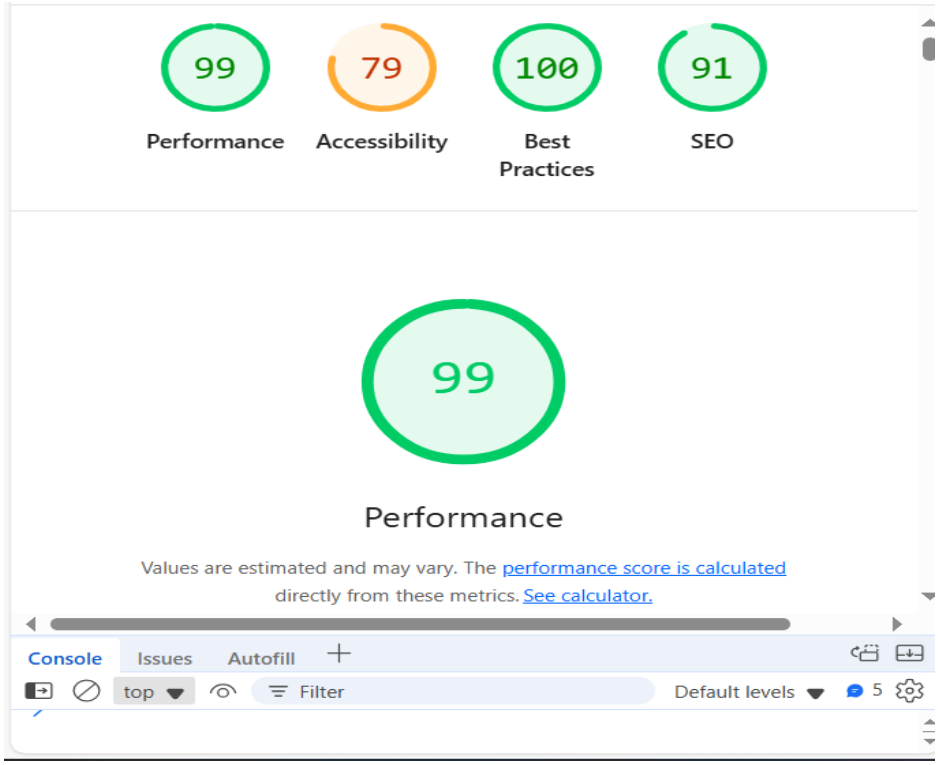
Ο τρόπος που λειτουργούσε το API ήταν σταθερός και αξιόπιστος. Οι αποκρίσεις του backend ήταν προβλέψιμες και τεκμηριωμένες πράγμα το οποίο διευκόλυνε τον έλεγχο για σφάλματα και τη διόρθωσή τους αργότερα. Επίσης, μπορεί να επεκταθεί αν χρειαστεί χωρίς προβλήματα.

6.3 Αξιολόγηση της εφαρμογής

Για την αξιολόγηση της ταχύτητας φόρτωσης, τις προσβασιμότητας της εφαρμογής και της συμμόρφωσης με βέλτιστες πρακτικές,, χρησιμοποιήθηκε το εργαλείο Lighthouse που διατίθεται δωρεάν απο τον browser Google Chrome. Η μέτρηση πραγματοποιήθηκε στην τελική έκδοση της εφαρμογής, μέσω της προσομοίωσης συνθηκών πραγματικής χρήσης.

Τα αποτελέσματα της ανάλυσης έδειξαν πολύ υψηλές επιδόσεις, με Performance: 99, Accessibility: 79, Best Practices: 100 και SEO: 91. Η υψηλή βαθμολογία στο Performance και Best Practices επιβεβαιώνει την άρτια υλοποίηση και βελτιστοποίηση του frontend, ενώ οι μικρές αποκλίσεις στην προσβασιμότητα οφείλονται κυρίως σε λεπτομέρειες σήμανσης HTML που μπορούν να διορθωθούν σε επόμενες εκδόσεις. Στην Εικόνα 6.4 παρουσιάζεται η αναφορά που προέκυψε από το εργαλείο Lighthouse, η οποία αποτυπώνει τα συνολικά σκορ ανά κατηγορία.

Εικόνα 6.4: Αξιολόγηση της εφαρμογής



7 Συμπεράσματα & Μελλοντικές Επεκτάσεις

7.1 Απολογισμός Εργασίας

Η παρούσα πτυχιακή εργασία είχε ως βασικό στόχο τη σχεδίαση και υλοποίηση μιας πλήρως λειτουργικής διαδικτυακής εφαρμογής με αντικείμενο τη διαχείριση κρατήσεων για ένα τουριστικό κατάλυμα. Η εφαρμογή αυτή είναι χρήσιμη τόσο για τους επισκέπτες, δηλαδή τα άτομα που θέλουν να μείνουν κάπου κατά τη διάρκεια των διακοπών τους, όσο και τους διαχειριστές, δηλαδή τα άτομα που διευθύνουν και διαχειρίζονται το κατάλυμα. Στόχος της είναι να διευκολυνθεί η διαδικασία πραγματοποίησης μιας κράτησης ενός δωματίου, να βελτιωθεί ο τρόπος λειτουργίας της και να διασφαλιστεί ότι και οι δύο πλευρές, και οι πελάτες και οι διαχειριστές, έχουν μια καλή εμπειρία από τη χρήση της.

Από την αρχή της ανάπτυξης τέθηκαν συγκεκριμένες απαιτήσεις όπως ότι θα έπρεπε να είναι εύχρηστη, να μπορεί να μεταβληθεί και να αναπτυχθεί αργότερα και να μπορεί να κάνει διαφορετικά πράγματα, αν χρειαστεί. Για τον λόγο αυτό, επιλέχθηκαν σύγχρονες και ευρέως χρησιμοποιούμενες τεχνολογίες. Το backend αναπτύχθηκε με Node.js και Express.js, προσφέροντας RESTful διαδρομές για την επεξεργασία των κρατήσεων, τη διαχείριση των δωματίων, την επικοινωνία και την αυθεντικοποίηση χρηστών. Η αποθήκευση των δεδομένων υλοποιήθηκε μέσω PostgreSQL, με σχεσιακή δομή που περιλαμβάνει πίνακες για δωμάτια, κρατήσεις, χρήστες και εικόνες.

Το frontend σχεδιάστηκε με HTML, CSS και JavaScript, δημιουργώντας μια διαδραστική διεπαφή που επιτρέπει την αναζήτηση δωματίων με τη χρήση φίλτρων που έχουν ως βάση τι ημερομηνίες και τον αριθμό των επισκεπτών, καθώς και την ολοκλήρωση κρατήσεων με εύκολο και κατανοητό τρόπο. Η εφαρμογή είναι πλήρως responsive, καθώς προσφέρει προσαρμοσμένη εμπειρία τόσο σε υπολογιστές όσο και σε κινητές συσκευές.

Η λειτουργικότητα περιλαμβάνει πλήρη υποστήριξη CRUD για κρατήσεις, με δυνατότητα επεξεργασίας ή ακύρωσης από τον διαχειριστή. Οι κρατήσεις οπτικοποιούνται με τρόπο που διευκολύνει τη διαχείρισή τους, ενώ οι έλεγχοι επικαλυπτόμενων ημερομηνιών διασφαλίζουν την ακρίβεια της διαθεσιμότητας. Παράλληλα, ενσωματώθηκε και μία φόρμα επικοινωνίας η οποία επιτρέπει την αποστολή email από τον χρήστη/επισκέπτη προς τον διαχειριστή μέσω του πακέτου Nodemailer.

Η ασφάλεια των χρηστών διασφαλίζεται με την αποθήκευση των κωδικών πρόσβασης σε κρυπτογραφημένη μορφή (bcrypt) και τη χρήση JSON Web Tokens (JWT) για την αυθεντικοποίηση και εξουσιοδότηση, επιτρέποντας τον διαχωρισμό πρόσβασης ανά ρόλο.

Επιπλέον, η εφαρμογή δημοσιεύθηκε σε δημόσιο αποθετήριο στο GitHub, υπό την άδεια MIT. Η τεκμηρίωση περιλαμβάνει οδηγίες εγκατάστασης, περιγραφή της αρχιτεκτονικής και σαφή δομή που επιτρέπει τη συμβολή τρίτων (open source), ενισχύοντας έτσι τη δυνατότητα μελλοντικής αξιοποίησης ή εξέλιξης της από άλλους φοιτητές ή επαγγελματίες.

Αυτή η εφαρμογή παρέχει ένα σταθερό και εύχρηστο σύστημα για τη δημιουργία ενός νέου εργαλείου κρατήσεων. Μπορεί να βελτιωθεί περισσότερο στο μέλλον, να προστεθούν νέες δυνατότητες ή ακόμη και να χρησιμοποιηθεί σε άλλα καταλύματα.

7.2 Περιορισμοί της Υλοποίησης

Παρόλο που η εφαρμογή κατασκευάστηκε με επιτυχία και λειτουργεί καλά, καλύπτοντας τις βασικές απαιτήσεις λειτουργίας ενός συστήματος κρατήσεων, κατά τη διάρκεια της ανάπτυξης έγιναν αντιληπτοί κάποιοι περιορισμοί, τόσο σε τεχνικό όσο και σε λειτουργικό επίπεδο. Οι περιορισμοί αυτοί δεν αναιρούν τη χρηστικότητα και την αξιοπιστία της εφαρμογή, αναδεικνύουν σημεία που θα μπορούσαν να βελτιωθούν στο μέλλον ή να αποτελέσουν βάση για περαιτέρω ανάπτυξη.

Ένας από τους βασικότερους περιορισμούς είναι η απουσία συστήματος πληρωμών. Η παρούσα έκδοση δεν υποστηρίζει καμία ενσωμάτωση με υπηρεσίες όπως κάρτες, PayPal ή τραπεζικές

διεπαφές, γεγονός που κάνει το σύστημα μη αυτόνομο ως προς την εμπορική ολοκλήρωση της κράτησης. Όλες οι κρατήσεις πραγματοποιούνται χωρίς χρηματική συναλλαγή, και η τελική επιβεβαίωση πραγματοποιείται χειροκίνητα από τον διαχειριστή.

Ένας ακόμη περιορισμός αφορά την υποστήριξη για κινητές συσκευές. Αν και η εφαρμογή εμφανίζεται σωστά σε οθόνες διαφορετικών μεγεθών, δεν έχει ενσωματωθεί κάποιο πλήρες responsive design framework (όπως Bootstrap ή Tailwind CSS). Αυτό περιορίζει τη συνολική αισθητική αρτιότητα και τη βελτιστοποιημένη εμπειρία χρήστη, ειδικά σε φορητές συσκευές.

Τέλος, αν και έχουν εφαρμοστεί βασικές τεχνικές ασφαλείας όπως η χρήση JWT για αυθεντικοποίηση και bcrypt για την κρυπτογράφηση των κωδικών πρόσβασης, το επίπεδο ασφαλείας είναι σε τέτοιο στάδιο ανεπτυγμένο που καλύπτει μόνο τα πολύ βασικά κομμάτια της εφαρμογής. Αυτό σημαίνει ότι δεν έχουν ενσωματωθεί προηγμένες πρακτικές, όπως περιορισμός αριθμού αιτημάτων (rate limiting), προσωρινός αποκλεισμός λογαριασμού μετά από επαναλαμβανόμενες αποτυχημένες συνδέσεις (account lockout), ή χρήση HTTPS σε παραγωγικό περιβάλλον.

7.3 Προτάσεις για Μελλοντική Εξέλιξη

Η παρούσα εφαρμογή, όπως έχουμε ήδη αναφέρει παραπάνω, παρέχει ένα ολοκληρωμένο σύστημα διαχείρισης κρατήσεων για καταλύματα, καλύπτοντας βασικές λειτουργίες τόσο στο backend όσο και στο frontend. Ωστόσο, η εφαρμογή διαθέτει σημαντικές δυνατότητες επέκτασης, οι οποίες μπορούν να βελτιώσουν τη λειτουργικότητα της, να ενισχύσουν την εμπειρία του χρήστη και να αυξήσουν την αποδοτικότητα της πλατφόρμας. Ακολουθούν προτάσεις για μελλοντικές επεκτάσεις και βελτιώσεις:

1. **Ενσωμάτωση Ηλεκτρονικών Πληρωμών**

   Η προσθήκη υποστήριξης για διαδικτυακές πληρωμές μέσω υπηρεσιών όπως το Stripe ή το PayPal θα επιτρέψει την πλήρη αυτοματοποίηση της διαδικασίας κράτησης. Οι επισκέπτες θα μπορούν να ολοκληρώνουν την επιλογή δωματίου και την πληρωμή εντός της ίδιας πλατφόρμας, προσδίδοντας με τον τρόπο αυτό επαγγελματισμό και αξιοπιστία στην εφαρμογή και καθιστώντας την κατάλληλη για χρήση σε πραγματικές συνθήκες.

2. **Responsive Σχεδίαση για Κινητές Συσκευές**

   Η υιοθέτηση σύγχρονων CSS frameworks, όπως το Tailwind CSS ή το Bootstrap, θα βελτιώσει την προσαρμοστικότητα της διεπαφής σε διαφορετικά μεγέθη οθόνης. Ένα πλήρως responsive περιβάλλον χρήσης εξασφαλίζει ομαλή και ευχάριστη πλοήγηση τόσο σε κινητά τηλέφωνα όσο και σε tablets, ενισχύοντας το πόσο προσβάσιμη και εύχρηστη είναι η εφαρμογή.

3. **Επέκταση του Μηχανισμού Ταυτοποίησης και Διαχείρισης Ρόλων**

   Η τρέχουσα υλοποίηση υποστηρίζει βασικό έλεγχο ρόλων μεταξύ χρηστών και διαχειριστών. Σε μεταγενέστερο χρόνο, μπορεί να ενσωματωθούν περισσότερα επίπεδα, όπως receptionist ή manager, με διαφορετικά δικαιώματα πρόσβασης και λειτουργίες, όπως η επεξεργασία κρατήσεων, η δημιουργία αναφορών και η διαχείριση περιεχομένου, επιτρέποντας έτσι πιο λεπτομερή έλεγχο και οργανωμένη διαχείριση.

4. **Υποστήριξη Πολυγλωσσικότητας**

   Η εφαρμογή μπορεί να εμπλουτιστεί με σύστημα διαχείρισης πολυγλωσσικού περιεχομένου βασισμένο σε αρχεία μετάφρασης (π.χ. JSON, i18n βιβλιοθήκες). Επιπλέον, μπορεί να προστεθεί λειτουργία αυτόματης ανίχνευσης γλώσσας μέσω του browser ή επιλογής γλώσσας από τις ρυθμίσεις, καθιστώντας την πιο προσβάσιμη σε διεθνές κοινό.

5. **Ενσωμάτωση Συστήματος Αξιολογήσεων και Κριτικών**

Η δυνατότητα υποβολής αξιολογήσεων και κριτικών από τους χρήστες μετά τη διαμονή τους, με κείμενο και βαθμολογία, θα συμβάλλει σημαντικά στην αύξηση της διαφάνειας και της εμπιστοσύνης. Το σύστημα θα περιλαμβάνει αποθήκευση των σχολίων στο backend, μηχανισμούς επαλήθευσης εγκυρότητας και εμφάνιση των κριτικών στο frontend με τρόπο φιλικό προς τον χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Mozilla Developer Network, “MDN Web Docs,” [Online]. Available: https://developer.mozilla.org/en-US/. (Accessed: 05/07/2025).

[2] W3Schools, “W3Schools Online Web Tutorials,” [Online]. Available: https://www.w3schools.com/. (Accessed: 30/08/2025).

[3] PostgreSQL Global Development Group, “PostgreSQL: The world's most advanced open source relational database,” [Online]. Available: https://www.postgresql.org/. (Accessed: 23/07/2025).

[4] Node.js Foundation, “Node.js,” [Online]. Available: https://nodejs.org. (Accessed: 15/05/2025).

[5] V8 Engine, “V8 JavaScript Engine,” [Online]. Available: https://v8.dev/. (Accessed: 15/05/2025).

[6] ExpressJS, “Express - Node.js web application framework,” [Online]. Available: https://expressjs.com/. (Accessed: 22/05/2025).

[7] PostgreSQL Documentation, “Official PostgreSQL Documentation,” [Online]. Available: https://www.postgresql.org/docs/. (Accessed: 23/05/2025).

[8] PostgreSQL Tutorial, “Learn PostgreSQL with examples,” [Online]. Available: https://www.postgresqltutorial.com/. (Accessed: 05/06/2025).

[9] Node-Postgres, “node-postgres,” [Online]. Available: https://node-postgres.com/. (Accessed: 05/07/2025).

[10] RESTfulAPI.net, “RESTful API Design and Best Practices,” [Online]. Available: https://restfulapi.net/. (Accessed: 05/06/2025).

[11] JSON.org, “Introducing JSON,” [Online]. Available: https://www.json.org/json-en.html. (Accessed: 12/06/2025).

[12] DcodeIO, “bcrypt.js,” GitHub Repository. [Online]. Available: https://github.com/dcodeIO/bcrypt.js. (Accessed: 15/07/2025).

[13] Kelektiv, “node.bcrypt.js,” GitHub Repository. [Online]. Available: https://github.com/kelektiv/node.bcrypt.js. (Accessed: 15/07/2025).

[14] JWT.io, “Introduction to JSON Web Tokens,” [Online]. Available: https://jwt.io/introduction/. (Accessed: 25/08/2025).

[15] OWASP, “Authentication Cheat Sheet,” [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html. (Accessed: 23/06/2025).

[16] Auth0, “jsonwebtoken,” GitHub Repository. [Online]. Available: https://github.com/auth0/node-jsonwebtoken. (Accessed: 15/07/2025).

[17] GitHub Docs, “GitHub Help Documentation,” [Online]. Available: https://docs.github.com/en. (Accessed: 14/06/2025).

[18] ChooseALicense, “MIT License,” [Online]. Available: https://choosealicense.com/licenses/mit/. (Accessed: 04/08/2025).

[19] Open Source Initiative, “The MIT License | Open Source Initiative,” [Online]. Available: https://opensource.org/licenses/MIT. (Accessed: 04/08/2025).

[20] Google Developers, “Responsive Web Design Basics,” [Online]. Available: https://web.dev/responsive-web-design-basics/. (Accessed: 18/06/2025).

[21] Node.js, “Node.js Documentation,” [Online]. Available: https://nodejs.org/en/docs/. (Accessed: 05/07/2025).

[22] Express, “Express.js Documentation,” [Online]. Available: https://expressjs.com/. (Accessed: 16/06/2025).

[23] bcrypt, “bcrypt npm package,” [Online]. Available: https://www.npmjs.com/package/bcrypt. (Accessed: 18/07/2025).

[24] Nodemailer, “Nodemailer Docs,” [Online]. Available: https://nodemailer.com/about/. (Accessed: 18/07/2025).

[25] PostgreSQL, “SQL Subqueries and Filtering,” [Online]. Available: https://www.postgresql.org/docs/current/functions-subquery.html. (Accessed: 25/07/2025).

[26] Palo Alto Networks, “What is a Data Flow Diagram (DFD)?” [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/data-flow-diagram. (Accessed: 23/06/2025).

[27] GeeksForGeeks, “Levels in Data Flow Diagrams (DFD),” [Online]. Available: https://www.geeksforgeeks.org/software-engineering/levels-in-data-flow-diagrams-dfd/. (Accessed: 25/06/2025).

[28] PostgreSQL Tutorial, “Design Tables in PostgreSQL,” [Online]. Available: https://www.postgresqltutorial.com/postgresql-create-table/. (Accessed: 25/06/2025).

[29] Lucidchart, “What is an Entity Relationship Diagram (ERD)?,” [Online]. Available: https://www.lucidchart.com/pages/er-diagrams. (Accessed: 27/06/2025).

[30] W3Schools, “SQL CRUD Operations,” [Online]. Available: https://www.w3schools.com/sql/sql_crud.asp. (Accessed: 30/06/2025).

[31] dotenv, “dotenv npm package,” [Online]. Available: https://www.npmjs.com/package/dotenv. (Accessed: 30/06/2025).

[32] Microsoft, “Visual Studio Code,” [Online]. Available: https://code.visualstudio.com/. (Accessed: 03/07/2025).

[33] Auth0, “Role-based Access Control,” [Online]. Available: https://auth0.com/docs/authorization/rbac. (Accessed: 05/07/2025).

[34] Apache Friends, “XAMPP,” [Online]. Available: https://www.apachefriends.org/index.html. (Accessed: 13/07/2025).

[35] pgAdmin, “pgAdmin,” [Online]. Available: https://www.pgadmin.org/. (Accessed: 13/07/2025).

[36] Google, “Gmail SMTP Setup,” [Online]. Available: https://support.google.com/mail/answer/7126229?hl=en. (Accessed: 13/07/2025).

[37] makeareadme.com, “README Best Practices,” [Online]. Available: https://www.makeareadme.com/. (Accessed: 15/07/2025).

[38] Stack Overflow, “Query Parameters in REST,” [Online]. Available: https://stackoverflow.com/questions/978061/http-get-with-request-body. (Accessed: 20/07/2025).

[39] Git SCM, “Git Documentation,” [Online]. Available: https://git-scm.com/doc. (Accessed: 26/07/2025).

[40] Mozilla Developer Network (MDN), “JavaScript DOM,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model. (Accessed: 26/07/2025).