ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΔΙΑΔΙΚΤΥΑΚΟ ΣΥΣΤΗΜΑ ΑΝΑΖΗΤΗΣΗΣ ΚΑΙ ΚΡΑΤΗΣΕΩΝ ΞΕΝΟΔΟΧΕΙΑΚΩΝ ΚΑΤΑΛΥΜΑΤΩΝ»

Των φοιτητών:
Τσιόκα Ελένη (ΑΜ: 164767)
Νίττα Ουρανία (ΑΜ: 144375)

Επιβλέπων
Τεκτονίδης Δημήτριος
Διδάσκων Καθηγητής

Ημερομηνία: 10/09/2024

Τίτλος Δ.Ε.: Διαδικτυακό Σύστημα Αναζήτησης και Κρατήσεων Ξενοδοχειακών Καταλυμάτων

Κωδικός Δ.Ε.: 22155

Ονοματεπώνυμα φοιτητών: Τσιόκα Ελένη, Νίττα Ουρανία
Ονοματεπώνυμο εισηγητή: Τεκτονίδης Δημήτριος

Ημερομηνία ανάληψης Δ.Ε.: 24/01/2022
Ημερομηνία περάτωσης Δ.Ε.: 10/09/2024

Βεβαιώνουμε ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Σε μια εποχή που η τεχνολογία αναπτύσσεται με ιλιγγιώδη ταχύτητα, πολλοί τομείς έχουν επηρεαστεί καθοριστικά, όπως η υγεία, η εκπαίδευση, η οικονομία, ο τουρισμός κλπ. Ο τομέας του τουρισμού αποτελεί έναν από τους βασικότερους τομείς της οικονομίας για κάθε χώρα, και φυσικά αυτό ισχύει κατά μεγάλο ποσοστό και για την Ελλάδα, αφού στηρίζεται στον τουρισμό της μεγάλο μέρος της οικονομίας της χώρας μας. Η χρήση της τεχνολογίας που συνεχώς αναπτύσσεται, είναι ένα πολύ σημαντικό εργαλείο στο κομμάτι του τουρισμού, και τόσο οι τουριστικοί προορισμοί όσο και οι τουριστικές και ξενοδοχειακές επιχειρήσεις αναζητούν τρόπους και μεθόδους μέσω νέων τεχνολογιών για να βελτιώσουν την ανταγωνιστικότητά τους.

Επιπρόσθετα, και από την πλευρά της ζήτησης, ο πελάτης/καταναλωτής που πια έχει αποκτήσει γνώσεις στις νέες τεχνολογίες γίνεται πιο απαιτητικός και αναζητά τέτοιου είδους ευέλικτα και εξειδικευμένα μέσα, που βοηθούν στην καλύτερη εξυπηρέτησή του.

Στόχος λοιπόν της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής ξενοδοχειακών κρατήσεων, με περιβάλλον φιλικό, εύκολο και απλό στη χρήση, τόσο από την πλευρά του εκάστοτε ξενοδοχείου όσο και από την πλευρά του χρήστη/πελάτη, Πράγμα που θεωρούμε σημαντικό και χρήσιμο για τη λειτουργία κάθε ξενοδοχειακής μονάδας.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως στόχο τη δημιουργία και ανάπτυξη μιας διαδικτυακής εφαρμογής διαχείρισης ξενοδοχειακών κρατήσεων, χρησιμοποιώντας σύγχρονες τεχνολογίες και κατάλληλα εργαλεία για την ανάπτυξη μια εύχρηστης εφαρμογής.

Αρχικά, για το περιβάλλον που αλληλοεπιδρά ο χρήστης, δηλαδή το Front – End κομμάτι, χρησιμοποιείται η βιβλιοθήκη React της Javascript και το εργαλείο Vite, τα οποία βοηθούν στη δημιουργία όμορφων διεπαφών χρήστη. Για τη λειτουργικότητα της εφαρμογής μας, δηλαδή το Back – End κομμάτι, χρησιμοποιείται το framework Node της Javascript, καθώς και το framework Express του Node framework. Τέλος, για την αποθήκευση των δεδομένων της εφαρμογής μας χρησιμοποιείται η βάση δεδομένων MongoDB ανοιχτού κώδικα.

Η συγκεκριμένη εφαρμογή απευθύνεται σε ξενοδοχειακές μονάδες, που αναζητούν τρόπους και μεθόδους για την βέλτιστη διαχείριση των κρατήσεών τους, καθώς και σε απλούς χρήστες που αναζητούν έναν εύκολο και γρήγορο τρόπο να πραγματοποιήσουν την κράτησή τους.

Λέξεις – Κλειδιά: Προγραμματισμός, Κράτηση, Διαμονή, Ξενοδοχείο, React, Node, MongoDB.

Online Hotel Accommodation Search and Booking System

ELENI TSIOKA, OURANIA NITTA

Abstract

This thesis aims to create and develop a web application for hotel reservation management, utilizing modern technologies and suitable tools to build a user-friendly application. Initially, for the user interface (Front-End), the React library of JavaScript and the Vite tool are used to create aesthetically pleasing user interfaces.

For the functionality of our application, i.e., the Back-End, the Node framework of JavaScript is employed, along with the Express framework of the Node framework. Finally, for storing the data of our application, the open-source MongoDB database is used.

This specific application is designed for hotel units seeking optimal ways and methods for managing their reservations. It also caters to ordinary users who are looking for an easy and quick way to make their reservations.

Keywords: Programming, Reservation, Accommodation, Hotel, React, Node, MongoDB.

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τον επιβλέποντα καθηγητή μας, τον κύριο Τεκτονίδη Δημήτριο, για τη βοήθεια που μας παρείχε και την εμπιστοσύνη που μας έδειξε, αναλαμβάνοντας την επίβλεψη της εν λόγω πτυχιακής εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο:Εισαγωγή

Η ηλεκτρονική κράτηση αναφέρεται στη διαδικασία κατά την οποία οι χρήστες μπορούν να κάνουν κρατήσεις υπηρεσιών, όπως ξενοδοχείων, αεροπορικών εισιτηρίων, αυτοκινήτων κ.ά., μέσω διαδικτύου ή ηλεκτρονικών εφαρμογών. Η διαδικασία αυτή επιτρέπει στους χρήστες να κάνουν κρατήσεις άνετα και γρήγορα, αποφεύγοντας την ανάγκη να επικοινωνήσουν απευθείας με τον πάροχο της υπηρεσίας.

Η διαδικασία της κράτησης των πελατών έχει αλλάξει σε τεράστιο βαθμό τις τελευταίες δεκαετίες. Από τις χειροκίνητες εγγραφές σε βιβλία, έχουμε περάσει σε προηγμένες ψηφιακές πλατφόρμες, οι οποίες έχουν προσαρμοστεί για να ανταποκρίνονται στις συνεχώς αυξανόμενες προσδοκίες του σύγχρονου καταναλωτή. Αρχικά, η δυνατότητα online κράτησης αποτελούσε μια καινοτομία για τις επιχειρήσεις. Σήμερα, είναι απαραίτητη για τη διατήρηση της ανταγωνιστικότητας και την αποτελεσματική λειτουργία στον τομέα του τουρισμού. Αυτά τα συστήματα βελτιώνουν τη διαχείριση των κρατήσεων, ενισχύουν τη διατήρηση των πελατών, ενώ προσφέρουν εξατομικευμένες και ευέλικτες επιλογές.

Κατά συνέπεια, και με γνώμονα την ζήτηση στην αγορά και την τεχνολογική εξέλιξη που έχει επηρεάσει και τον τουρισμό σε μεγάλο βαθμό, έχει δημιουργηθεί η ανάγκη ανάπτυξης ειδικών διαδικτυακών εφαρμογών, εύκολων ως προς τη χρήση και την διαχείρισή τους από την εκάστοτε επιχείρηση ή τον εκάστοτε χρήστη.

Στη παρούσα εργασία επικεντρωθήκαμε στο κομμάτι των ξενοδοχειακών κρατήσεων. Μια ξενοδοχειακή επιχείρηση για να επιβιώσει σε μια ανταγωνιστική αγορά χρειάζεται τουλάχιστον δύο πράγματα: να έχει μια διαδικτυακή παρουσία που διευκολύνει τη διαδικασία κράτησης και να προσφέρει εξαιρετική εξυπηρέτηση πελατών, έτσι ώστε οι επισκέπτες που επιστρέφουν στην περιοχή να επιλέγουν ξανά το εκάστοτε κατάλυμα. Η αυτοματοποίηση των καθημερινών λειτουργιών και των διοικητικών εργασιών αποτελεί σημαντικό μέρος της επίτευξης ικανοποίησης των πελατών, καθώς βοηθά στην παροχή αξιόπιστης και ποιοτικής εξυπηρέτησης κάθε φορά.

Ένα ξενοδοχείο είναι ένα σύνθετο σύστημα που περιλαμβάνει τις δραστηριότητες πολλών τμημάτων, και κάθε λειτουργία πρέπει να παρακολουθείται. Για τον σκοπό αυτό, οι ξενοδόχοι χρησιμοποιούν διάφορα εργαλεία, όπως υπολογιστικά φύλλα, έντυπα και ενοποιημένα συστήματα διαχείρισης καταλυμάτων (PMS). Το PMS (Property Management System) είναι ένα λογισμικό που διευκολύνει τη διαχείριση κρατήσεων και τις διοικητικές εργασίες ενός ξενοδοχείου. Οι πιο σημαντικές λειτουργίες περιλαμβάνουν τις εργασίες της υποδοχής, τις κρατήσεις, την καθαριότητα, τη διαχείριση τιμών και πληρότητας, και την επεξεργασία πληρωμών. Παρόλο που το λογισμικό PMS κυρίως ελέγχει τις κρατήσεις και τις χρηματοοικονομικές συναλλαγές, μπορεί επίσης να επιτρέψει τη διαχείριση της καθαριότητας καθώς και την διαχείριση ανθρώπινων πόρων. Γενικά, το PMS διευκολύνει τις κύριες διαδικασίες σε ένα ξενοδοχείο που σχετίζονται με εσωτερικές και εξωτερικές λειτουργίες.[1]

Τα συστήματα διαχείρισης κράτησης σε καταλύματα επίσης μπορούν να χωριστούν σε δύο βασικές κατηγορίες ως προς την λειτουργικότητα τους, σε αυτά που έχουν έναν ρόλο μηχανής αναζήτησης ανάμεσα από μια ποικιλία καταλυμάτων και σε αυτά που μπορούν να ενσωματωθούν στον ιστότοπο του εκάστοτε καταλύματος. Στην πρώτη κατηγορία τα συστήματα αυτά βοηθούν τον χρήστη να επιλέξει το καταλληλότερο κατάλυμα με βάση τις ανάγκες του μέσα από μια διαδικασία επιβολής φίλτρων στην αναζήτηση, όπως για παράδειγμα οι παροχές που επιθυμεί ο χρήστης. Ενώ στην δεύτερη κατηγορία ανήκουν τα συστήματα τα οποία διευκολύνουν τον χρήστη να πραγματοποιήσει την κράτησή του μέσω της σελίδας που έχει επιλέξει και την ευκαιρία να αναζητήσει συγκεκριμένο δωμάτιο που θα καλύψει τις ανάγκες του. Τα συστήματα και των δύο κατηγοριών είναι ευρέως διαδεδομένα και χρησιμοποιούνται

καθημερινά από εκατομμύρια χρήστες.

1.1 Αντικείμενο Εργασίας

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο της τη δημιουργία και ανάπτυξη μιας μορφής ενός τέτοιου συστήματος (PMS) που ωστόσο περιορίζεται στις βασικές χρήσεις δηλαδή την δημιουργία κράτησης δωματίων και διαχείρισης αυτών σε μια ξενοδοχειακή μονάδα, δηλαδή ανήκει στην δεύτερη από τις προαναφερθείσες κατηγορίες. Οι χρήσεις της εφαρμογής χωρίζονται σε τέσσερις κατηγορίες, του διαχειριστή (admin) της εφαρμογής, του ιδιοκτήτη (owner) του ξενοδοχείου, του υπεύθυνου κρατήσεων (employee) και του χρήστη - πελάτη (unauthorized user).

Ο διαχειριστής (admin):

- Είναι υπεύθυνος για την σωστή λειτουργία της εφαρμογής.
- Ελέγχει τις καταχωρήσεις του κάθε καταλύματος.
- Διαχειρίζεται τις συνδρομές που αντιστοιχούν σε κάθε κατάλυμα.

Ο ιδιοκτήτης (owner):

- Καταχωρεί/Διαγράφει το ξενοδοχείο.
- Δημιουργεί και δίνει πρόσβαση στον χρήστη ο οποίος είναι υπεύθυνος κρατήσεων(employee).
- Ενημερώνει/Ακυρώνει την συνδρομή για το ξενοδοχείο.
- Προσθέτει/Ενημερώνει/Αφαιρεί τα διαθέσιμα δωμάτια του ξενοδοχείου.
- Αποδέχεται/Απορρίπτει τις κρατήσεις του χρήστη-πελάτη.
- Προσθέτει/Αφαιρεί φωτογραφικό υλικό για το ξενοδοχείο και τα δωμάτια.

Ο υπεύθυνος κρατήσεων (employee):

- Αποδέχεται/Απορρίπτει τις κρατήσεις του χρήστη-πελάτη.
- Προσθέτει/Ενημερώνει/Αφαιρεί τα διαθέσιμα δωμάτια του ξενοδοχείου.

Ο χρήστης - πελάτης (unauthorized user):

- Έχει τη δυνατότητα αναζήτησης δωματίου και την επιλογή φίλτρων για τις βέλτιστες προτάσεις.
- Πραγματοποιεί την κράτηση του δωματίου.

1.2 Στόχος Εργασίας

Ως στόχο, η παρούσα εργασία, έχει τη δημιουργία μιας διαδικτυακής εφαρμογής, εύκολης στη χρήση αλλά και τη διαχείριση και από την πλευρά του εκάστοτε ιδιοκτήτη/υπαλλήλου αλλά και από την πλευρά του χρήστη-πελάτη.

Το ζητούμενο είναι να προσφέρουμε μια εύχρηστη εφαρμογή κρατήσεων που θα μπορεί εύκολα να ενσωματωθεί σε οποιαδήποτε ιστοσελίδα ξενοδοχείου, με αποτέλεσμα να το αναβαθμίσει αλλά και να το ανεξαρτητοποιήσει από εφαρμογές κρατήσεων οι οποίες κρατούν ποσοστά επί των κερδών τους για κάθε κράτηση. Επομένως, θα διευκολύνει και την λειτουργία και την εξέλιξη της επιχείρησης.

Κεφάλαιο 2ο: Αξιολόγηση και ενσωμάτωση χαρακτηριστικών εφαρμογών από παρεμφερή συστήματα κρατήσεων

Στο παρόν κεφάλαιο, θα αναλύσουμε τα χαρακτηριστικά από ήδη υπάρχουσες διαδικτυακές εφαρμογές, που αφορούν επίσης την αναζήτηση και κράτηση σε ξενοδοχειακά καταλύματα, και θα αναφερθούμε στην διαδικασία διαλογής και ενσωμάτωσης αυτών στην δική μας εφαρμογή. Κατά τη διάρκεια της ανάπτυξης της εφαρμογής μας, πραγματοποιήσαμε έρευνα και αντλήσαμε έμπνευση από διάφορες κορυφαίες πλατφόρμες κρατήσεων, όπως οι Booking, AirBnb, Hotels.com, Agoda, Trivago, Kayak, Trip.com και SiteMinder.

2.1 Σύντομη αναφορά σε ήδη υπάρχοντα Συστήματα Κρατήσεων

Κάποιες από τις εφαρμογές που επιλέξαμε να εξετάσουμε είναι:

Booking: Η Booking.com με έτος ίδρυσης το 1996 στο Άμστερνταμ και από μια μικρή ολλανδική start-up και εξελίχθηκε σε μια από τις πιο σημαντικές διαδικτυακές εταιρείες ταξιδιών παγκοσμίως. Επενδύοντας σε τεχνολογίες που κάνουν τα ταξίδια πιο απλά, η Booking.com συνδέει εύκολα εκατομμύρια ταξιδιώτες με μοναδικές εμπειρίες, πολλές επιλογές μετακίνησης και καταλύματα, από διαμερίσματα και ξενοδοχεία μέχρι διάφορες άλλες επιλογές. Ως μία από τις κορυφαίες πλατφόρμες ηλεκτρονικού εμπορίου στον τομέα των ταξιδιών, καλύπτει τόσο μεγάλες αλυσίδες όσο και επιχειρήσεις διαφόρων μεγεθών, η Booking.com επιτρέπει σε καταλύματα από όλο τον κόσμο να έρθουν σε επαφή με ένα διεθνές κοινό και να ενισχύσουν τις δραστηριότητές τους.

Η Booking.com είναι προσβάσιμη σε 43 γλώσσες και περιλαμβάνει περισσότερες από 28 εκατομμύρια καταχωρήσεις καταλυμάτων, εκ των οποίων πάνω από 6,6 εκατομμύρια αφορούν σπίτια, διαμερίσματα και άλλα μοναδικά μέρη διαμονής. Ανεξάρτητα από τον προορισμό, η Booking.com διευκολύνει την οργάνωση ενός ταξιδιού, προσφέροντας παράλληλα 24ωρη εξυπηρέτηση πελατών, καθημερινά.

AirBnb: Η Airbnb δημιουργήθηκε το 2007, με αρχικό όνομα AirBed & Breakfast το οποίο άλλαξε το 2009 στο πλέον γνώριμο σε όλους Airbnb.Η ιδέα της δημιουργίας του προέκυψε με αφορμή δύο οικοδεσπότες που φιλοξένησαν τρεις επισκέπτες στο σπίτι τους στο Σαν Φρανσίσκο. Το 2008 ξεκίνησε η λειτουργία της διαδικτυακής εφαρμογής αυξάνοντας την δημοτικότητα του σε σύντομο χρονικό διάστημα. Μέχρι τον Νοέμβριο του 2010, από τις 700.000 διανυκτερεύσεις που είχαν κρατηθεί, το 80% είχε πραγματοποιηθεί τους προηγούμενους έξι μήνες. Από τότε, έχει επεκταθεί σε περισσότερους από 5 εκατομμύρια οικοδεσπότες, οι οποίοι έχουν υποδεχθεί πάνω από 1,5 δισεκατομμύριο αφίξεις επισκεπτών σχεδόν σε κάθε χώρα του κόσμου. Καθημερινά, οι οικοδεσπότες προσφέρουν ξεχωριστές διαμονές και εμπειρίες, επιτρέποντας στους επισκέπτες να συνδεθούν με τις τοπικές κοινότητες με έναν πιο αυθεντικό τρόπο.

Η airbnb.com περιλαμβάνει πλέον παραπάνω από 8 εκατομμύρια καταχωρήσεις καταλυμάτων σε πάνω από 100 χιλιάδες περιοχές και πόλεις και με πάνω από 5 εκατομμύρια οικοδεσπότες σε όλο τον κόσμο.

Trivago: Η trivago ξεκίνησε το 2005 από τρεις συμφοιτητές στο Ντίσελντορφ της Γερμανίας και έκτοτε έχει εξελιχθεί σε κορυφαία πλατφόρμα παγκοσμίως για την αναζήτηση καταλυμάτων. Επικεντρώνεται στο να αλλάξει τον τρόπο με τον οποίο εκατομμύρια ταξιδιώτες αναζητούν και συγκρίνουν ξενοδοχεία και άλλους τύπους καταλυμάτων. Η αποστολή της trivago είναι να αποτελεί την πρώτη επιλογή για τους ταξιδιώτες όταν αναζητούν ξενοδοχείο. Το 2010 τίθεται σε λειτουργία η διαδικτυακή εφαρμογή και από το 2011 έως το 2013 χρησιμοποιείται ευρέως στην Ευρώπη, την Ασία και την Αμερική. Το 2019 μετράει πάνω από 5 εκατομμύρια καταχωρήσεις καταλυμάτων σε 190 χώρες παγκοσμίως και διατίθεται σε 31

γλώσσες.

Agoda: Η Agoda είναι μια ψηφιακή πλατφόρμα ταξιδιών που προσφέρει ταξιδιωτικές εμπειρίες υψηλής αξίας σε ανθρώπους σε όλο τον κόσμο μέσω της εφαρμογής και του ιστότοπού της. Σκοπός της είναι να επιτρέψει σε όλους να ταξιδέψουν, παρέχοντας οικονομικές προσφορές για ξενοδοχεία, πτήσεις, δραστηριότητες και πολλά άλλα, με μια απλουστευμένη εμπειρία κράτησης από την αρχή έως το τέλος. Από το 2005, διευκολύνει την αναζήτηση και την κράτηση ταξιδιών, καθιστώντας την όσο πιο απλή γίνεται. Η Agoda είναι μια εταιρεία τεχνολογίας ταξιδιών με εκατομμύρια εγγεγραμμένους χρήστες, παρέχοντας αξιόπιστες υπηρεσίες καθημερινά. Μετά από χρόνια δραστηριοποίησης η Agoda το 2021 πλέον φτάνει τις 2,9 εκατομμύρια καταχωρήσεις καταλυμάτων σε 56 χιλιάδες πόλεις παγκοσμίως και διατίθεται σε 38 διαφορετικές γλώσσες.

SiteMinder: Η siteminder.com ιδρύθηκε το 2006 από δύο φίλους στο Σίδνεϊ θέτοντας κατευθείαν σε λειτουργία την διαδικτυακή τους εφαρμογή και μεσα σε ελάχιστο χρονικό διάστημα, μόλις έξι μήνες, πάνω από 70 ξενοδοχειακές μονάδες το εμπιστεύτηκαν για την διαχείριση των επιχειρήσεων τους. Πλέον, το 2024 μετράει πάνω από 44 χιλιάδες καταχωρήσεις καταλυμάτων σε 150 χώρες παγκοσμίως δίνοντας έτσι την επιλογή σε μικρές ξενοδοχειακές μονάδες και επιχειρήσεις να είναι πιο εύκολα προσβάσιμες στους απλούς χρήστες και τις καθιστά πιο ανταγωνιστικές στην αγορά.

2.2 Θετικά και αρνητικά χαρακτηριστικά που παρατηρήθηκαν κατα την αναζήτησή μας

Κατά τη διαδικασία αναζήτησης και αξιολόγησης των εφαρμογών, με σκοπό τη συλλογή θετικών στοιχείων για ενσωμάτωση στην εφαρμογή μας, καθώς και την αποφυγή λιγότερο αποτελεσματικών χαρακτηριστικών, προχωρήσαμε στην ανάλυση διαφόρων παραδειγμάτων. Ενδεικτικά, εξετάστηκαν οι παρακάτω εφαρμογές και προέκυψαν τα εξής:

Η **Booking.com** ξεχωρίζει για την ποικιλία καταλυμάτων που προσφέρει, συμπεριλαμβανομένων διαμερισμάτων, εξοχικών κατοικιών και πολλών άλλων επιλογών. Ένα από τα πιο θετικά χαρακτηριστικά της είναι το **πρόγραμμα αφοσίωσης Genius**, το οποίο επιβραβεύει τους χρήστες με βάση τη συχνότητα των κρατήσεων, προσφέροντάς τους οφέλη, όπως εκπτώσεις ή επιπλέον παροχές, όπως το πρωινό. Επίσης, η πλατφόρμα συχνά προσφέρει **φθηνότερες τιμές** σε σχέση με τους ανταγωνιστές και πολλά καταλύματα παρέχουν δυνατότητα δωρεάν ακύρωσης, καθιστώντας τη μια αποδοτική επιλογή για τους χρήστες.

Ωστόσο, εντοπίσαμε και κάποια μειονεκτήματα. Ένα από τα πιο συχνά προβλήματα είναι οι **πολύπλοκες πολιτικές ακύρωσης**, οι οποίες διαφέρουν σημαντικά ανάμεσα στα καταλύματα, δημιουργώντας σύγχυση στους χρήστες. Επίσης, η πλατφόρμα δεν διαθέτει επιλογή για **ευρεία αναζήτηση** («αναζήτηση παντού»), κάτι που αρκετοί χρήστες, ιδιαίτερα όσοι δεν έχουν καταλήξει σε συγκεκριμένο προορισμό, θεωρούν ως σημαντική δυσλειτουργία.

Η **Airbnb** ξεχωρίζει για την προσφορά **μοναδικών καταλυμάτων**, τα οποία δεν είναι συνήθως διαθέσιμα σε παραδοσιακές ιστοσελίδες κρατήσεων. Η πλατφόρμα παρέχει επίσης **τεράστια γεωγραφική κάλυψη**, με δυνατότητα ενοικίασης σε περιοχές όπου τα παραδοσιακά ξενοδοχεία είτε δεν είναι διαθέσιμα είτε είναι υπερβολικά ακριβά. Επιπλέον, μέσω της επαφής με τους τοπικούς οικοδεσπότες, οι χρήστες μπορούν να βιώσουν μια **αυθεντική τοπική εμπειρία**, κάτι που πολλές φορές απουσιάζει από τις παραδοσιακές διαμονές σε ξενοδοχεία.

Ωστόσο, υπάρχουν και ορισμένα μειονεκτήματα. Συγκεκριμένα, έχουν αναφερθεί **προβλήματα ασφαλείας**, όπως περιστατικά με κρυφές κάμερες ή απάτες σε καταχωρήσεις. Επίσης, πολλές καταχωρήσεις συνοδεύονται από **επιπλέον χρεώσεις** (π.χ. τέλη καθαρισμού και υπηρεσιών), κάτι που αυξάνει το τελικό κόστος.

Η **Trivago** προσφέρει **ολοκληρωμένη σύγκριση τιμών** ανάμεσα σε πολλές πλατφόρμες κρατήσεων,

γεγονός που την καθιστά ισχυρό εργαλείο για τους χρήστες που επιθυμούν να βρουν την καλύτερη προσφορά. Επίσης, η πλατφόρμα διαθέτει **αποτελεσματική λειτουργία συγχρονισμού σε πολλαπλές συσκευές**, κάτι που είναι ιδιαίτερα χρήσιμο για όσους χρησιμοποιούν διάφορες συσκευές κατά την αναζήτηση. Ένα άλλο θετικό χαρακτηριστικό είναι η **ποικιλία φίλτρων αναζήτησης**, τα οποία επιτρέπουν στους χρήστες να προσαρμόζουν τις επιλογές τους ανάλογα με τις προτιμήσεις τους.

Παρόλα αυτά, η πλατφόρμα περιλαμβάνει **πολλές διαφημίσεις** και χορηγούμενο περιεχόμενο, το οποίο μπορεί να μην εξυπηρετεί πάντα τα καλύτερα συμφέροντα των χρηστών. Επιπλέον, οι χρήστες πρέπει να **ανακατευθύνονται σε άλλες ιστοσελίδες** για την ολοκλήρωση της κράτησης, προσθέτοντας ένα επιπλέον βήμα στη διαδικασία.

Η **Agoda** διακρίνεται για τη μεγάλη **ποικιλία καταλυμάτων**, από πολυτελή ξενοδοχεία παγκόσμιας κλάσης έως μικρά boutique ξενοδοχεία σε τοπικές γειτονιές. Η εφαρμογή διαθέτει επίσης εύχρηστο UI/UX, κάτι που την καθιστά ιδιαίτερα φιλική προς τον χρήστη.

Από την άλλη, ένα από τα μεγαλύτερα μειονεκτήματα της Agoda είναι η **μηδαμινή υποστήριξη πελατών**, κάτι που αναφέρουν συχνά οι χρήστες ως πρόβλημα. Επίσης, οι πολιτικές ακύρωσης διαφέρουν ανάλογα με τον πάροχο του καταλύματος, προκαλώντας ανησυχία στους χρήστες για τυχόν απρόβλεπτες καταστάσεις.

Με βάση τα παραπάνω, προχωρούμε στην αξιοποίηση των θετικών στοιχείων και στην αποφυγή των αρνητικών για τη δική μας εφαρμογή

2.3 Αξιολόγηση και Ενσωμάτωση των κατάλληλων χαρακτηριστικών στην εργασία μας

Με βάση τη διεξαγωγή της έρευνάς μας, αποφασίσαμε πως τα παρακάτω αποτελούν τα θεμελιώδη χαρακτηριστικά για την ομαλή λειτουργία μιας διαδικτυακής εφαρμογής και θα πρέπει να εξεταστούν εκτενέστερα.

Η **ευκολία χρήσης** είναι το πρώτο βασικό χαρακτηριστικό. Όλες οι πλατφόρμες προσφέρουν ένα φιλικό και εύχρηστο περιβάλλον για τον χρήστη, επιτρέποντας εύκολη και γρήγορη αναζήτηση, πλοήγηση και πραγματοποίηση κράτησης. Εξίσου σημαντικό είναι το χαρακτηριστικό των κριτικών. Οι περισσότερες πλατφόρμες επιτρέπουν στους χρήστες που έχουν ολοκληρώσει τη διαμονή τους σε ένα κατάλυμα να πραγματοποιήσουν αξιολογήσεις και κριτικές με βάση την προσωπική τους εμπειρία. Αυτό δημιουργεί ένα νέο φίλτρο αναζήτησης που βοηθά τους επόμενους χρήστες στην επιλογή του κατάλληλου καταλύματος.

Ένα συχνό χαρακτηριστικό στις εφαρμογές είναι οι **προσαρμοσμένες προσφορές**. Οι ιδιοκτήτες ή διαχειριστές των καταλυμάτων μπορούν να εφαρμόζουν ειδικές προσφορές και εκπτώσεις ανάλογα με την πληρότητα, την περίοδο ή άλλες περιστάσεις, καθιστώντας τα καταλύματά τους πιο ανταγωνιστικά και ελκυστικά για διαφορετικές κατηγορίες πελατών. Οι πλατφόρμες παρέχουν επίσης δυνατότητες για άμεση ενημέρωση των χρηστών σχετικά με τις αλλαγές στη διαθεσιμότητα και τις τιμές σε πραγματικό χρόνο, ανάλογα με τις ημερομηνίες που έχουν επιλέξει.

Όσον αφορά την **αξιοπιστία των πληροφοριών**, όλες οι πλατφόρμες προσφέρουν αξιόπιστες, επαληθευμένες και ενημερωμένες πληροφορίες σχετικά με το κατάλυμα, τις παροχές του, τις οδηγίες πρόσβασης και τους τρόπους επικοινωνίας. Σε πολλές εφαρμογές παρατηρείται και η λειτουργία συστημάτων **επιβράβευσης χρηστών**. Οι χρήστες συχνα απολαμβάνουν εκπτώσεις στην επόμενη κράτησή τους ή επιπλέον παροχές, όπως δωρεάν πρωινό, κάτι που λειτουργεί ως κίνητρο για να συνεχίσουν να χρησιμοποιούν την πλατφόρμα και να εκμεταλλεύονται τα οφέλη αυτών των προγραμμάτων.

Ένα ακόμη χρήσιμο χαρακτηριστικό που εντοπίσαμε είναι η **δυνατότητα ειδοποιήσεων**. Οι χρήστες ενημερώνονται μέσω γραπτών μηνυμάτων ή email, ενώ σε ορισμένες περιπτώσεις οι ημερομηνίες των κρατήσεων ενσωματώνονται στο Google Calendar για να υπενθυμίζουν τις επερχόμενες κρατήσεις και να διευκολύνουν τη διαχείρισή τους. Το customer support αποτελεί επίσης ένα πολύ σημαντικό εργαλείο, το

οποίο επιτρέπει στους χρήστες να επικοινωνούν άμεσα με την υποστήριξη πελατών της πλατφόρμας για να λύσουν οποιοδήποτε πρόβλημα ή να λάβουν επιπλέον πληροφορίες σχετικά με την κράτησή τους.

Όσον αφορά τις **μεθόδους πληρωμής**, οι πλατφόρμες δίνουν τη δυνατότητα στους χρήστες να πραγματοποιούν πληρωμές με διαφορετικούς τρόπους, λειτουργώντας πολλές φορές ως διαμεσολαβητές μεταξύ του χρήστη και του καταλύματος. Όλες οι μέθοδοι πληρωμής δίνουν προτεραιότητα στην ασφάλεια των συναλλαγών, και σε ορισμένες περιπτώσεις, οι χρήστες έχουν τη δυνατότητα να εξοφλήσουν μερικώς ή να πραγματοποιήσουν πληρωμές σε δόσεις. Αυτή η διαδικασία θα ήταν ιδιαίτερα δύσκολη χωρίς τη διαμεσολάβηση των διαδικτυακών εφαρμογών, καθώς κάθε κατάλυμα θα έπρεπε να διαχειρίζεται ξεχωριστ2ά τις πληρωμές για κάθε χρήστη.

Τέλος, σε κάποιες από τις πιο καινοτόμες πλατφόρμες γίνεται **χρήση εργαλείων τεχνητής νοημοσύνης** (AI) σε πιλοτικό στάδιο. Οι χρήστες έχουν τη δυνατότητα να συνομιλήσουν με έναν "Διοργανωτή ταξιδιού" (Trip Planner), ο οποίος τους παρέχει προσωποποιημένες πληροφορίες με βάση το είδος του ταξιδιού που θέλουν να οργανώσουν. Επιπλέον, ο χρήστης μπορεί να ρωτήσει διάφορες απορίες που προκύπτουν κατά την αναζήτηση και κράτηση καταλύματος, διευκολύνοντας την οργάνωση του ταξιδιού του.

Αναλύοντας τα παραπάνω χαρακτηριστικά και αξιολογώντας τα με βάση τις ανάγκες της δικής μας εφαρμογής, αποφασίσαμε να εστιάσουμε σε αυτά που θεωρούμε πιο θεμελιώδη για την ομαλή λειτουργία της εφαρμογής και τη διευκόλυνση των χρηστών στην επίτευξη του στόχου τους, που είναι η πραγματοποίηση της κράτησης με τις απαραίτητες παροχές.

Πρώτα, επικεντρωθήκαμε στην **ευκολία χρήσης**. Δημιουργήσαμε ένα απλό και φιλικό περιβάλλον τόσο για τους χρήστες που αναζητούν δωμάτιο, καθιστώντας την αναζήτηση και την κράτηση εύκολη με λίγα βήματα, όσο και για τους διαχειριστές του καταλύματος. Οι διαχειριστές έχουν τη δυνατότητα να προσθέσουν και να ενημερώσουν τα απαραίτητα στοιχεία και τα διαθέσιμα δωμάτια, καθώς και να διαχειριστούν τον όγκο των κρατήσεων αποτελεσματικά.

Παρέχουμε επίσης τη δυνατότητα στους ιδιοκτήτες ή διαχειριστές να εφαρμόσουν **πακέτα προσαρμοσμένων προσφορών** στο κατάλυμά τους, έχοντας πλήρη και αποκλειστικό έλεγχο σε αυτά. Η λεπτομερής καταχώρηση πληροφοριών για το κατάλυμα και για κάθε δωμάτιο ξεχωριστά είναι επίσης απαραίτητη ευθύνη του διαχειριστή, ώστε οι χρήστες να έχουν πλήρη και αξιόπιστη ενημέρωση.

Η ενημέρωση των χρηστών σχετικά με τη **διαθεσιμότητα** των δωματίων και τις τιμές γίνεται σε **πραγματικό χρόνο**, προσφέροντάς τους άμεση εικόνα των διαθέσιμων επιλογών. Τέλος, προσθέσαμε **πολλαπλούς τρόπους πληρωμής** στην εφαρμογή μας, δίνοντας στον χρήστη τη δυνατότητα να επιλέξει τον καταλληλότερο τρόπο για τις ανάγκες του, πάντα μέσα στο πλαίσιο των ασφαλών συναλλαγών.

Πριν καταλήξουμε στην χρήση των παραπάνω χαρακτηριστικών για την εφαρμογή μας και μέσω της διαδικασίας της αξιολόγησης, υπήρξε και το στάδιο δοκιμής μερικών ακόμη βασικών χαρακτηριστικών τα οποία εν τέλει δεν ταίριαζαν με την μορφή και τον χαρακτήρα της συγκεκριμένης διαδικτυακής εφαρμογής. Ένα χαρακτηριστικό παράδειγμα αυτού αποτέλεσε η ενσωμάτωση κριτικών και αξιολογήσεων από τους χρήστες για το κάθε κατάλυμα. Η συγκεκριμένη πρακτική θεωρούμε πως δεν θα είναι αρκετά χρήσιμη και θα χάσει τον αρχικό της στόχο, δηλαδή να αποτελεί ένα από τα κριτήρια επιλογής καταλύματος, διότι ο χρήστης έχει ήδη επιλέξει το συγκεκριμένο κατάλυμα για να αναζητήσει διαμονή και δεν υπάρχει περιθώριο σύγκρισης με κάποιο άλλο. Θα μπορούσαν να υπάρχουν κριτικές και αξιολογήσεις στον αρχικό ιστότοπο του καταλύματος, ώστε να το κάνουν ακόμα πιο θελκτικό και να πείσουν περισσότερο τον χρήστη να μπει στην διαδικασία κράτησης. Συνεπώς αποφασίσαμε πως θα ήταν άσκοπο να αποτελέσει αυτοτελή λειτουργία της δικής μας εφαρμογής.

Εκτός από τα συστατικά χαρακτηριστικά αυτών τον συστημάτων, κάναμε επίσης μια τεχνική έρευνα ώστε να συμβουλευτούμε για την δική μας εφαρμογή. Παρατηρησαμε λοιπόν, πως σχεδόν όλα έχουν χρησιμοποιήσει κάποια απο τα frameworks της Javascript, όπως Vue.js, React.js, για να υλοποιήσουν το "Front-End" μέρος τους. Σχετικά με το "Back-End", παρατηρήθηκε πως τα περισσότερα συστήματα έχουν στηριχθεί στην γλώσσα προγραμματισμού της PHP, όμως υπάρχουν και κάποια παραδείγματα χρήσης

ενός ακόμη framework της Javascript, την node.js. Αν και δεν αποτέλεσε μοναδικό παράγοντα για να αποφασίσουμε ποιες τεχνολογίες θα χρησιμοποιούσαμε, σίγουρα έπαιξε σημαντικό ρόλο. Η έρευνα και αξιολόγηση εφαρμογών που ήδη χρησιμοποιούνται ευρέως μας βοήθησαν να 2κατανοήσουμε ποιες τεχνικές έχουν αποδειχθεί χρήσιμες, αποτελεσματικές και δημοφιλής όπως επίσης και ποιες ενδεχομένως εμπεριέχουν περιθώρια σφάλματος. Ακόμη, μας οδήγησαν στην καλύτερη κατανόηση των αναγκών ενός χρήστη και να προσπαθήσουμε να βελτιστοποιήσουμε την εμπειρία του, θέτοντας έτσι την βάση για την σχεδίαση και υλοποιήση της εφαρμογής μας.

Κεφάλαιο 3ο:Μέθοδοι Ανάπτυξης Διαδικτυακών Εφαρμογών

Στο τρέχον κεφάλαιο, θα αναφερθούμε εν συντομία σε όλες τις διαθέσιμες τεχνικές που μπορούν να χρησιμοποιηθούν για την ανάπτυξη μιας διαδικτυακής εφαρμογής. Θα επικεντρωθούμε, ειδικότερα, στην ανάλυση και επιλογή των μεθόδων που χρησιμοποιούνται για την υλοποίηση της εφαρμογής που παρουσιάζεται στο πλαίσιο της παρούσας διπλωματικής εργασίας.

3.1 Διαδικτυακές Εφαρμογές

Μια δικτυακή εφαρμογή, γνωστή και ως ηλεκτρονική εφαρμογή ή web app, αποτελεί λογισμικό που δεν απαιτεί εγκατάσταση και, αντίθετα, επιτρέπει την πρόσβαση μέσω ενός προγράμματος περιήγησης Ιστού. Οι web apps σχεδιάζονται για αλληλεπίδραση, επιτρέποντας στους χρήστες να μεταδίδουν και να λαμβάνουν δεδομένα μεταξύ του προγράμματος περιήγησης και του web server. Κατ' αυτόν τον τρόπο, ο χρήστης μπορεί να αποκτήσει πρόσβαση σε λειτουργίες και πληροφορίες χωρίς την ανάγκη για τοπική εγκατάσταση λογισμικού.

Μια τυπική ροή μιας διαδικτυακής εφαρμογής ξεκινά όταν ο χρήστης ενεργοποιεί ένα αίτημα στον διακομιστή ιστού μέσω του Διαδικτύου, είτε μέσω ενός προγράμματος περιήγησης Ιστού είτε μέσω της διεπαφής χρήστη της εφαρμογής. Ο διακομιστής ιστού προωθεί αυτό το αίτημα στον κατάλληλο διακομιστή εφαρμογών Ιστού. Στη συνέχεια, ο διακομιστής εφαρμογών εκτελεί την εργασία που ζητήθηκε, όπως η αναζήτηση στη βάση δεδομένων ή η επεξεργασία των δεδομένων, και δημιουργεί τα αποτελέσματα των ζητούμενων δεδομένων. Αυτά τα αποτελέσματα αποστέλλονται πίσω στον διακομιστή ιστού, ο οποίος απαντά στον πελάτη με τις ζητούμενες πληροφορίες ή τα επεξεργασμένα δεδομένα. Τελικά, οι πληροφορίες αυτές εμφανίζονται στην οθόνη του χρήστη.

Ένα διαδικτυακό πρόγραμμα απαρτίζεται από δύο κατηγορίες κώδικα που συνεργάζονται για να εξασφαλίσουν την ορθή λειτουργία του. Αναφερόμαστε στο μέρος του front-end, που αντιπροσωπεύει το ορατό επίπεδο της εφαρμογής και αλληλεπιδρά με τον χρήστη, προσφέροντας την όψη και τη διάταξη του. Το δεύτερο μέρος αφορά το back-end, που φέρει την ευθύνη του για το λογικό επίπεδο και τις λειτουργίες της εφαρμογής, οι οποίες δεν είναι ορατές από τον χρήστη, σε πρώτο επίπεδο.

3.1.1 Ορισμός και Δομή Αρχιτεκτονικής Διαδικτυακής Εφαρμογής

Η αρχιτεκτονική διαδικτυακής εφαρμογής αναφέρεται στον σχεδιασμό και την οργάνωση των διαφόρων συστατικών μιας εφαρμογής που λειτουργεί μέσω του Διαδικτύου. Η σωστή αρχιτεκτονική είναι κρίσιμη για την απόδοση, την επεκτασιμότητα, την ασφάλεια και τη συντηρησιμότητα μιας εφαρμογής.

Συνήθως, μια αρχιτεκτονική διαδικτυακής εφαρμογής περιλαμβάνει διάφορα συστατικά. Το πρώτο είναι το **Client-Side**, το οποίο αναφέρεται στο Front-End της εφαρμογής και εκτελείται στον περιηγητή του χρήστη. Σε αυτήν την κατηγορία υπάγονται τεχνολογίες όπως HTML, CSS, JavaScript και frameworks όπως το React ή το Angular.

Το δεύτερο συστατικό είναι το **Server-Side**, το οποίο περιλαμβάνει το Back-End της εφαρμογής που εκτελείται στον διακομιστή. Εδώ χρησιμοποιούνται τεχνολογίες όπως Node.js, Django, Ruby on Rails και άλλες.

Τέλος, υπάρχει η **Βάση Δεδομένων**, η οποία αναλαμβάνει την αποθήκευση και τη διαχείριση των

δεδομένων της εφαρμογής. Συχνά, για αυτό το σκοπό χρησιμοποιούνται συστήματα όπως MySQL, PostgreSQL και MongoDB.

3.1.2 Αρχιτεκτονική Επιπέδων

Η αρχιτεκτονική επιπέδων στον προγραμματισμό αναφέρεται στην οργάνωση του λογισμικού σε διάφορα επίπεδα, κάθε ένα από τα οποία εκτελεί συγκεκριμένες λειτουργίες, ενώ τα επίπεδα αυτά αλληλεπιδρούν μεταξύ τους με συγκεκριμένο τρόπο. Η δομή αυτή επιτρέπει την εύκολη συντήρηση, επέκταση και ανάπτυξη του λογισμικού, καθώς κάθε επίπεδο είναι ανεξάρτητο από τα άλλα.

Συνεπώς, η αρχιτεκτονική μας δίνει πνευματικό έλεγχο πάνω στο πολύ πολύπλοκο επιτρέποντάς μας να αντικαταστήσουμε το σύμπλεγμα με ένα σύνολο αλληλεπιδρώντων κομματιών, που ουσιαστικά, το καθένα από αυτά, είναι απλούστερο από το σύνολο. [2]

3.1.2.1 Αρχιτεκτονική Τριών Επιπέδων

Η πλειοψηφία των Διαδικτυακών Εφαρμογών υλοποιούνται με βάση τον διαχωρισμό της κύριας λειτουργίας τους σε επίπεδα. Συνεπώς χρησιμοποιούν την αρχιτεκτονική τριών επιπέδων, η οποία αποτελείται από τα εξής τρία(3) παρακάτω επίπεδα:

1. Επίπεδο Παρουσίασης
2. Επίπεδο εφαρμογής και
3. Επίπεδο Δεδομένων [3]

Μία Διαδικτυακή Εφαρμογή αποτελείται από το Front – End μέρος, το οποίο περιλαμβάνει όλες τις οπτικές πτυχές μιας Διαδικτυακής Εφαρμογής, με τις οποίες μπορεί να αλληλεπιδράσει ο χρήστης, όπως τα χρώματα, τη διάταξη και τις γραμματοσειρές, και το Back – End μέρος, το οποίο δημιουργεί την «αόρατη» δομή που καθιστά δυνατή την ορθή λειτουργία του Front – End μέρους και του συνόλου των λειτουργιών της Διαδικτυακής Εφαρμογής. Όπως φαίνεται στο Σχήμα 2.2, το Front – End μέρος υλοποιείται στο Επίπεδο Παρουσίασης, το οποίο χειρίζεται τις αλληλεπιδράσεις των χρηστών με την εφαρμογή. Το Back – End μέρος υλοποιείται στο Επίπεδο Εφαρμογής, το οποίο ενορχηστρώνει το σύνολο των εργασιών της εφαρμογής και διαχειρίζεται όλα τα αιτήματα των χρηστών, και στο Επίπεδο Δεδομένων, το οποίο διαχειρίζεται την αποθήκευση και τη διαχείριση των δεδομένων της εφαρμογής .

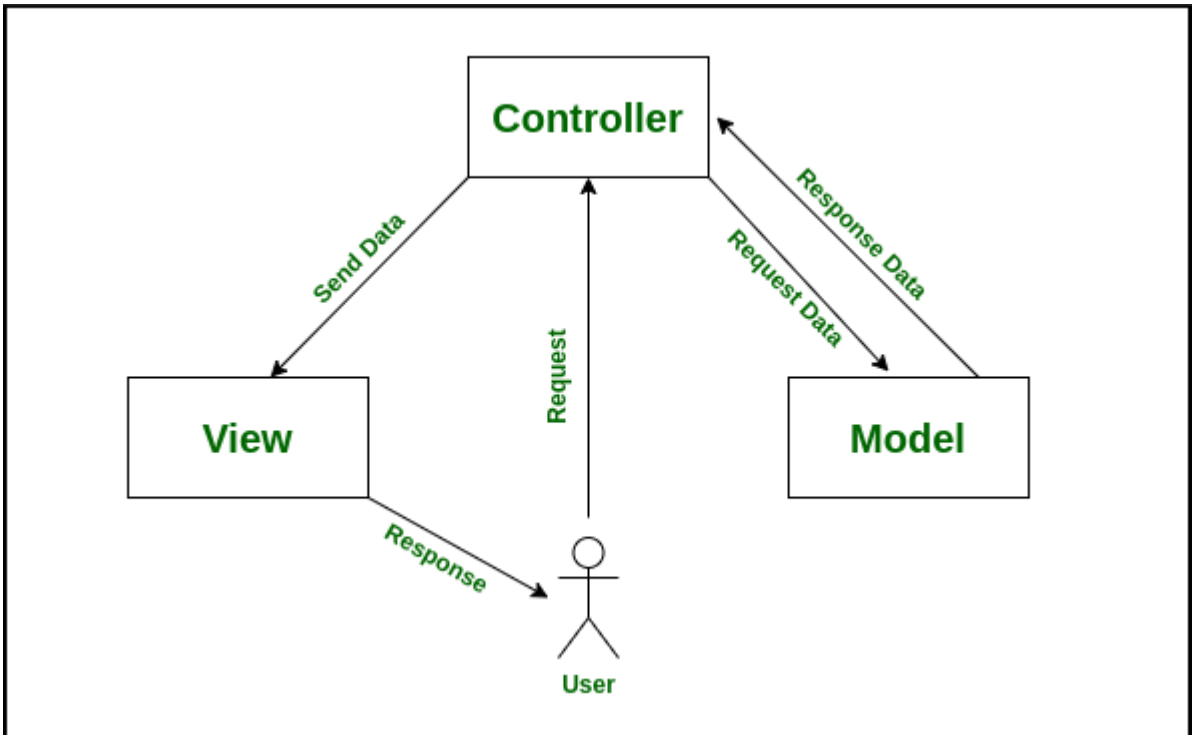
Η Αρχιτεκτονική Τριών (3) Επιπέδων είναι ασφαλέστερη για πολλούς λόγους. Πρώτον, ο χρήστης δεν έχει άμεση πρόσβαση στα δεδομένα, γεγονός που ενισχύει την ασφάλεια των πληροφοριών. Δεύτερον, η ακεραιότητα των δεδομένων βελτιώνεται, καθώς όλα τα δεδομένα περνούν μέσω του Διακομιστή Ιστού (Web Server), ο οποίος έχει τον έλεγχο και αποφασίζει πώς και από ποιον θα υπάρχει πρόσβαση σε αυτά. Επιπλέον, αυτή η αρχιτεκτονική μπορεί να αναπτυχθεί περαιτέρω, επιτρέποντας την ανεξάρτητη ανάπτυξη κάθε στοιχείου και επιπέδου της εφαρμογής, χωρίς να επηρεάζει τα υπόλοιπα. Τέλος, παρέχει υψηλότερη επεκτασιμότητα, καλύτερη απόδοση και επαναχρησιμοποίηση των στοιχείων της εφαρμογής, υποστηρίζοντας διαφορετικές βάσεις δεδομένων όπως MySQL, PostgreSQL και MongoDB.

3.1.2.2 Αρχιτεκτονική MVC

Η αρχιτεκτονική MVC, η οποία εισήχθη από τον Trygve Reenskaug, προγραμματιστή σε Smalltalk στο Κέντρο Ερευνών του Xerox Palo Alto το 1979, αποτελεί ένα μοντέλο λογισμικού που διευκολύνει τον

αποσυμπλεγμένο χειρισμό της πρόσβασης και της επιχειρηματικής λογικής από τον τρόπο παρουσίασης στον χρήστη. Το MVC διακρίνεται σε τρία κύρια στοιχεία.

Πρώτον, το **μοντέλο (Model)**, το οποίο αναπαριστά τα δεδομένα και τους κανόνες που διέπουν την πρόσβαση και τις ενημερώσεις αυτών των δεδομένων. Στον τομέα του λογισμικού επιχειρήσεων, ένα μοντέλο συχνά χρησιμεύει ως λογισμική προσέγγιση των διεργασιών του πραγματικού κόσμου. Δεύτερον, η **προβολή (View**), η οποία απεικονίζει το περιεχόμενο ενός μοντέλου και καθορίζει ακριβώς πώς θα πρέπει να παρουσιαστούν τα δεδομένα του. Όταν τα δεδομένα του μοντέλου αλλάξουν, η προβολή ενημερώνει την παρουσίασή της σύμφωνα με τις απαιτήσεις. Τέλος, ο **ελεγκτής (Controller)** μεταφράζει τις αλληλεπιδράσεις του χρήστη με την προβολή σε ενέργειες που θα πραγματοποιήσει το μοντέλο. Σε ένα αυτόνομο πελάτη γραφικού περιβάλλοντος χρήστη, οι αλληλεπιδράσεις αυτές μπορεί να περιλαμβάνουν κλικ σε κουμπιά ή επιλογές μενού, ενώ σε μια επιχειρηματική διαδικτυακή εφαρμογή εμφανίζονται ως αιτήσεις HTTP GET και POST. Ανάλογα με το πλαίσιο, ο ελεγκτής μπορεί επίσης να επιλέξει μια νέα προβολή, όπως μια σελίδα αποτελεσμάτων, για να παρουσιάσει στον χρήστη.[4]



Σχήμα Αρχιτεκτονική MVC

3.2 Τεχνολογίες Ανάπτυξης Front – End Μέρους Διαδικτυακής Εφαρμογής

Το Front-End μέρος αναφέρεται στο χρηστικό και οπτικό μέρος μιας διαδικτυακής εφαρμογής που αλληλεπιδρά με τον τελικό χρήστη. Περιλαμβάνει τη σχεδίαση, το layout, την πλοήγηση, καθώς και την χρήση γραφικών στοιχείων για τη βελτίωση της εμπειρίας του χρήστη. Είναι στην ουσία ο τρόπος με τον οποίο μπορεί ένας απλός χρήστης να αλληλεπιδράσει με τον με τον Διακομιστή Ιστού (Web Server) και την Υπηρεσία Υποστήριξης (Back – End Service) μέσω ενός Προγράμματος Περιήγησης (Web Browser).

Περιλαμβάνει κυρίως τρεις γλώσσες προγραμματισμού και σχετικές τεχνολογίες ανάπτυξης. Ανάμεσά τους, η γλώσσα JavaScript, η HTML και η CSS. Άλλες τεχνολογίες ανάπτυξης που χρησιμοποιούνται είναι το DOM, η AJAX, κ.ά. Το σύστημα τεχνολογίας ανάπτυξης για τον front-end του ιστού μπορεί να παρατηρηθεί στο Σχήμα 1 [5]

3.2.1 Frameworks Διαδικτυακών Εφαρμογών

Ένα πλαίσιο (framework) είναι ένα κομμάτι λογισμικού που διευκολύνει την ανάπτυξη και τη συντήρηση μεγάλων projects. Τα πλαίσια είναι συλλογές θεμελιωδών λειτουργικών μονάδων λογισμικού που περιλαμβάνουν προετοιμασμένο κώδικα που θα μπορούσαν να χρησιμοποιήσουν οι προγραμματιστές για να διορθώσουν κοινές εργασίες προγραμματισμού, όπως το χειρισμό αιτημάτων AJAX ή την προσπάθεια ορισμού μιας δομής αρχείου. Καθορίζουν επίσης τους κανόνες για την ανάπτυξη της αρχιτεκτονικής της εφαρμογής: έχετε μια σκελετική δομή που πρέπει να επεκταθεί και να αλλάξει σύμφωνα με τις απαιτήσεις. Δείτε την καλύτερη πιστοποίηση προγραμματιστή Full Stack για περισσότερες λεπτομέρειες.

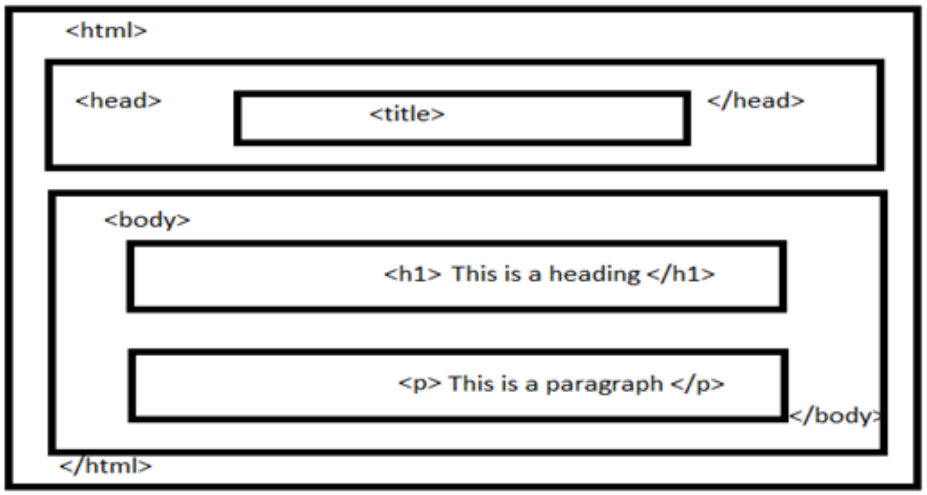
Τα βοηθητικά προγράμματα, οι βιβλιοθήκες κώδικα, οι γλώσσες δέσμης ενεργειών και άλλο λογισμικό που διευκολύνει την ανάπτυξη και την υλοποίηση πολλαπλών στοιχείων ενός μεγάλου προϊόντος λογισμικού είναι παραδείγματα πλαισίων. Με τα πλαίσια, οι προγραμματιστές δεν χρειάζεται να ξεκινούν έργα από το μηδέν, αλλά αντίθετα έχουν τη βάση για την εφαρμογή άλλων λειτουργιών που αφορούν συγκεκριμένα έργα, συνεπώς, μπορούν να επικεντρωθούν στη λογική της εφαρμογής, αντί να ξοδεύουν χρόνο σε βασικές λειτουργίες.

3.2.2 HTML

Η HTML (HyperText Markup Language) είναι μια γλώσσα σήμανσης που καθορίζει το νόημα και τη δομή του περιεχομένου μας και είναι το πιο βασικό στοιχείο για τη δημιουργία και την οργάνωση του περιεχομένου στον Παγκόσμιο Ιστό. Στο πλαίσιο της HTML, χρησιμοποιούνται ετικέτες (tags), οι οποίες αποτελούνται από το όνομα του στοιχείου περικυκλωμένο από "<" και ">" και περιγράφουν τα διάφορα τμήματα του περιεχομένου, όπως κείμενο, εικόνες, συνδέσμους και πίνακες, προσδίδοντας τους διάφορες δομικές και λειτουργικές ιδιότητες, ώστε να εμφανιστούν με συγκεκριμένο τρόπο ή να ενεργούν με συγκεκριμένο τρόπο.

Η HTML (HyperText Markup Language) έχει μια ιεραρχική δομή που αναπαρίσταται με τη χρήση στοιχείων και ετικετών. Το κεντρικό μέρος <html>, χωρίζεται σε επικεφαλίδα <head> και σώμα <body>. Στο σώμα, χρησιμοποιούνται επιπλέον στοιχεία όπως <header>, <nav>, <section>, <article>, και <footer> για να οργανώσουν τη σελίδα, αλλά και ετικέτες όπως <title>, <p>, , <img>, <lists> κ.α. για να διαμορφώσουν οπτικά την ιστοσελίδα.

Εν κατακλείδι, η HTML είναι ο βασικός πυλώνας που επιτρέπει σε διαφορετικά στοιχεία - κείμενο, εικόνες, συνδέσμους - να ενσωματωθούν με λογικό τρόπο στην δομή των ιστοσελίδων και να προβληθούν στην οθόνη του χρήστη, “διαβάζοντας” και “μεταφράζοντας” τις ετικέτες. [6][7]



Σχήμα 1:Δομή Σελίδας Html

3.2.3 JavaScript

Η JavaScript είναι μια προγραμματιστική γλώσσα σεναρίων που χρησιμοποιείται κυρίως για τη δημιουργία διαδραστικών και δυναμικών ιστοσελίδων. Είναι υπεύθυνη για την ενίσχυση της διαδραστικότητας στις ιστοσελίδες με την εκτέλεση κώδικα απευθείας στον φυλλομετρητή του χρήστη.

Όταν η JavaScript εμφανίστηκε για πρώτη φορά το 1995, ο βασικός της σκοπός ήταν να αντιμετωπίζει μέρος της επικύρωσης εισόδου που προηγουμένως είχε αφεθεί σε γλώσσες στην πλευρά του διακομιστή. Πριν από αυτό το χρονικό διάστημα, ήταν απαραίτητη μια επίσκεψη στον διακομιστή για να καθορίσει εάν ένα υποχρεωτικό πεδίο είχε αφεθεί κενό ή αν μια εισαγμένη τιμή ήταν άκυρη. Η Netscape Navigator επιδίωξε να αλλάξει αυτό με την εισαγωγή της JavaScript. Η δυνατότητα να χειρίζεται μερική βασική επικύρωση στον πελάτη ήταν μια συναρπαστική νέα λειτουργία σε ένα χρονικό σημείο όπου η χρήση τηλεφωνικών modems ήταν ευρέως διαδεδομένη. Οι συνοδευτικές χαμηλές ταχύτητες μετέτρεψαν κάθε ταξίδι στον διακομιστή σε άσκηση υπομονής.

Από τότε, η JavaScript έχει εξελιχθεί σε μια σημαντική λειτουργία κάθε κύριου προγράμματος περιήγησης στην αγορά. Δεν περιορίζεται πλέον στην απλή επικύρωση δεδομένων, καθώς η JavaScript αλληλεπιδρά τώρα με σχεδόν όλες τις πτυχές του παραθύρου του προγράμματος περιήγησης και του περιεχομένου του.

Ουσιαστικά, ξεκίνησε ως γλώσσα πελάτη (client-side), χρησιμοποιούμενη για την επέκταση της διαδραστικότητας στις ιστοσελίδες στον πελάτη, δηλαδή στον φυλλομετρητή του χρήστη. Ωστόσο, με την εισαγωγή του Node.js, η JavaScript επεκτάθηκε και στην πλευρά του διακομιστή (server-side).

Το Node.js είναι ένα πλαίσιο (framework) που επιτρέπει την εκτέλεση της JavaScript στον διακομιστή. Αυτό σημαίνει ότι μπορείτε να χρησιμοποιήσετε JavaScript για την ανάπτυξη server-side εφαρμογών, διαχείριση βάσεων δεδομένων, και άλλες λειτουργίες που εκτελούνται στον διακομιστή.

Έτσι, σήμερα η JavaScript μπορεί να χρησιμοποιηθεί τόσο στην πλευρά του πελάτη όσο και στην πλευρά του διακομιστή, καθιστώντας την μια από τις πιο ευέλικτες και ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού.[8]

3.2.4 AngularJS

Το Angular JS είναι ένα διαδεδομένο πλαίσιο που χρησιμοποιείται για την ανάπτυξη εφαρμογών ιστού.

Βοηθά στη δημιουργία δυναμικών μοτίβων προβολής στη δημιουργία εφαρμογών. Επίσης, μπορεί να ρυθμιστεί με μετασχηματισμούς που μπορούν να μετατρέψουν ένα μοντέλο σε ένα πρότυπο κωδικοποίησης. Όταν δημιουργείται αυτό το πρότυπο, ένας προγραμματιστής μπορεί να συμπληρώσει το πρότυπο για να προσαρμόσει τις εργασίες του. Το Angular JS είναι ένα πολύτιμο πλαίσιο JavaScript για τη δημιουργία μονοσέλιδων ιστοσελίδων. Σχεδιάστηκε για να υποστηρίζει δυναμικές προβολές προκειμένου να καθιστά την ιστοσελίδα ομαλή στην εμφάνιση. Ορισμένα σημαντικά χαρακτηριστικά του Angular JS περιλαμβάνουν:

• **Μοντέλο Προβολής Ελέγχοντας (Model View Controller - MVC):** Βοηθά στον διαχωρισμό της εφαρμογής σε τρία επίπεδα. Πρώτον, μια προβολή για το διεπαφή χρήστη. Δεύτερον, τα δεδομένα που παρουσιάζονται στους χρήστες. Και τρίτον, ο ελεγκτής, ο οποίος είναι η λογική που ελέγχει τα δεδομένα που εμφανίζονται σε μια προβολή καταναλωτή.

• **Πρότυπο (Template):** ο HTML κώδικας που μπορεί να μετατραπεί σε ένα αντικείμενο μοντέλου ενώ δείχνει μια διεπαφή.

• **Διπλή Κατεύθυνση Δεδομένων (Two-way Data Binding):** Βοηθά τις προβολές να αλλάζουν με την αλλαγή του μοντέλου.

• **Εισαγωγή Εξαρτήσεων (Dependency Injection):** Φορτώνει όλες τις υπηρεσίες πλήρως πριν από την επεξεργασία.

Τα πλεονεκτήματα της χρήσης του πλαισίου AngularJS στην ανάπτυξη διαδικτυακών εφαρμογών περιλαμβάνουν το γεγονός ότι είναι ένα δωρεάν πλαίσιο ανοιχτού κώδικα JavaScript, το οποίο χρησιμοποιεί και ελέγχει πλήρως την αρχιτεκτονική MVC. Επιπλέον, υποστηρίζει την αμφίδρομη ανανέωση των στοιχείων του μοντέλου και της προβολής της αρχιτεκτονικής MVC, προσφέροντας ευκολία στην ανάπτυξη. Είναι εύκολα επεκτάσιμο, προσαρμοζόμενο και δοκιμαζόμενο, χωρίς να απαιτεί εκμάθηση νέας γλώσσας, καθώς οι προγραμματιστές μπορούν να χρησιμοποιούν τις γνωστές γλώσσες JavaScript και HTML. Το AngularJS συμβάλλει επίσης στην ανάπτυξη μίας αποκριτικής διαδικτυακής εφαρμογή και υποστηρίζει την ανάπτυξη εφαρμογών μονής σελίδας (Single Page Applications - SAPs) καθώς και εφαρμογών REST (Representational State Transfer).

Αν και υπάρχουν σημαντικά πλεονεκτήματα, το AngularJS επιφέρει και ορισμένα μειονεκτήματα. Ως πλαίσιο της JavaScript, στηρίζεται αποκλειστικά στη γλώσσα JavaScript, γεγονός που μπορεί να περιορίσει την ευχρηστία του για προγραμματιστές που δεν είναι εξοικειωμένοι με αυτήν. Επιπλέον, δεν ακολουθεί συγκεκριμένο μοτίβο σχεδίασης και ανάπτυξης, καθιστώντας δύσκολη την παρακολούθηση της εκμάθησης του και τον εντοπισμό σφαλμάτων. Τέλος, δεν υποστηρίζεται ευρέως, καθώς δεν διαθέτει πλήρως ενεργή κοινότητα προγραμματιστών.

3.2.5 React

Η ReactJS είναι μια βιβλιοθήκη JavaScript που βασίζεται στη σύνθετη δομή των στοιχείων (component-based) και ακολουθεί τον δηλωτικό προγραμματιστικό παραδειγματισμό. Οι δηλωτικές προβολές μπορούν να χρησιμοποιηθούν για τη δημιουργία πολύπλοκων διαδραστικών διεπαφών χρήστη, οι οποίες λειτουργούν ως το στοιχείο παρουσίασης του διαδικτυακού τόπου.

Η βιβλιοθήκη είναι εύκολη στην εκμάθηση, προωθεί την επαναχρησιμοποίηση κώδικα, προσφέρει ελαφρύ DOM για βελτιωμένη απόδοση, επιβάλλει ροή δεδομένων μιας διεύθυνσης, εξασφαλίζει βελτιστοποίηση για τις μηχανές αναζήτησης (SEO), διαμορφώνει τις προβολές της αρχιτεκτονικής Model View Controller (MVC) και υποστηρίζει το εικονικό DOM .

Το εικονικό DOM διαχειρίζεται από τη διαδικασία συμφιλίωσης όπου μια ιδέα ή εικονική αναπαράσταση

μιας διεπαφής χρήστη αποθηκεύεται στη μνήμη και συγχρονίζεται με τον πραγματικό DOM από τη βιβλιοθήκη όπως το ReactDOM. Λόγω αυτών των χαρακτηριστικών και οφελών, επιχειρήσεις όπως η Facebook, Netflix, Yahoo, Atlassian, Khan Academy, Pinterest, Dropbox και άλλες έχουν με επιτυχία μεταβεί στη χρήση του ReactJS.

Το εικονικό DOM (Document Object Model) είναι ένα έξυπνο σύστημα που χρησιμοποιείται σε ορισμένα πλαίσια εργασίας, όπως το ReactJS, για την αποδοτική διαχείριση των αλλαγών που πραγματοποιούνται σε μια διεπαφή χρήστη.

Στο κλασικό μοντέλο DOM, κάθε φορά που υπάρχει μια αλλαγή στα δεδομένα της εφαρμογής, η εφαρμογή πρέπει να ενημερώσει τον πραγματικό DOM. Αυτή η διαδικασία μπορεί να είναι αργή, ειδικά όταν πρόκειται για μεγάλες και πολύπλοκες διεπαφές χρήστη.

Το εικονικό DOM λειτουργεί ως μια ελαφριά αντιπροσωπεία του πραγματικού DOM. Κατά τη διάρκεια ενημερώσεων στα δεδομένα, το εικονικό DOM υπολογίζει τις αλλαγές που απαιτούνται στο πραγματικό DOM, αλλά δεν εφαρμόζει αμέσως αυτές τις αλλαγές. Αντίθετα, συγκρίνει τον εικονικό DOM με τον πραγματικό DOM και εντοπίζει τις διαφορές.

Αφού έχουν εντοπιστεί οι αλλαγές, αυτές εφαρμόζονται μαζικά στον πραγματικό DOM. Η προσέγγιση αυτή μειώνει τον αριθμό των αλλαγών που πραγματοποιούνται στον πραγματικό DOM, βελτιώνοντας την απόδοση της εφαρμογής. Επιπλέον, η χρήση του εικονικού DOM συμβάλλει στην αποφυγή περιττών επανασχεδιασμών της διεπαφής χρήστη και εξασφαλίζει αποδοτικότητα σε διαδραστικές εφαρμογές.

3.2.6: VueJS

Το VueJS είναι ένα framework της JavaScript για την κατασκευή διεπαφών χρήστη. Υλοποιείται πάνω σε πρότυπα HTML, CSS και JavaScript και παρέχει ένα δηλωτικό, βασισμένο σε συνιστώσες μοντέλο προγραμματισμού που βοηθά στην ανάπτυξη αποδοτικών διεπαφών χρήστη οποιασδήποτε πολυπλοκότητας. Το VueJS είναι προσανατολισμένο κυρίως στο front-end κομμάτι, αλλά μπορούμε να προσθέσουμε ό,τι χρειάζεται και να δημιουργήσουμε ισχυρές Προοδευτικές Εφαρμογές Ιστού (PWAs) με όλα τα εργαλεία που διαθέτει.[9]

Ο στόχος του κατασκευαστή του, Evan You, ήταν να δημιουργήσει ένα frontend πλαίσιο που να είναι τόσο ισχυρό όσο το Angular, αλλά και πιο «ελαφρύ» και ευέλικτο χωρίς όλα τα περιττά plugins και έννοιες που συνοδεύουν το Angular.[10]

Συνεπώς, ένα από τα βασικά πλεονεκτήματα του συγκεκριμένου framework είναι το πόσο “ελαφρύ” είναι, αφού το μέγεθός του περιορίζεται μόλις γύρω στα 20 KB. Δεν καταλαμβάνει καθόλου χώρο και δεν απαιτείται χρόνος αναμονής για να κατέβει και να εγκατασταθεί.

Επίσης, Το Vue.js χρησιμοποιεί την τεχνική της reactive data binding , η οποία διευκολύνει τη συγχρονισμένη ενημέρωση του UI με τις αλλαγές στα δεδομένα, βελτιώνοντας την εμπειρία χρήστη και μειώνοντας τον κώδικα που απαιτείται. [11]

Ένα ακόμη πλεονέκτημα, είναι το περιβάλλον του Vue.js, το οποίο περιλαμβάνει πολλά χρήσιμα εργαλεία και βιβλιοθήκες, όπως το Vue Router για δρομολόγηση, το Vuex για διαχείριση κατάστασης (state management), και το Vue CLI για γρήγορη εκκίνηση και ανάπτυξη έργων. [12]
Όσα αναφέρονται παραπάνω είναι μερικά από τα αρκετά πλεονεκτήματα του συγκεκριμένου πλαισίου. Συνολικά, το Vue.js είναι ένα ισχυρό εργαλείο που προσφέρει ευελιξία, απόδοση και αποδοτικότητα στην ανάπτυξη εφαρμογών.

3.3: Τεχνολογίες Ανάπτυξης Back – End Μέρους Διαδικτυακής Εφαρμογής

Το Back - End μέρος μιας διαδικτυακής εφαρμογής, που αποτελεί την πλευρά του server, λειτουργεί

παρασκηνιακά για την αποθήκευση και την οργάνωση των δεδομένων, ενώ παράλληλα εξασφαλίζει την ομαλή λειτουργία της πλευράς του client, χωρίς την άμεση αλληλεπίδραση με τον χρήστη. Οι δραστηριότητες του Back - End περιλαμβάνουν τη συγγραφή APIs, την ανάπτυξη βιβλιοθηκών και την αλληλεπίδραση με στοιχεία του συστήματος χωρίς διεπαφές χρήστη. Στο πεδίο της ανάπτυξης διαδικτυακών εφαρμογών, η τεχνολογία Back - End περιλαμβάνει συγκεκριμένες γλώσσες προγραμματισμού που είναι υπεύθυνες για την διαχείριση εργασιών όπως η αποθήκευση δεδομένων, ο έλεγχος ταυτότητας χρηστών και η παροχή API που είναι ζωτικής σημασίας για τη λειτουργικότητα του front-end, δηλαδή θεμελιώδεις πτυχές ενός δικτυακού τόπου ή μιας εφαρμογής.[13][14]

3.3.1: NodeJS

NodeJS: Το Node.js έχει κερδίσει δημοτικότητα στη γλώσσα σεναρίων της πλευράς του διακομιστή και προσφέρει ολοκληρωμένη ανάπτυξη πελάτη-διακομιστή, συμβάλλοντας στην επαναχρησιμοποίηση του κώδικα σε εφαρμογές ιστού, και είναι το ιδανικό εργαλείο για την ανάπτυξη γρήγορων, επεκτάσιμων δικτυακών εφαρμογών . Το Node.js είναι ένα πλαίσιο (framework) για την ανάπτυξη υψηλής απόδοσης, συναρτησιακών προγραμμάτων που δεν βασίζονται στην κλασική προσέγγιση της πολυνηματικότητας, αλλά χρησιμοποιούν ασύγχρονη εισόδου/εξόδου με ένα μοντέλο προγραμματισμού βασισμένο σε γεγονότα. Ο Kai Lei και συνεργάτες πραγματοποίησαν μια μελέτη για να συγκρίνουν την απόδοση του Node.js, του Python-Web και του PHP χρησιμοποιώντας δοκιμές μέτρησης και σεναριακές δοκιμές. Τα πειραματικά αποτελέσματα παρέχουν χρήσιμα δεδομένα απόδοσης, δείχνοντας ότι το PHP και το Python-Web χειρίζονται πολύ λιγότερα αιτήματα σε συγκεκριμένο χρονικό διάστημα σε σύγκριση με το Node.js. Η μελέτη αποδεικνύει ότι το Node.js είναι ελαφρύ και αποδοτικό, κατάλληλο για ιστοσελίδες που απαιτούν έντονη είσοδο/έξοδο, ενώ το PHP είναι κατάλληλο μόνο για μικρές και μεσαίες εφαρμογές, και το Python-Web είναι φιλικό προς τους προγραμματιστές και κατάλληλο για μεγάλες αρχιτεκτονικές ιστού

3.3.2: Java

Η Java αποτελεί μια γλώσσα προγραμματισμού και πλατφόρμα υπολογιστών που πρωτοκυκλοφόρησε από τη Sun Microsystems το 1995. Από τα πρώτα της βήματα, η εξέλιξή της την κατέστησε κινητήρια δύναμη πολλών υπηρεσιών και εφαρμογών στον ψηφιακό κόσμο της εποχής μας, παρέχοντας μια αξιόπιστη πλατφόρμα πάνω στην οποία βασίζονται πολλές υπηρεσίες και εφαρμογές. Προϊόντα και ψηφιακές υπηρεσίες που σχεδιάζονται για το μέλλον συνεχίζουν να βασίζονται στη Java για την εξέλιξή τους.[15]

Η Java είναι μια ελεύθερη και ευέλικτη γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία τοπικών και κατανεμημένων λογισμικών. Ορισμένες από τις συνηθισμένες χρήσεις της περιλαμβάνουν την ανάπτυξη παιχνιδιών, καθώς πολλά δημοφιλή παιχνίδια για κινητά, υπολογιστές και βιντεοπαιχνίδια κατασκευάζονται σε Java. Ακόμη και σύγχρονα παιχνίδια που ενσωματώνουν προηγμένες τεχνολογίες, όπως η μηχανική μάθηση ή η εικονική πραγματικότητα, κατασκευάζονται με τη βοήθεια της Java.

Επιπλέον, η Java είναι ιδανική για το υπολογιστικό νέφος, καθώς αναφέρεται συχνά ως WORA (Write Once and Run Anywhere), γεγονός που την καθιστά κατάλληλη για αποκεντρωμένες εφαρμογές στο νέφος. Οι πάροχοι υπηρεσιών νέφους επιλέγουν τη γλώσσα Java για την εκτέλεση προγραμμάτων σε ποικιλία υποκείμενων πλατφορμών. Στον τομέα των μεγάλων δεδομένων, η Java χρησιμοποιείται για μηχανές επεξεργασίας δεδομένων που μπορούν να διαχειρίζονται πολύπλοκα σύνολα δεδομένων και real-time data.

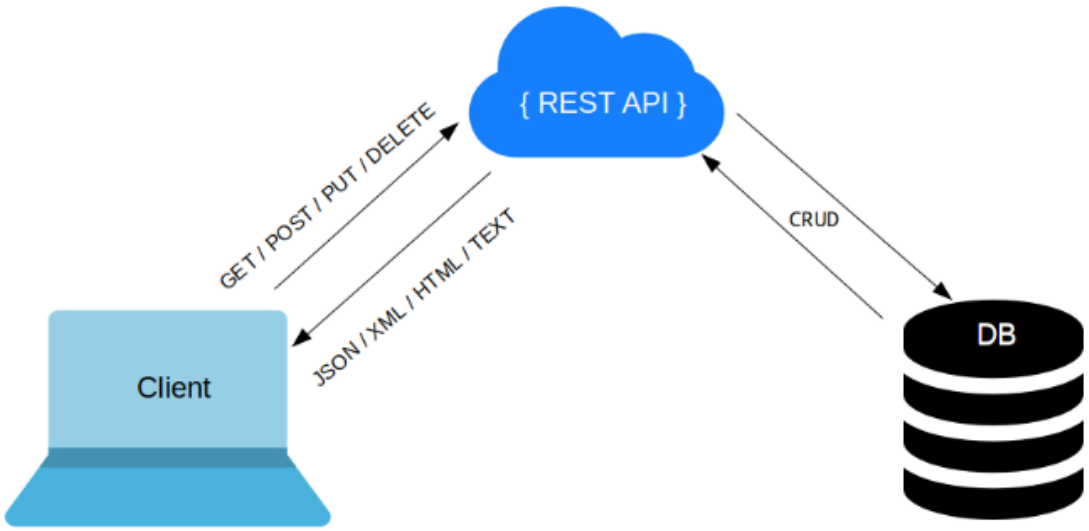
Αποτελεί επίσης σημαντικό πόρο για βιβλιοθήκες μηχανικής μάθησης, καθώς η σταθερότητά της και η ταχύτητά της την καθιστούν ιδανική για την ανάπτυξη εφαρμογών τεχνητής νοημοσύνης, όπως η επεξεργασία φυσικής γλώσσας και το deep learning. Τέλος, η Java έχει χρησιμοποιηθεί για τον προγραμματισμό αισθητήρων και hardware σε edge devices, τα οποία μπορούν να συνδέονται απομακρυσμένα στο Διαδίκτυο, συμβάλλοντας στην ανάπτυξη του Διαδικτύου των Πραγμάτων (IoT).

3.4: Διεπαφές Προγραμματισμού Εφαρμογών (APIs)

Οι Διεπαφές Προγραμματισμού Εφαρμογών (APIs) αντιπροσωπεύουν ένα κεντρικό στοιχείο στην ψηφιακή μας πραγματικότητα, επιτρέποντας δισεκατομμύρια ψηφιακές εμπειρίες κάθε λεπτό, κάθε ημέρα.

Ο όρος API σημαίνει Διεπαφή Προγραμματισμού Εφαρμογών και αποτελεί ένα λογισμικό ενδιάμεσο που δίνει τη δυνατότητα σε δύο εφαρμογές να επικοινωνούν μεταξύ τους. Αλλιώς, μια API αποτελεί τον μεσολαβητή που μεταφέρει το αίτημά μας στον πάροχο από τον οποίο θέτουμε το αίτημα και στη συνέχεια παραδίδει την απάντηση πίσω σε εμάς.

Οι Διεπαφές Προγραμματισμού Εφαρμογών (APIs) διαδραματίζουν κεντρικό ρόλο στις πρωτοβουλίες για ψηφιακή μετασχηματιστικότητα σε διάφορους κλάδους. Επιτρέπουν σε οργανισμούς να συνδέουν και να ενσωματώνουν διάφορα συστήματα, εφαρμογές και πηγές δεδομένων, προωθώντας την ομαλή ροή πληροφοριών και λειτουργικότητας. Οι API λειτουργούν ως γέφυρες μεταξύ ασυνάρτητων τεχνολογιών, επιτρέποντας στις επιχειρήσεις να εκμεταλλεύονται τη δύναμη των ψηφιακών οικοσυστημάτων και να ανοίγουν νέες προοπτικές. [17]



Σχήμα 2:Απεικόνιση λειτουργίας API

3.4.1: Τρόπος Λειτουργίας των API

Όταν δύο συστήματα (websites, desktops, smartphones) συνδέονται μέσω ενός API, λέμε ότι είναι "ενσωματωμένα" (integrated). Σε μια ενσωμάτωση, υπάρχουν δύο πλευρές, καθεμία με ένα ειδικό όνομα. Η μία πλευρά ονομάζεται διακομιστής (server). Αυτή είναι η πλευρά που προσφέρει πραγματικά το API. Βοηθάει να θυμόμαστε ότι το API είναι απλώς ένα άλλο πρόγραμμα που τρέχει στον διακομιστή. Μπορεί να ανήκει στο ίδιο πρόγραμμα που διαχειρίζεται την κίνηση στον ιστό, ή μπορεί να είναι εντελώς ξεχωριστό. Σε κάθε περίπτωση, εκείνο καθίσταται, περιμένοντας να του ζητήσουν δεδομένα.

Η άλλη πλευρά είναι ο "πελάτης" (client). Αυτό είναι ένα ξεχωριστό πρόγραμμα που γνωρίζει ποια δεδομένα είναι διαθέσιμα μέσω του API και μπορεί να τα χειριστεί, συνήθως κατόπιν αιτήματος ενός χρήστη. Ένα εξαιρετικό παράδειγμα είναι μια εφαρμογή smartphone που συγχρονίζεται με μια ιστοσελίδα. Όταν πατάμε το κουμπί ανανέωσης στην εφαρμογή, αυτή επικοινωνεί με ένα διακομιστή μέσω ενός API

και φέρνει τις πιο πρόσφατες πληροφορίες.

Η ίδια αρχή ισχύει και για ιστοσελίδες που είναι ενσωματωμένες. Όταν μια ιστοσελίδα ανακτά δεδομένα από μια άλλη, η ιστοσελίδα που παρέχει τα δεδομένα ενεργεί ως διακομιστής, και η ιστοσελίδα που τα ανακτά είναι ο πελάτης.[18]

3.4.1.2: Πλεονεκτήματα των APIs

Τα κυριότερα πλεονεκτήματα:

Applications: Η πρόσβαση σε APIs εξασφαλίζει μεγαλύτερη ευελιξία στις διαδικασίες μεταφοράς πληροφοριών.

Επικοινωνία: Τα APIs μας επιτρέπουν να δημιουργούμε στρώματα σε εφαρμογές προκειμένου να διανέμουμε πληροφορίες σε διαφορετικό κοινό.

Προσαρμογή: Επιπλέον, μπορεί να λειτουργήσει ως λύση για τη δημιουργία διαφορετικών εμπειριών για τους χρήστες, επιτρέποντας την προσαρμογή πρωτοκόλλων, λειτουργιών και εντολών σύμφωνα με συγκεκριμένες απαιτήσεις.

Αποτελεσματικότητα: Όταν έχουμε περιεχόμενο που δημοσιεύεται αυτόματα και είναι διαθέσιμο σε διάφορα κανάλια ταυτόχρονα, τα APIs επιτρέπουν πιο αποτελεσματική διανομή δεδομένων.

Προσαρμοστικότητα: Ένα από τα μεγαλύτερα πλεονεκτήματα των APIs είναι η ικανότητά τους να προσαρμόζονται σε αλλαγές μέσω της μεταφοράς δεδομένων και της ευελιξίας των υπηρεσιών.[19]

3.4.1.3: Μειονεκτήματα των APIs

Αν και τα APIs προσφέρουν σημαντικά πλεονεκτήματα στην ανάπτυξη και τη λειτουργία εφαρμογών, δεν παύουν να έχουν και κάποια μειονεκτήματα/περιορισμούς. Η χρήση των APIs ενδέχεται να συνοδεύεται από ορισμένα μειονεκτήματα λοιπόν, που αξίζει να αναφερθούν. Τα κυριότερα από αυτά είναι:

Έλλειψη αποτελεσματικής προστασίας έναντι απειλών: Τα APIs δεν προσφέρουν την ίδια προστασία με ένα Gateway, το οποίο λειτουργεί ως το κεντρικό σημείο ελέγχου για την πρόσβαση σε μια εφαρμογή ή ένα δίκτυο, προσφέροντας προστασία μέσω πολλαπλών επιπέδων ασφαλείας. Αντιθέτως, συνήθως παρέχουν απευθείας πρόσβαση σε συγκεκριμένες λειτουργίες ή δεδομένα της εφαρμογής μέσω καθορισμένων σημείων επαφής (endpoints) και δεν προσφέρουν πάντα το ίδιο επίπεδο συνολικής προστασίας ή ελέγχου πρόσβασης.

Επηρεάζουν και άλλες εφαρμογές: Εάν παραβιαστεί ένα API, μπορεί να επηρεάσει αρνητικά άλλες εφαρμογές που βασίζονται στο ίδιο API, και να τις εκθέσει σε επιθέσεις.

Σύνθετη μακροχρόνια διαχείριση: Παρά την ευκολία ανάπτυξής τους, τα APIs μπορούν να γίνουν πολύπλοκα στη διαχείριση με την πάροδο του χρόνου, καθώς απαιτούν συνεχή συντήρηση και παρακολούθηση.

Απαιτούμενη εξειδίκευση: Η σωστή διαχείριση και συντήρηση των APIs απαιτεί υψηλό επίπεδο τεχνικής γνώσης και εμπειρίας, που μπορεί να είναι δύσκολο να βρεθεί. [20]

Αυτά τα μειονεκτήματα απαιτούν προσεκτική διαχείριση και στρατηγική προσέγγιση.

3.4.1.4: Συμπέρασμα

Παρά τα αναμφισβήτητα μειονεκτήματα, τα συνολικά οφέλη των APIs υπερτερούν. Η ευελιξία που προσφέρουν, η ικανότητά τους να συνδέουν διάφορα συστήματα και η δυνατότητα για προσαρμογή και βελτίωση της αποδοτικότητας κάνουν τα APIs έναν εξαιρετικά πολύτιμο εργαλείο στην ανάπτυξη

σύγχρονων εφαρμογών. Με σωστή διαχείριση και κατάλληλα μέτρα ασφαλείας, τα πλεονεκτήματα των APIs ξεπερνούν τα μειονεκτήματά τους, καθιστώντας τα αναγκαία για την επιτυχία και την καινοτομία στον τεχνολογικό τομέα.

3.5: Βάσεις Δεδομένων Διαδικτυακών Εφαρμογών

Ως «δεδομένα» ορίζεται μια συλλογή μικρών μονάδων πληροφοριών, όπως κείμενα, αριθμοί, bytes κ.λπ., τα οποία μπορούν να αποθηκευτούν σε ηλεκτρονική μνήμη. Η «βάση δεδομένων» αναφέρεται σε μια οργανωμένη συλλογή δεδομένων, σχεδιασμένη έτσι ώστε να επιτρέπει εύκολη πρόσβαση, διαχείριση και τροποποίηση αυτών των δεδομένων. Τα δεδομένα συνήθως οργανώνονται σε πίνακες, σειρές και στήλες για να διευκολύνεται η αναζήτηση και η επεξεργασία τους[21].

Οι βάσεις δεδομένων διακρίνονται κυρίως σε δύο κατηγορίες:

SQL (Structured Query Language): Οι βάσεις δεδομένων SQL χρησιμοποιούν ένα οργανωμένο μοντέλο δεδομένων, συνήθως το μοντέλο Οντοτήτων – Σχέσεων, που βασίζεται σε πίνακες και σχέσεις μεταξύ τους. Αυτό το μοντέλο επιτρέπει την αυστηρή τήρηση των σχέσεων μεταξύ δεδομένων και παρέχει ισχυρές δυνατότητες αναζήτησης και συναλλαγών. Οι SQL βάσεις δεδομένων είναι ιδανικές για εφαρμογές που απαιτούν ακεραιότητα των δεδομένων και ισχυρές συναλλαγές, όπως τραπεζικές εφαρμογές και συστήματα ERP. Δημοφιλή παραδείγματα περιλαμβάνουν[22]:

- **MySQL:** Είναι μια από τις πιο δημοφιλείς βάσεις δεδομένων ανοιχτού κώδικα. Χρησιμοποιείται ευρέως σε ιστότοπους και εφαρμογές διαδικτύου λόγω της ευκολίας στη χρήση, της ταχύτητας και της ευελιξίας της. Είναι γνωστή για την υποστήριξη υψηλών ταχυτήτων ανάγνωσης και γραφής, γεγονός που την καθιστά κατάλληλη για εφαρμογές με υψηλό φόρτο εργασίας.

  Χαρακτηριστικά: Υποστηρίζει SQL, είναι κατάλληλη για εφαρμογές που απαιτούν βασικές λειτουργίες συναλλαγών και διαχείριση δεδομένων. Παρέχει υποστήριξη για διάφορους τύπους αποθήκευσης δεδομένων, όπως InnoDB και MyISAM.

  Χρήσεις: Ιστότοποι, εφαρμογές ηλεκτρονικού εμπορίου, blogs, CMS[23].

- **PostgreSQL:** Είναι μια ισχυρή, ανοιχτού κώδικα βάση δεδομένων, γνωστή για την επεκτασιμότητα και τη συμμόρφωσή της με τα πρότυπα SQL. Υποστηρίζει σύνθετες ερωτήσεις, πολυδιάστατους δείκτες, και ποικιλία τύπων δεδομένων, όπως JSONB για την αποθήκευση μη δομημένων δεδομένων.

  Χαρακτηριστικά: Προσφέρει ισχυρή υποστήριξη για ACID συναλλαγές, πλούσιο σύνολο χαρακτηριστικών για αναλυτικές ερωτήσεις, και προσαρμοσμένα extensions.

  Χρήσεις: Εφαρμογές που απαιτούν σύνθετους τύπους δεδομένων, γεωχωρικές αναλύσεις, και προσαρμοσμένες λειτουργίες.[24].

- **Oracle Database**: Είναι μια εμπορική βάση δεδομένων που προσφέρει υψηλή διαθεσιμότητα, ασφάλεια, και επεκτασιμότητα. Είναι κατάλληλη για μεγάλες και σύνθετες εφαρμογές που απαιτούν προηγμένα χαρακτηριστικά διαχείρισης δεδομένων.

  Χαρακτηριστικά: Παρέχει δυνατότητες όπως Oracle Real Application Clusters (RAC) για υψηλή διαθεσιμότητα, Data Guard για αναπαραγωγή δεδομένων, και υποστήριξη για μεγάλες επιχειρησιακές εφαρμογές.

  Χρήσεις: Μεγάλες επιχειρησιακές εφαρμογές, χρηματοοικονομικά συστήματα, διαχείριση αλυσίδας εφοδιασμού[25].

- **Microsoft SQL Server:** Ο Microsoft SQL Server είναι μια εμπορική βάση δεδομένων που ενσωματώνεται καλά με άλλα προϊόντα της Microsoft. Παρέχει πλήρη υποστήριξη για SQL και έχει χαρακτηριστικά που διευκολύνουν τη διαχείριση δεδομένων και τη δημιουργία αναφορών.

  Χαρακτηριστικά: Περιλαμβάνει εργαλεία για BI (Business Intelligence), ανάλυση δεδομένων, και υποστήριξη για τη δημιουργία και διαχείριση μεγάλων βάσεων δεδομένων.
  Χρήσεις: Εφαρμογές επιχειρησιακής ανάλυσης, ERP, CRM, και εφαρμογές διαχείρισης δεδομένων που ενσωματώνονται με περιβάλλοντα Microsoft[26].

NoSQL (Not Only SQL): Οι βάσεις δεδομένων NoSQL δεν ακολουθούν το παραδοσιακό μοντέλο των σχεσιακών βάσεων δεδομένων και παρέχουν ευέλικτες δομές για την αποθήκευση και διαχείριση των δεδομένων. Αυτές οι βάσεις είναι σχεδιασμένες για να υποστηρίζουν μεγάλες ποσότητες δεδομένων, να κλιμακώνονται οριζόντια και να παρέχουν υψηλή απόδοση και ευελιξία. Είναι κατάλληλες για εφαρμογές που απαιτούν δυναμική αποθήκευση δεδομένων ή επεξεργασία δεδομένων σε πραγματικό χρόνο. Δημοφιλή παραδείγματα περιλαμβάνουν[22]:

- **MongoDB:** Είναι μια βάση δεδομένων εγγράφων NoSQL που αποθηκεύει δεδομένα σε μορφή JSON (ή BSON, Binary JSON), επιτρέποντας ευέλικτη αποθήκευση και επεξεργασία μη δομημένων δεδομένων. Είναι ιδανική για εφαρμογές που χρειάζονται γρήγορη ανάπτυξη και ευέλικτους τύπους δεδομένων.

  Χαρακτηριστικά: Υποστηρίζει δυναμική κατασκευή ερωτήσεων, αναπαραγωγή και κατανομή δεδομένων, και έχει ισχυρές δυνατότητες για κλιμάκωση.

  Χρήσεις: Εφαρμογές που απαιτούν γρήγορη ανάπτυξη, ανάλυση μεγάλων δεδομένων, και επεξεργασία δεδομένων σε πραγματικό χρόνο[27].

.

- **CouchDB**: Είναι μια βάση δεδομένων εγγράφων που αποθηκεύει δεδομένα σε JSON και παρέχει χαρακτηριστικά replication και offline access. Εξαιρετική για εφαρμογές που χρειάζονται συνεπή αποθήκευση και συγχρονισμό δεδομένων.

  Χαρακτηριστικά: Υποστηρίζει αποθήκευση και ανακτήση εγγράφων, συνεπή replication, και εύκολη ενσωμάτωσή της με web εφαρμογές.

  Χρήσεις: Εφαρμογές με ανάγκη για offline λειτουργία, συνεπή συγχρονισμό δεδομένων, και διαχείριση εγγράφων. [28]

- **Cassandra**: Είναι μια κατανεμημένη βάση δεδομένων που προορίζεται για την υποστήριξη μεγάλων ποσοτήτων δεδομένων και ταυτόχρονων αναγνώσεων και εγγραφών. Εξαιρετική για περιβάλλοντα με υψηλές απαιτήσεις αποδοτικότητας και κλιμάκωσης.

  Χαρακτηριστικά: Υποστηρίζει horizontal scaling, υψηλή διαθεσιμότητα και αντοχή σε σφάλματα, καθώς και κατανομή δεδομένων χωρίς κεντρικό κόμβο.

  Χρήσεις: Εφαρμογές με υψηλό φόρτο δεδομένων, μεγάλα κοινωνικά δίκτυα, και συστήματα τηλεπικοινωνιών.[29]

- **Redis**: Είναι μια βάση δεδομένων in-memory που χρησιμοποιείται κυρίως ως cache και για

εφαρμογές που απαιτούν εξαιρετικά γρήγορη πρόσβαση σε δεδομένα. Υποστηρίζει επίσης δομές δεδομένων όπως λίστες, σύνολα, και hash tables.

Χαρακτηριστικά: Παρέχει υψηλή ταχύτητα και χαμηλή καθυστέρηση, υποστηρίζει persistence για δεδομένα που δεν είναι σε μνήμη, και δυνατότητες replication και clustering.

Χρήσεις: Cache, session management, real-time analytics, και message brokering[30].

3.5.1: Επιλογή Βάσης Δεδομένων

Η επιλογή της κατάλληλης βάσης δεδομένων για μια διαδικτυακή εφαρμογή εξαρτάται από τις ανάγκες και απαιτήσεις της εφαρμογής. Εάν η εφαρμογή απαιτεί μεγάλη ευελιξία, δυνατότητες κλιμάκωσης και χαμηλότερο κόστος συντήρησης, οι βάσεις δεδομένων NoSQL είναι συχνά η προτιμώμενη επιλογή. Αντίθετα, αν απαιτείται αξιόπιστη και συνεπής βάση δεδομένων με ισχυρές δυνατότητες συναλλαγών και κατακόρυφης κλιμάκωσης, οι βάσεις δεδομένων SQL συνήθως παρέχουν την καλύτερη λύση.

Η σωστή επιλογή της βάσης δεδομένων μπορεί να έχει σημαντικό αντίκτυπο στην απόδοση, την επεκτασιμότητα και τη συντήρηση της εφαρμογής. Επομένως, είναι σημαντικό να εξεταστούν προσεκτικά οι απαιτήσεις της εφαρμογής και να επιλεγεί η βάση δεδομένων που θα ανταποκριθεί καλύτερα σε αυτές τις ανάγκες.

3.6: Έλεγχος Ταυτότητας και Εξουσιοδότηση

Ο Έλεγχος Ταυτότητας (Authentication – AuthN) και η Εξουσιοδότηση (Authorization – AuthZ) αποτελούν θεμελιώδεις διαδικασίες ασφαλείας που χρησιμοποιούνται για την προστασία συστημάτων, δικτύων και πόρων από μη εξουσιοδοτημένη πρόσβαση. Αυτές οι διαδικασίες διασφαλίζουν ότι μόνο εξουσιοδοτημένοι χρήστες ή συσκευές μπορούν να έχουν πρόσβαση σε ευαίσθητες πληροφορίες ή να εκτελούν συγκεκριμένες λειτουργίες.

3.6.1: Έλεγχος Ταυτότητας (Authentication)

Ο Έλεγχος Ταυτότητας αφορά τη διαδικασία επαλήθευσης της ταυτότητας του χρήστη ή της συσκευής που προσπαθεί να αποκτήσει πρόσβαση σε ένα σύστημα. Είναι η διαδικασία κατά την οποία το σύστημα επαληθεύει την εγκυρότητα των διαπιστευτηρίων του χρήστη, όπως το όνομα χρήστη και ο κωδικός πρόσβασης.

Διαπιστευτήρια: Τα πιο κοινά διαπιστευτήρια περιλαμβάνουν ονόματα χρήστη και κωδικούς πρόσβασης, αλλά μπορεί να επεκταθούν σε βιομετρικά δεδομένα (π.χ., δακτυλικά αποτυπώματα, αναγνώριση προσώπου), πιστοποιητικά ασφαλείας, ή ακόμα και tokens δύο παραγόντων (2FA).

Μέθοδοι Ελέγχου Ταυτότητας:

- ***Single-Factor Authentication (SFA):*** Απαιτεί μόνο ένα σύνολο διαπιστευτηρίων, όπως κωδικό πρόσβασης.
- ***Two-Factor Authentication (2FA):*** Συνδυάζει δύο διαφορετικούς τύπους διαπιστευτηρίων, π.χ., κωδικό πρόσβασης και ένα SMS token.
- ***Multi-Factor Authentication (MFA):*** Χρησιμοποιεί περισσότερους από δύο παράγοντες για την επαλήθευση, όπως κωδικό πρόσβασης, βιομετρικά δεδομένα, και μια έξυπνη κάρτα.

Πρωτόκολλα Ελέγχου Ταυτότητας: Οι διαδικασίες ελέγχου ταυτότητας μπορούν να υλοποιηθούν μέσω διαφόρων πρωτοκόλλων, όπως το OAuth, το OpenID Connect, και το SAML (Security Assertion Markup Language), τα οποία παρέχουν ασφαλείς και τυποποιημένους τρόπους διαχείρισης ταυτοτήτων σε διαφορετικά συστήματα και υπηρεσίες.

3.6.2: Εξουσιοδότηση (Authorization)

Η Εξουσιοδότηση αφορά τον καθορισμό των δικαιωμάτων πρόσβασης που επιτρέπονται στον χρήστη ή στη συσκευή, αφού έχει ολοκληρωθεί ο έλεγχος ταυτότητας. Η εξουσιοδότηση ορίζει τα δικαιώματα και τους περιορισμούς που έχουν ανατεθεί σε κάθε χρήστη.

Μοντέλα Εξουσιοδότησης:

- ***Discretionary Access Control (DAC):*** Το δικαίωμα πρόσβασης καθορίζεται από τον ιδιοκτήτη των δεδομένων ή των πόρων.
- ***Mandatory Access Control (MAC):*** Οι κανόνες πρόσβασης καθορίζονται από την πολιτική ασφαλείας του συστήματος και όχι από τον χρήστη.
- ***Role-Based Access Control (RBAC):*** Οι χρήστες ανατίθενται σε συγκεκριμένους ρόλους, και κάθε ρόλος έχει συγκεκριμένα δικαιώματα πρόσβασης.
- ***Attribute-Based Access Control (ABAC):*** Η πρόσβαση βασίζεται σε συνδυασμό χαρακτηριστικών του χρήστη και του πόρου (π.χ., ιδιότητες χρήστη, περιεχόμενο πόρου, κατάσταση του περιβάλλοντος).

Πολιτικές και Κανόνες: Οι πολιτικές εξουσιοδότησης καθορίζουν τι επιτρέπεται ή όχι σε κάθε χρήστη βάσει του ρόλου του, της τοποθεσίας του, της ώρας της ημέρας, και άλλων παραγόντων.

Συνήθη Πρωτόκολλα Εξουσιοδότησης: Όπως το OAuth, που επιτρέπει την ασφαλή ανάθεση δικαιωμάτων πρόσβασης σε τρίτες εφαρμογές χωρίς να αποκαλύπτονται τα διαπιστευτήρια του χρήστη.

3.6.3: Ολοκλήρωση και Συνεργασία AuthN και AuthZ

Σε ένα πλήρες σύστημα ασφαλείας, ο έλεγχος ταυτότητας (AuthN) προηγείται της εξουσιοδότησης (AuthZ). Αρχικά, το σύστημα επαληθεύει την ταυτότητα του χρήστη μέσω του ελέγχου ταυτότητας και στη συνέχεια καθορίζει τα δικαιώματα πρόσβασης μέσω της εξουσιοδότησης. Αυτή η προσέγγιση διασφαλίζει ότι μόνο χρήστες που έχουν ταυτοποιηθεί και πιστοποιηθεί μπορούν να εκτελούν συγκεκριμένες ενέργειες ή να έχουν πρόσβαση σε συγκεκριμένες πληροφορίες[31].

3.6.4: Σημασία και Επιπτώσεις

Η σωστή υλοποίηση του ελέγχου ταυτότητας και της εξουσιοδότησης είναι κρίσιμη για την προστασία των συστημάτων από παραβιάσεις ασφαλείας. Οι οργανισμοί πρέπει να σχεδιάζουν τα συστήματά τους με ισχυρά μέτρα ελέγχου ταυτότητας και εξουσιοδότησης για να διασφαλίζουν την προστασία ευαίσθητων δεδομένων, τη συμμόρφωση με τους κανονισμούς και την εμπιστοσύνη των χρηστών.

Κεφάλαιο 4ο: Υλοποίηση Front – End Μέρους Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων

Εφόσον έχουμε παρουσιάσει αναλυτικά τις τεχνολογίες και τις μεθόδους που είναι διαθέσιμες για την ανάπτυξη του Front – End μέρους μίας Διαδικτυακής Εφαρμογής, στο συγκεκριμένο κεφάλαιο θα περιγράψουμε την διαδικασία επιλογής της καταλληλότερης διαθέσιμης τεχνολογίας για την υλοποίηση του Front - End και θα δούμε αναλυτικά τη διαδικασία της ανάπτυξης της Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων.

4.1 Επιλογή Τεχνολογίας Ανάπτυξης Front – End

Όπως αναφέρθηκε στο κεφάλαιο 2.2, το framework AngularJS όπως και οι βιβλιοθήκες ReactJS και NextJS, αποτελούν τις τεχνολογίες που χρησιμοποιούνται για την υλοποίηση του Front - End στις σύγχρονες Διαδικτυακές Εφαρμογές.

Για την επιλογή της πλέον καταλληλότερης – από τις προαναφερθείσες τρεις – για την ανάπτυξη του Front – End μέρους της Διαδικτυακής εφαρμογής αναζήτησης και κρατήσεων ξενοδοχειακών καταλυμάτων, θα πρέπει να πληροί ορισμένες προϋποθέσεις για τη διευκόλυνση και του σχεδιαστή – προγραμματιστή, αλλά και των τελικών χρηστών της Εφαρμογής. Οι προϋποθέσεις είναι:

- Να μπορεί να δημιουργεί διατηρήσιμο κώδικα, ο οποίος να είναι απλός και εύκολος στη σύνταξη, ανάγνωση, δοκιμή και τροποποίησή του.
- Να προσφέρει τη δυνατότητα επαναχρησιμοποίησης στοιχείων, και επομένως, να εξαλείφει τη σύνταξη εκτεταμένου κώδικα.
- Να δημιουργεί συνεπών, αποτελεσματικές, διαδραστικές και αισθητικά όμορφες Διεπαφές Χρήστη.
- Να διαθέτει ενεργή κοινότητα προγραμματιστών, η οποία βελτιώνει τα ήδη υπάρχοντα εργαλεία και τις βιβλιοθήκες της ή ακόμη αναπτύσσει και προσφέρει νέα.

Λαμβάνοντας υπόψη τα παραπάνω, επιλέξαμε τη Βιβλιοθήκη React της JavaScript, καθώς:

- Βασίζεται στις γλώσσες HTML, CSS και JavaScript.
- Βασίζεται σε Στοιχεία (Components), η λογική των οποίων γράφεται σε JavaScript, καθιστώντας επαναχρησιμοποιήσιμο το εκάστοτε Στοιχείο και τον τελικό κώδικα συντομότερο.
- Διαθέτει ποικίλες επεκτάσεις, οι οποίες μπορούν να χρησιμοποιηθούν για τη δημιουργία μίας πλήρως αποτελεσματικής εφαρμογής Διεπαφής Χρήστη.
- Διαθέτει μία τεράστια κοινότητα προγραμματιστών και εταιρειών παγκοσμίως, η οποία δημιουργεί βιβλιοθήκες και περαιτέρω εργαλεία.

4.2 Δημιουργία Front – End Μέρους Διαδικτυακής Εφαρμογής Ηλεκτρονικών Ξενοδοχειακών Κρατήσεων

Για τη δημιουργία του Front – End της Διαδικτυακής Εφαρμογής χρησιμοποιείται η Βιβλιοθήκη React της JavaScript. Έτσι, επιτρέπεται στους χρήστες να καταχωρούν, να ανανεώνουν και να επεξεργάζονται στοιχεία. Οι σελίδες που δημιουργούνται και χρησιμοποιούνται είναι οι εξής παρακάτω:

1. Login Χρήστη,
2. Αρχική Σελίδα,
3. Register Νέου Χρήστη (μόνο για τον Admin και τον Owner),
4. Λίστα Χρηστών (μόνο για τον Admin),

5. Λίστα Υπαλλήλων (μόνο για τον Owner)
6. Προβολή στοιχείων χρήστη και ανανέωσή του κωδικού του (Update) (μόνο για τον Admin και τον Owner),
7. Δημιουργία Νέου Υπαλλήλου,
8. Προβολή Λίστας Ξενοδοχείων με Αναζήτηση (μόνο για τον Admin και τον Owner),
9. Προβολή Λίστας Δωματίων με Αναζήτηση (μόνο για τον Admin και τον Owner),
10. Προβολή Λίστας κρατήσεων με Αναζήτηση (μόνο για τον Admin και τον Owner),
11. Προβολή Λεπτομερειών Συνδρομής (μονο για τον Owner),
12. Δημιουργία Ξενοδοχείου (μόνο για τον Admin και τον Owner),
13. Δημιουργία Δωματίου (μόνο για τον Admin και τον Owner),
14. Προβολή Στοιχείων Ξενοδοχείου

Αρχικά, για τη δημιουργία της React εφαρμογής μας, εκτελούμε την εντολή "npx create-react-app". Στη συνέχεια, στο αρχείο index.js, περιστοιχίζουμε την εφαρμογή μας με έναν Provider, που παρέχεται από τη βιβλιοθήκη του Redux, περνώντας παραμετρικά το δημιουργημένο Store, όπως φαίνεται στο Σχήμα 3.1. Με αυτόν τον τρόπο, έχουμε πρόσβαση στο State του Redux σε ολόκληρη την εφαρμογή μας.

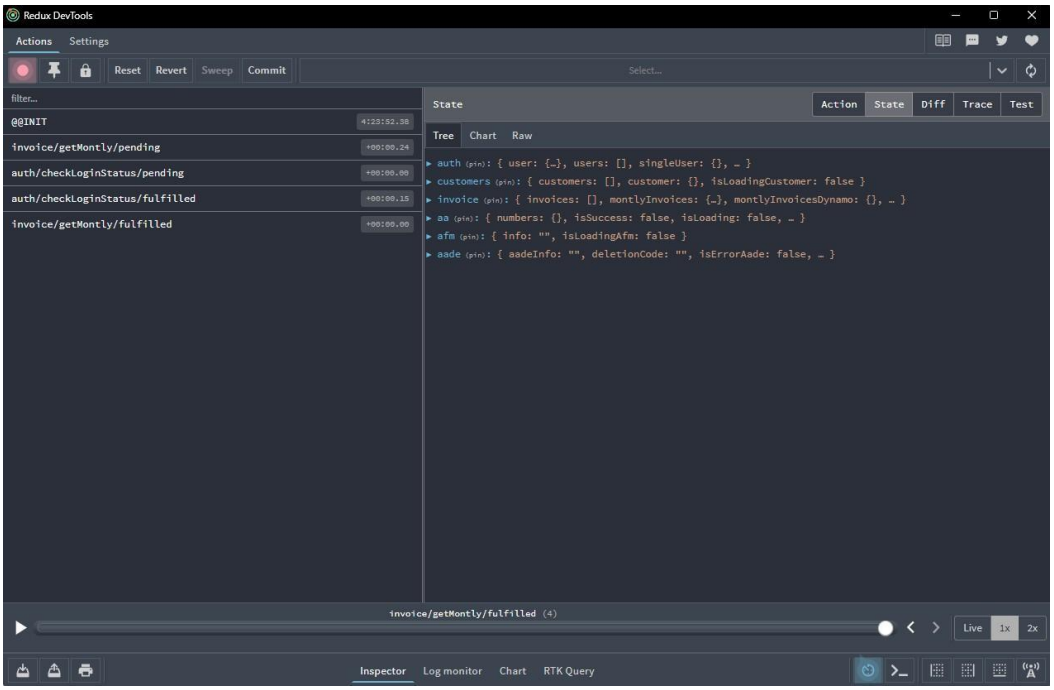
```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import './style.scss'
import store from './app/store'
import { Provider } from 'react-redux'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
)
```

Σχήμα 3:Πρόσβαση στο State του Redux

```
import React from 'react'
import ReactDOM from 'react-dom'
import { Provider } from 'react-redux'
import { store } from './app/store'
import './index.css'
import App from './App'

ReactDOM.render(
    <React.StrictMode>
        <Provider store={store}>
            <App />
        </Provider>
    </React.StrictMode>,
    document.getElementById('root')
)
```

Σχήμα 4:Χρήση του Provider της Βιβλιοθήκης Redux

Χρησιμοποιώντας την επέκταση του Redux για τον Chrome, είναι δυνατόν να παρακολουθούμε την κατάσταση του State ανά πάσα στιγμή, όπως παρουσιάζεται στο Σχήμα . Επιπλέον, με τη χρήση της βιβλιοθήκης του Redux, και συγκεκριμένα των Hooks "useDispatch" και "useSelector", μπορούμε να αλληλεπιδρούμε με το Back-End μας, καθώς επίσης να ανακτούμε τιμές από μεταβλητές που αποθηκεύονται globally στο State του Redux.



Σχήμα 5:Παρακολούθηση της κατάστασης του State.

Για την διαδικασία της περιήγησης στην Διαδικτυακή Εφαρμογή έχει χρησιμοποιηθεί το πακέτο React - Router - DOM, συγκεκριμένα η έκδοση 6.18.0. Τα βασικά σχέδια της Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων είναι τρία. Το πρώτο αφορά τις σελίδες “Σύνδεση Χρήστη” και “Δημιουργία Λογαριασμού”, το δεύτερο είναι για τις σελίδες που χρησιμοποιούνται για την διαχείριση των κρατήσεων

και το τελευταίο για τις σελίδες που περιέχουν την δημιουργία και την προβολή της κράτησης του χρήστη. Για το δεύτερο σχέδιο αξιοποιήθηκε ένα νέο χαρακτηριστικό του React – Router – DOM που ονομάζεται Outlet. Με την χρήση του Outlet γίνεται σταθερά render το στοιχείο Navbar και αυτό που γίνεται render δυναμικά, μέσω του περιεχομένου του Outlet, είναι το στοιχείο που διαλέγει ο χρήστης, δηλαδή η εκάστοτε σελίδα.

Σχήμα 6:Χρήση Outlet

Για να μεταβληθεί το περιεχόμενο του Outlet γίνεται import και χρήση του “CreateBrowserRouter” Component του React - DOM.

```
const router = createBrowserRouter([
    {
        path: '/',
        element: <ProtectedRoutes />,
        children: [
            {
                path: '/',
                element: <Home />,
            },
            {
                path: '/my-account',
                element: <MyAccount />,
            },
            {
                path: '/all-users',
                element: <GetAllUsers />,
            },
            {
                path: '/my-users',
                element: <GetMyUsers />,
            },
            {
                path: '/user/new',
                element: <AddNewUser />,
            },
            {
                path: '/user/:id',
                element: <GetSingleUser />,
            },
            {
                path: '/all-hotels',
                element: <GetAllHotels />,
            },
            {
                path: '/my-hotels',
                element: <GetMyHotels />,
            },
            {
                path: '/hotel/new',
                element: <AddNewHotel />,
            },
            {
                path: '/hotel/:id',
                element: <SingleHotel />,
            },
```

Σχήμα 7:Χρήση του CreateBrowserRouter

Αυτό υποδηλώνει πως κάθε Path που έχουμε ορίσει αποτελεί Παιδί (Child) του Outlet. Κατά συνέπεια, αναλόγως με το Path που βρισκόμαστε γίνεται render το αντίστοιχο Component. Για παράδειγμα, έαν βρισκόμαστε στην “Προβολή Στοιχείων Ξενοδοχείου” , δηλαδή είμαστε στο Path “ /hotel/:id”, θα απεικονιζέται το Component “SingleHotel”.

Το Component PrivateRoute δέχεται ως παράμετρο το children, το οποίο αντιστοιχεί στην σελίδα που θα γίνει render. Καθώς γίνεται render με την βοήθεια του Hook “useEffect()”, γίνεται ένα αίτημα “GET” στην διεύθυνση “/api/user/showMe”, το οποίο έχει σαν παράμετρο το HttpOnly cookie που περιέχει τα στοιχεία του χρήστη. Εφόσον το αίτημα είναι έγκυρο, επιστρέφει ένα αντικείμενο με τα στοιχεία του χρήστη το οποίο αποθηκεύεται στο local storage.Έπειτα, το Component ελέγχει εάν υπάρχει χρήστης στο local

storage ώστε να προσδιορίσει την κατάσταση της σύνδεσης του χρήστη.

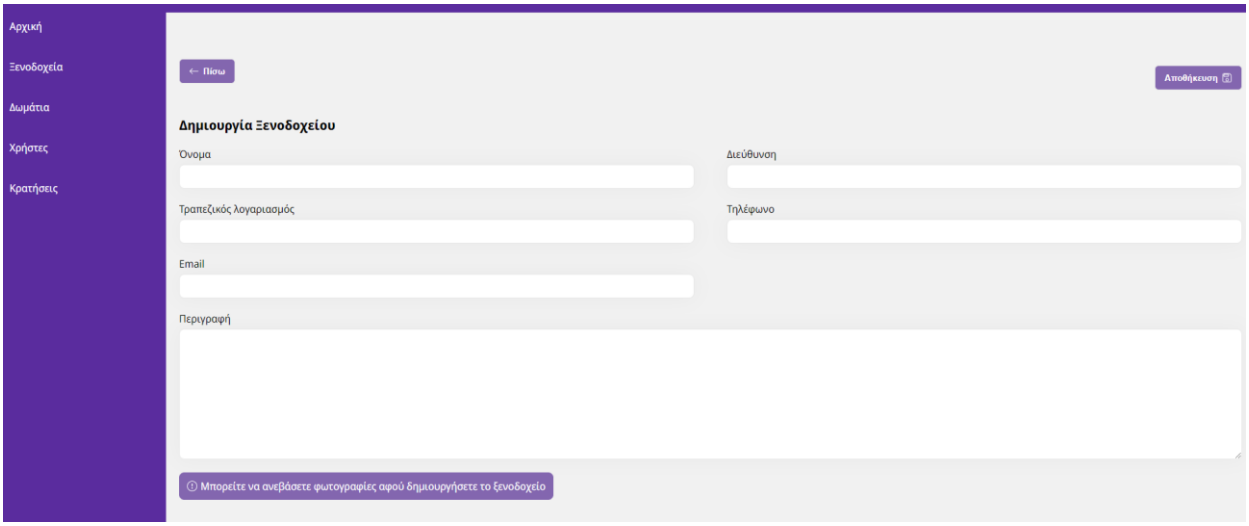
Στην συνέχεια, έχοντας τα στοιχεία του χρήστη, γίνεται έλεγχος για το αν έχει ενεργό subscription. Εάν το subscription του χρήστη είναι έγκυρο τότε προχωρά στην περιήγηση στην Εφαρμογή αλλιώς γίνεται ανακατεύθυνση στην σελίδα "Δημιουργίας/Ανανέωσης Συνδρομής".

```
function PrivateRoute({ children }) {

   let user = JSON.parse(localStorage.getItem('user'))

   const dispatch = useDispatch()
   const location = useLocation()

   useEffect(() => {
      dispatch(checkLoginStatus())
   }, [])

   if (user?.subscription === 'none') return <Navigate to='/subscription' state={{ prevUrl: location.pathname }} />

   if (user?.subscription === 'invalid')
      return <Navigate to='/renew-subscription' state={{ prevUrl: location.pathname }} />

   if (user?.subscription === 'valid') return children

   return <Navigate to='/login' state={{ prevUrl: location.pathname }} />
}

export default PrivateRoute
```

Σχήμα 8:Έλεγχος εγκυρότητας subscription

4.3 Σελίδα «Δημιουργία Ξενοδοχείου»

Όπως φαίνεται στο Σχήμα 3.7, η σελιδα “Δημιουργία Ξενοδοχείου” αποτελείται από μια φόρμα όπου ο εκάστοτε ιδιοκτήτης συμπληρώνει το Όνομα του Ξενοδοχείου,έναν Τραπεζικό Λογαριασμό, την Διεύθυνση, το τηλέφωνο, το email και την Περιγραφή.



Σχήμα 9:Σελίδα Δημιουργίας Ξενοδοχείου

```
function AddNewHotel() {
  const { user, isLoading } = useSelector((state) => state.auth)

  if (isLoading) return <Spinner />

  if (user.role === 'employee') return <Navigate to='/not-found' />

  return (
    <div className='create-hotel main-page'>
      <CreateHotel />
      <div className='informative-message'>
        <BsExclamationCircle color='#fff' /> Μπορείτε να ανεβάσετε φωτογραφίες αφού δημιουργήσετε το ξενοδοχείο
      </div>
    </div>
  )
}
```

Σχήμα 10:Εμφάνιση του Create Hotel Component

Με την χρήση του Hook useSelector() παίρνουμε τα στοιχεία του χρήστη που βρίσκονται αποθηκευμένα στο Redux και στην συνέχεια ελέγχουμε τον ρόλο του συγκεκριμένου χρήστη. Εάν ο χρήστης έχει τον ρόλο του υπαλλήλου ανακατευθύνεται στην σελίδα “Not Found”. Αντιθέτως, εάν ο ρόλος του χρήση δεν είναι αυτός του υπαλλήλου, γίνεται render το Component “CreateHotel” που είναι υπεύθυνο για την δημιουργία νέου ξενοδοχείου και εμφανίζεται και το σχετικό μήνυμα για την αποθήκευση των φωτογραφιών του ξενοδοχείου.

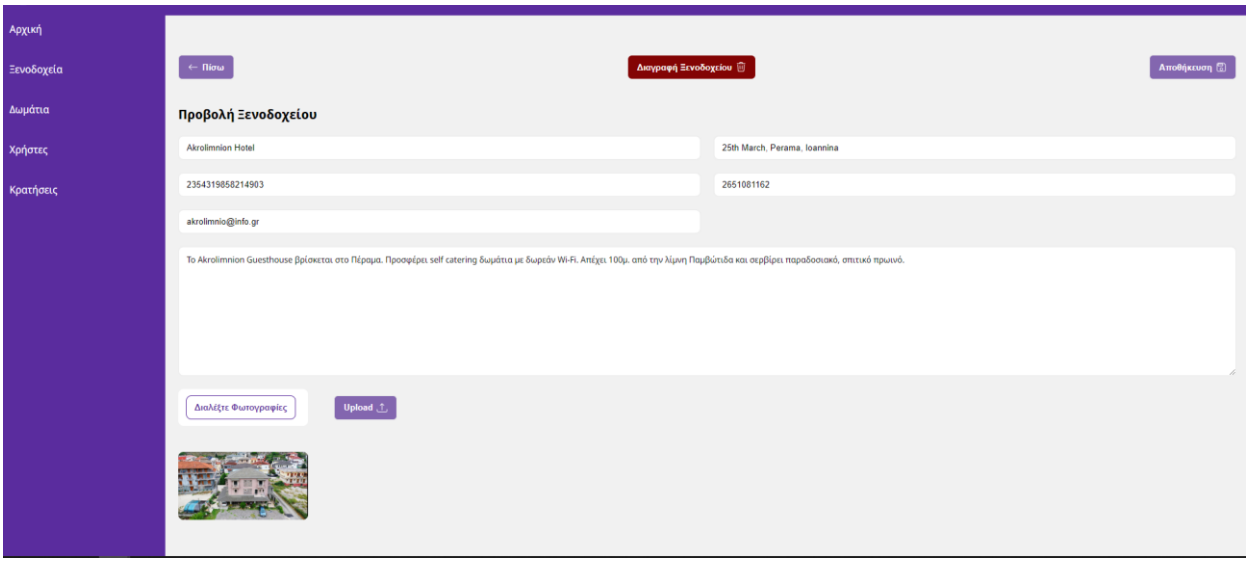
```
// Submit the hotel data
const submitHotelData = (e) => {
    e.preventDefault()

    const { name, description, email, phone, address } = formData
    dispatch(createHotel({ name, description, email, phone, address }))
        .unwrap()
        .then((res) => {
            toast.success('Το ξενοδοχείο δημιουργήθηκε επιτυχώς')
            navigate(`/hotel/${res._id}`)
        })
        .catch((error) => toast.error(error))
}
```

Σχήμα 11:Υλοποίηση της submitHotelData() function

Το συγκεκριμένο Component αποτελείται από μια φόρμα εισαγωγής των στοιχείων που έχουν περιγραφεί άνωθεν και της function “submitHotelData()”. Με την βοήθεια του Hook useDispatch() δημιουργείται ένα αίτημα POST στην διεύθυνση “api/hotels/createHotel”. Εάν το αίτημα είναι επιτυχές, τότε εμφανίζεται σχετικό μήνυμα και ο χρήστης ανακατευθύνεται στην σελίδα με τα στοιχεία του ξενοδοχείου που του ανήκουν, εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους.

4.4 Σελίδα «Επεξεργασία Ξενοδοχείου»

Στην σελίδα “Επεξεργασία Ξενοδοχείου”, εμφανίζονται τα στοιχεία του ξενοδοχείου, δηλαδή το Όνομα, η Διεύθυνση,τον Τραπεζικό Λογαριασμό, το τηλέφωνο, το email και η Περιγραφή, καθώς και η επιλογή να ανέβει μια φωτογραφία του ξενοδοχείου.



Σχήμα 12:Σελίδα Επεξεργασίας των στοιχείων του ξενοδοχείου

```
function SingleHotel() {
    const { id } = useParams()
    const { user } = useSelector((state) => state.auth)
    const { hotel, isLoading } = useSelector((state) => state.hotels)

    const dispatch = useDispatch()

    useEffect(() => {
        dispatch(getSingleHotel(id))
            .unwrap()
            .then((res) => {})
            .catch((error) => toast.error(error))
    }, [])

    if (isLoading) return <Spinner />

    return (
        <div className='single-hotel main-page'>
            {user.role === 'employee' ? <DisplayHotel hotel={hotel} /> : <UpdateHotel hotel={hotel} />}
        </div>
    )
}
```

Σχήμα 13: Σύνταξη συνάρτησης “getSingleHotel()”

Με την χρήση του Hook useParams() συλλέγουμε το id του ξενοδοχείου από το URL της εφαρμογής. Στη συνέχεια με την χρήση του Hook useSelector() παίρνουμε τα στοιχεία του χρήστη που βρίσκονται αποθηκευμένα στο Redux. ‘Επειτα, με τη χρήση του Hook useDispatch() δημιουργούμε ένα αίτημα GET στη διεύθυνση api/hotels/getSingleHotel?id= έχοντας σαν query το id του ξενοδοχείο. Ελέγχουμε τον ρόλο του συγκεκριμένου χρήστη. Εάν ο χρήστης έχει τον ρόλο του υπαλλήλου ανακατευθύνεται στην σελίδα “Display Hotel”, όπου μπορεί μόνο να δει τις πληροφορίες του συγκεκριμένου ξενοδοχείου. Αντιθέτως, εάν ο ρόλος του χρήση δεν είναι αυτός του υπαλλήλου, γίνεται render το Component “updateHotel” που είναι υπεύθυνο για την επεξεργασία των στοιχείων του ξενοδοχείου.

```
const updateHotelInfo = (e) => {
    e.preventDefault()

    const { name, email, phone, address, description } = formData

    dispatch(
        updateHotel({
            name,
            email,
            phone,
            address,
            description,
            imgs: hotelImages,
            createdBy: hotel.owner.owner_id,
            id: hotel._id,
        })
    )
        .unwrap()
        .then((res) => {
            toast.success('Το ξενοδοχείο ενημερώθηκε')
            window.location.reload()
        })
        .catch((error) => toast.error(error))
}
```

Σχήμα 14: Σύνταξη συνάρτησης “updateHotelInfo()”

```
const uploadImages = async (e) => {
    e.preventDefault()

    if (imagesForUpload.length === 0) {
        return toast.error('Παρακαλώ διαλέξτε φωτογραφίες')
    }

    await dispatch(uploadHotelImages({ imgs: imagesForUpload, hotelId: id }))
        .unwrap()
        .then((res) => {
            toast.success('Οι φωτογραφίες ανέβηκαν επιτυχώς')
        })
        .catch((error) => toast.error(error))
}
```

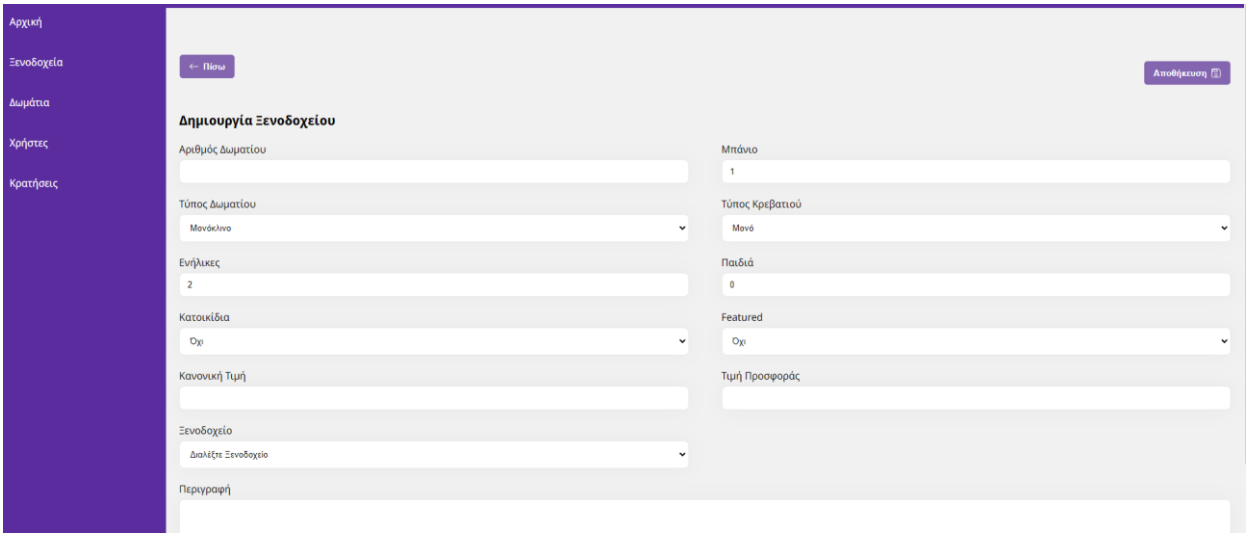
Σχήμα 15: Σύνταξη συνάρτησης “uploadImages()”

Το συγκεκριμένο Component αποτελείται από μια φόρμα επεξεργασίας των στοιχείων του ξενοδοχείου καθώς και τη δυνατότητα αποθήκευσης φωτογραφιών. Αφου ο χρήστης κάνει κλικ στο κουμπί “Διαλέξτε Φωτογραφίες” του εμφανίζεται ένα dialog που μπορεί να επιλέξει μια φωτογραφία. Αφού επιλέξει φωτογραφία, θα πρέπει να κάνει κλικ στο κουμπί “Upload”, το οποίο, καλεί την function uploadImages, η οποία, ελέγχει αν ο χρήστης έχει επιλέξει φωτογραφία και με την βοήθεια του Hook useDispatch()

δημιουργείται ένα αίτημα POST στην διεύθυνση "/api/hotels/uploadHotelImages?id= 'εχοντας σαν query το id του ξενοδοχείου και σαν παράμετρο τη φωτογραφία. Εάν το αίτημα είναι επιτυχές, τότε εμφανίζεται σχετικό μήνυμα. Εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους. Στη συνέχεια, ο χρήστης εαν θέλει επεξεργάζεται τα στοιχεία του ξενοδοχείου και πατώντας το κουμπί "Αποθήκευση" καλείται η συνάρτηση updateHotelInfo, η οποία στο Hook useDispatch() δημιουργεί ένα αίτημα PATCH στη διεύθυνση "/api/hotels/updateHotel" έχοντας σαν παράμετρο τα επεξεργασμένα στοιχεία του ξενοδοχείου και τη φωτογραφία. Εάν το αίτημα είναι επιτυχές, τότε εμφανίζεται σχετικό μήνυμα ,ενώ εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους.

4.5 Σελίδα «Δημιουργία Δωματίου»

Όπως φαίνεται στο Σχήμα, η σελίδα "Δημιουργία Δωματίου" αποτελείται από μια φόρμα εισαγωγής στοιχείων όπου ο χρήστης έχει την δυνατότητα συμπλήρωσης του Αριθμού Δωματίου,του Τύπου Δωματίου, τον μέγιστο αριθμό Ενηλίκων και Παιδιών που μπορούν να μείνουν στο συγκεκριμένο δωμάτιο, τον αριθμό των Μπάνιων του δωματίου, τον Τύπο Κρεβατιού, εάν επιτρέπονται Κατοικίδια, την Κανονική Τιμή του δωματίου και την Τιμή Προσφοράς, εφόσον υπάρχει, όπως επίσης αν έχει την ιδιότητα "Featured" και τέλος το Ξενοδοχείο στο οποίο ανήκει.



Σχήμα 16: Σελίδα δημιουργίας δωματίου

```
function AddNewRoom() {
    const { user, isLoading } = useSelector((state) => state.auth)

    if (isLoading) return <Spinner />

    if (user.role === 'employee') return <Navigate to='/not-found' />

    return (
        <div className='create-hotel main-page'>
            <CreateRoom role={user.role} />
        </div>
    )
}

export default AddNewRoom
```

Σχήμα 17: Σύνταξη συνάρτησης “addNewRoom()”

Με την χρήση του Hook useSelector() παίρνουμε τα στοιχεία του χρήστη που βρίσκονται αποθηκευμένα στο Redux και στην συνέχεια ελέγχουμε τον ρόλο του συγκεκριμένου χρήστη. Εάν ο χρήστης έχει τον ρόλο του υπαλλήλου ανακατευθύνεται στην σελίδα “Not Found”. Αντιθέτως, εάν ο ρόλος του χρήση δεν είναι αυτός του υπαλλήλου, γίνεται render το Component “CreateRoom” που είναι υπεύθυνο για την δημιουργία νέου δωματίου.

```
// Submit the hotel data
const submitRoomData = (e) => {
    e.preventDefault()

    const { roomNumber,description,roomType,bedType,bathroom,petsAllowed,regularPrice,discountPrice,hotelId,isFeatured,adults,children,} = formData

    if (roomNumber === '') { return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (description === '') { return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (roomType === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (bedType === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (bathroom === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (petsAllowed === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (regularPrice === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (isFeatured === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (hotelId === '' && role !== 'employee') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (adults === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}
    if (children === '') {return toast.error('Συμπληρώστε όλα τα πεδία')}

    const people = { adults, children }

    dispatch(
        createRoom({
            roomNumber,
            description,
            roomType,
            bedType,
            bathroom,
            petsAllowed,
            regularPrice,
            discountPrice,
            hotelId,
            isFeatured,
            people,
        })
    )
        .unwrap()
        .then((res) => {
            toast.success('Το δωμάτιο δημιουργήθηκε επιτυχώς')
            navigate(`/room/${res._id}`)
        })
        .catch((error) => toast.error(error))
}
```

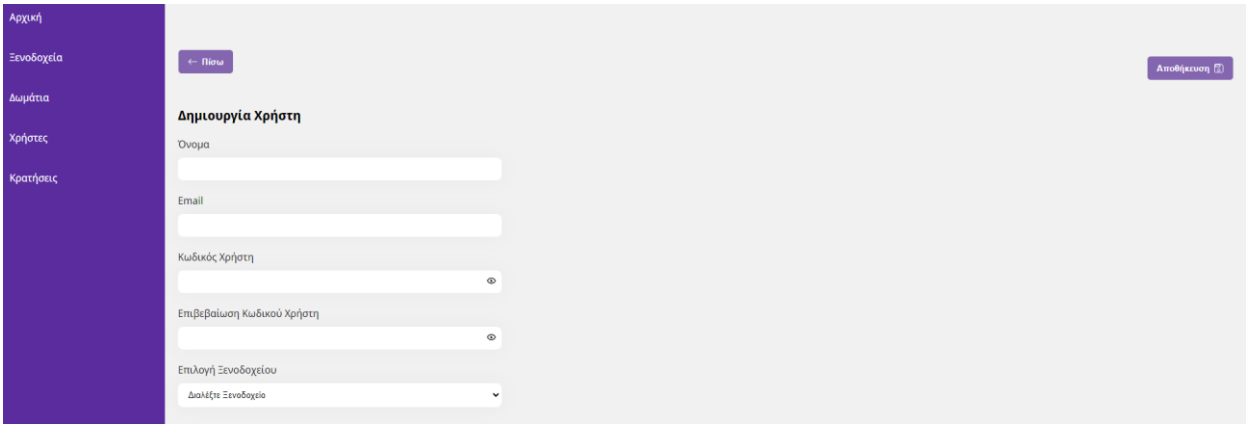
Σχήμα 18: Σύνταξη συνάρτησης “submitRoomData()”

Το συγκεκριμένο Component αποτελείται από μια φόρμα εισαγωγής των στοιχείων έχουν περιγραφεί άνωθεν και της function “submitRoomData()”. Στην οποία, αρχικά γίνεται έλεγχος της ορθότητας των στοιχείων εισαγωγής και στην συνέχεια με την βοήθεια του Hook useDispatch() δημιουργείται ένα αίτημα

POST στην διεύθυνση “api/rooms/createRoom” έχοντας σαν παράμετρο τα στοιχεία του δωματίου. Εάν το αίτημα είναι επιτυχές, τότε εμφανίζεται σχετικό μήνυμα και ο χρήστης ανακατευθύνεται στην σελίδα του συγκεκριμένου δωματίου, εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους.

4.6 Σελίδα «Δημιουργία Υπαλλήλου»

Όπως φαίνεται στο Σχήμα, η σελίδα «Δημιουργία Υπαλλήλου» αποτελείται από μία φόρμα, όπου ο εκάστοτε Ιδιοκτήτης συμπληρώνει το Όνομα(“Name”), το Email και το Κωδικό Χρήστη (“Password”) και διαλέγει μέσω ενός drop-down menu το ξενοδοχείο στο οποίο ανήκει ο υπάλληλος. Στη συνέχεια, κάνοντας κλικ στο κουμπί «Αποθήκευση» αποθηκεύεται ο υπάλληλος. Στο Σχήμα 4.9, απεικονίζεται ο κώδικας για τη δημιουργία της εν λόγω σελίδας.



Σχήμα 19: Σελίδα Δημιουργία Χρήστη

```
import { useSelector } from 'react-redux'
import Spinner from '../../components/Spinner'
import CreateUser from '../../components/users/CreateUser'

function AddNewUser() {
   const { user, isLoading } = useSelector((state) => state.auth)

   if (isLoading) return <Spinner />

   if (user.role === 'employee') return <Navigate to='/not-found' />

   return (
      <div className='main-page add-new-user'>
         <CreateUser />
      </div>
   )
}

export default AddNewUser
```

Σχήμα 20: Σύνταξη συνάρτησης “AddNewUser()”

Με την χρήση του Hook useSelector() παίρνουμε τα στοιχεία του χρήστη που βρίσκονται αποθηκευμένα

στο Redux και στην συνέχεια ελέγχουμε τον ρόλο του συγκεκριμένου χρήστη. Εάν ο χρήστης έχει τον ρόλο του υπαλλήλου ανακατευθύνεται στην σελίδα “Not Found”. Αντιθέτως, εάν ο ρόλος του χρήση δεν είναι αυτός του υπαλλήλου, γίνεται render το Component “CreateUser” που είναι υπεύθυνο για την δημιουργία νέου χρήστη.

```
// Submit the user data
const submitUserData = (e) => {
    e.preventDefault()

    const { name, email, password, confirm_password } = formData

    if (!name || name.length < 5 || name.length > 30) {
        return toast.error('Παρακαλώ συμπληρώστε έγκυρο όνομα χρήστη')
    }

    if (!email) {
        return toast.error('Παρακαλώ συμπληρώστε email')
    }

    if (password !== confirm_password || password.length < 5) {
        return toast.error('Παρακαλώ παρέχετε σωστούς κωδικούς χρήστη')
    }

    if (!chooseHotel || chooseHotel === 'undefined') {
        return toast.error('Παρακαλώ συμπληρώστε Ξενοδοχείο')
    }

    const userData = {
        name,
        email,
        password,
        hotelId: chooseHotel,
    }

    dispatch(createEmployee(userData))
        .unwrap()
        .then((res) => {
            toast.success('Ο χρήστης δημιουργήθηκε επιτυχώς')
            navigate(`/my-users`)
        })
        .catch((error) => toast.error(error))
}
```

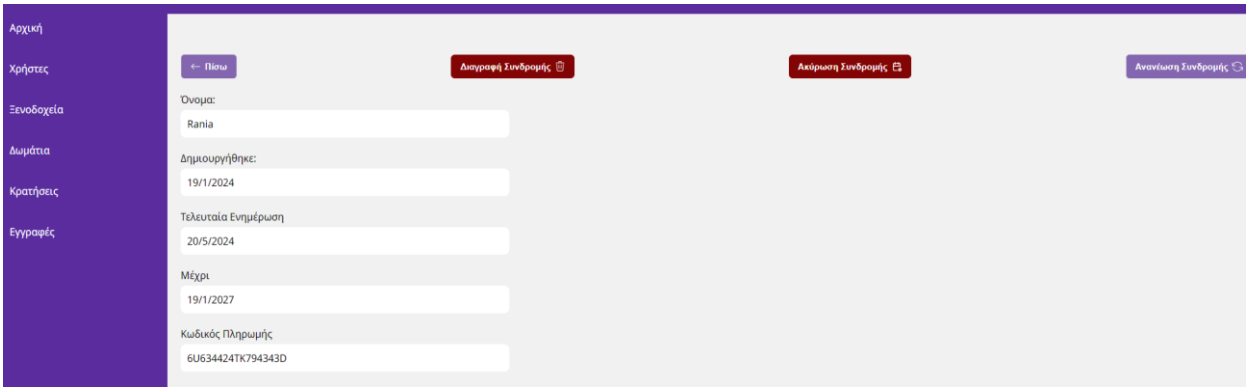
Σχήμα 21: Σύνταξη συνάρτησης “submitUserData()”

Το συγκεκριμένο Component αποτελείται από μια φόρμα εισαγωγής των στοιχείων έχουν περιγραφεί άνωθεν και της function “submitUserData()”. Στην οποία, αρχικά γίνεται έλεγχος της ορθότητας των στοιχείων εισαγωγής και στην συνέχεια με την βοήθεια του Hook useDispatch() δημιουργείται ένα αίτημα POST στην διεύθυνση “api/users/registerEmployee”. Εάν το αίτημα είναι επιτυχές, τότε εμφανίζεται σχετικό μήνυμα και ο χρήστης ανακατευθύνεται στην σελίδα με την λίστα υπαλλήλων, εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους.

.

4.7 Σελίδα «Διαχείριση Εγγραφής»

Όπως φαίνεται στο Σχήμα, η σελίδα "Διαχείριση Εγγραφής", η οποία είναι διαθέσιμη μόνο για τον "Admin" δηλαδή διαχειριστή της εφαρμογής, αποτελείται από τα στοιχεία μιας ενεργής συνδρομής ενός ξενοδοχείου, το όνομα, την ημερομηνία δημιουργίας και λήξης, την τελευταία ενημέρωση και τον κωδικό πληρωμής, και από τις επιλογές "Διαγραφή Συνδρομής", ώστε να διαγράψει την συνδρομή του εκάστοτε χρήστη, την επιλογη "Ακύρωση Συνδρομης", για να ακυρώσει προσωρινά την συνδρομή και την "Ανανέωση Συνδρομής", μέσω της οποίας η συνδρομή του εκάστοτε χρήστη ανανεώνεται για ένα ακόμη έτος.



Σχήμα 22: Σελίδα Διαχείρισης Εγγραφής

```
function SingleSubscription() {
    const { id } = useParams()
    const { user } = useSelector((state) => state.auth)
    const { subscription, isLoading } = useSelector((state) => state.subscriptions)

    const dispatch = useDispatch()

    useEffect(() => {
        dispatch(getSubscription(id))
            .unwrap()
            .then((res) => {})
            .catch((error) => toast.error(error))
    }, [])

    if (isLoading) return <Spinner />

    if (user.role !== 'admin') return <Navigate to='/not-found' />

    return (
        <div className='single-hotel main-page'>
            <UpdateSubscription subscription={subscription} />
        </div>
    )
}

export default SingleSubscription
```

Σχήμα 23: Σύνταξη συνάρτησης "SingleSubscription()"

Με την χρήση του του Hook useSelector() παίρνουμε τα στοιχεία των συνδρομών που βρίσκονται αποθηκευμένα στο Redux, στην συνέχεια ελέγχουμε τον ρόλο του συγκεκριμένου χρήστη. Εάν ο χρήστης δεν έχει τον ρόλο του Admin ανακατευθύνεται στην σελίδα “Not Found”. Αντιθέτως, εάν ο ρόλος του χρήση είναι αυτός του Admin, γίνεται render το Component “UpdateSubscription” που είναι υπεύθυνο για την εμφάνιση της συνδρομής με το συγκεκριμένο ‘id”.

```
function UpdateSubscription({ subscription, isLoading }) {
  const navigate = useNavigate()
  const dispatch = useDispatch()

  const renewThisSubscription = () => {
    dispatch(updateSubscription(subscription._id))
      .unwrap()
      .then((res) => {
        toast.success('Η συνδρομή ανανεώθηκε για ένα χρόνο')
        window.location.reload(false)
      })
      .catch((error) => toast.error(error))
  }

  const cancelThisSubscription = () => {
    dispatch(cancelSubscription(subscription._id))
      .unwrap()
      .then((res) => {
        toast.success('Η συνδρομή ακυρώθηκε')
        window.location.reload(false)
      })
      .catch((error) => toast.error(error))
  }

  const deleteCurrentSubscription = () => {
    //e.preventDefault()
    dispatch(deleteSubscription(subscription._id))
      .unwrap()
      .then((res) => {
        toast.success('Η συνδρομή διαγράφηκε')
        navigate('/all-subscriptions')
      })
      .catch((error) => toast.error(error))
  }
  return (
    <>
      <div className='page-buttons'>
        <BackButton />
        <button className='delete' onClick={deleteCurrentSubscription} disabled={isLoading}>
          Διαγραφή Συνδρομής <GoTrash size={16} />
        </button>
        <button className='delete' onClick={cancelThisSubscription} disabled={isLoading}>
          Ακύρωση Συνδρομής <TbCalendarCancel size={16} />
        </button>
        <button className='main-button' disabled={isLoading} onClick={renewThisSubscription}>
          Ανανέωση Συνδρομής
          <TfiReload size={16} />
        </button>
```

Σχήμα 24: Σύνταξη συνάρτησης “UpdateSubscription()”

Το συγκεκριμένο Component αποτελείται από μια φόρμα προβολής των στοιχείων και των κουμπιών/επιλογών που έχουν περιγραφεί άνωθεν και της function “UpdateSubscription()”. Στην οποία, ανάλογα με την επιλογή του χρήστη δημιουργείται το αντίστοιχο αίτημα, “Διαγραφή Συνδρομής DELETE, “Ακύρωση Συνδρομής”/”Ανανέωση Συνδρομής” PATCH, στις διευθύνσεις

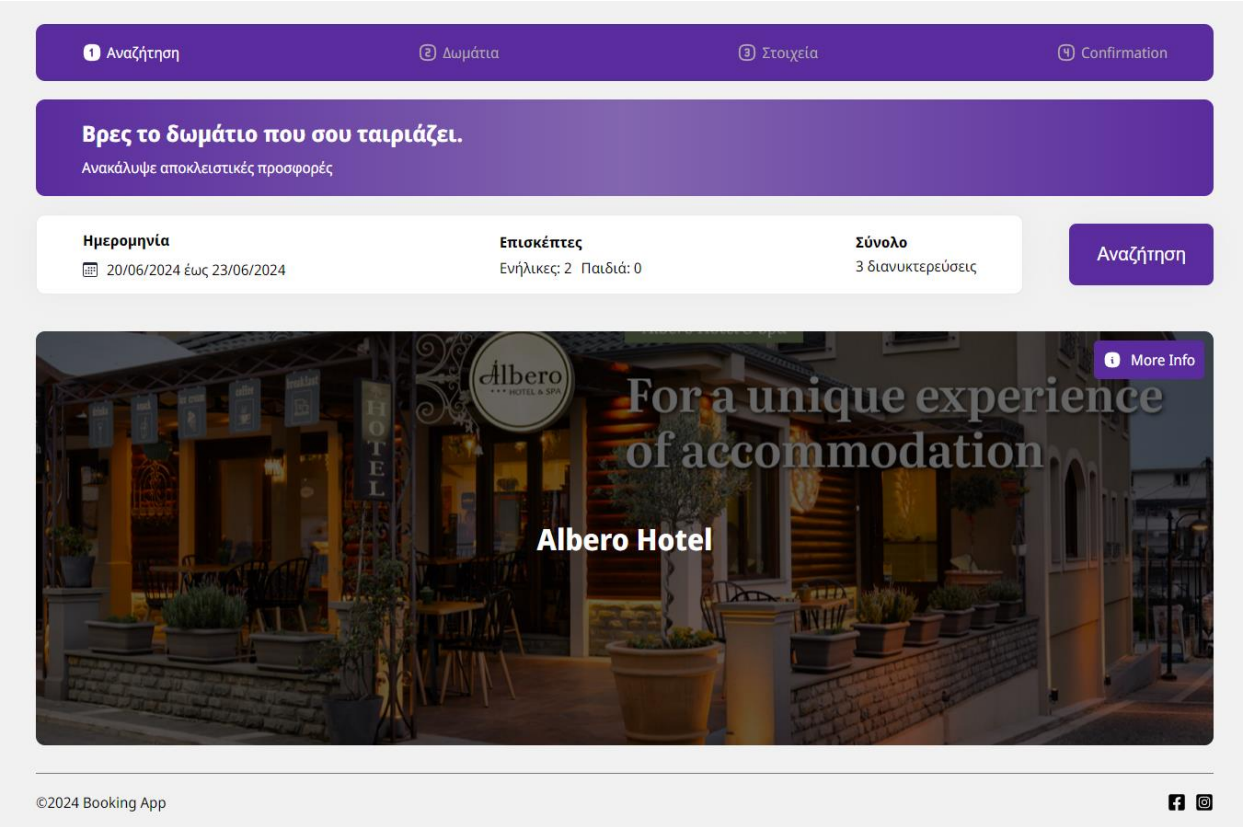
“api/subscriptions/[id]/deleteSubscription” και “api/subscriptions/[id]/renewSubscription” και “api/subscriptions/[id]/cancelSubscription” αντίστοιχα. Εάν τα αίτηματα είναι επιτυχή, τότε εμφανίζεται σχετικό μήνυμα, εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους.

4.8 «Δημιουργία Κράτησης»

Η δημιουργία κράτησης, η οποία είναι διαθέσιμη μόνο για τον απλό χρήστη/πελάτη, αποτελείται από τέσσερα στάδια. Την αναζήτηση δωματίου στο συγκεκριμένο ξενοδοχείο, τη λίστα των διαθέσιμων δωματίων, τα στοιχεία της κράτησης και την επιβεβαίωσή της.

4.8.1 Βήμα 1ο - Αναζήτηση

Όπως φαίνεται στο Σχήμα, η σελίδα της αναζήτησης αποτελείται από το πεδίο εισαγωγής ημερομηνίας το οποίο επιτυγχάνεται μέσω ενός pop-up date range, την εισαγωγή αριθμού επισκεπτών (ενηλίκων και παιδιών) και το σύνολο των διανυκτερεύσεων το οποίο υπολογίζεται δυναμικά με βάση τις ημερομηνίες που έχει επιλέξει ο χρήστης. Επιπροσθέτως, υπάρχει και ένα preview του ξενοδοχείου, στο οποίο πατώντας το κουμπί “More Info”, εμφανίζεται ένα pop-up dialog με τις βασικές πληροφορίες του καταλύματος.



Σχήμα 25: Σελίδα δημιουργίας κράτησης-Βήμα 1ο

```
function StepOne() {
   const { id } = useParams()
   const dispatch = useDispatch()
   const { hotel, isLoading } = useSelector((state) => state.hotels)

   useEffect(() => {
      dispatch(getSingleHotel(id))
         .unwrap()
         .then((res) => {})
         .catch((error) => toast.error(error))
   }, [])

   return (
      <div className='container step-one'>
         <div>
            <CheckOutSteps step1 />
            <div className='step-one-title'>
               <h2>Βρες το δωμάτιο που σου ταιριάζει.</h2>
               <p>Ανακάλυψε αποκλειστικές προσφορές</p>
            </div>
            <DateAndPeople id={id} />
            <Hotel hotel={hotel} />
         </div>
         <Footer />
      </div>
   )
}

export default StepOne
```

Σχήμα 26: Σύνταξη συνάρτησης “getSingleHotel()”

Με την χρήση του Hook useSelector() παίρνουμε τα στοιχεία του ξενοδοχείου που βρίσκονται αποθηκευμένα στο Redux.

Το βήμα αυτό αποτελείται από δύο Components, το Hotel, και το DateAndPeople.

```jsx
function Hotel({ hotel }) {
  const [showInfo, setShowInfo] = useState(false)
  return (
    <div className='book-display-hotel'>
      <div className='image' style={{ backgroundImage: `url(${hotel.imgs})` }}>
        <h1>{hotel.name}</h1>
        <div className='more-info' onClick={() => setShowInfo(true)}>
          <TbInfoSquareRoundedFilled size={20} />
          More Info
        </div>
      </div>
      {showInfo && (
        <div className='hotel-details'>
          <div className='hotel-container'>
            <div className='top'>
              <div className='hotel-img'>
                <img src={hotel.imgs} alt='hotel image' />
              </div>
              <div className='desc'>{hotel.description}</div>
            </div>
            <div className='info'>
              <div>
                <MdOutlineAlternateEmail size={20} /> {hotel.email}
              </div>
              <div>
                <GiRotaryPhone size={20} /> {hotel.phone}
              </div>
              <div>
                <FaMapPin size={20} /> {hotel.address}
              </div>
            </div>
          </div>
          <div className='close-details' onClick={() => setShowInfo(false)}>
            <CgCloseR color='#ff0000' size={25} />
          </div>
        </div>
      )}
    </div>
  )
}
```

Σχήμα 27: Εμφάνιση φωτογραφίας και στοιχείων ξενοδοχείου

Το συγκεκριμένο Component αποτελείται από την φωτογραφία του εκάστοτε ξενοδοχείου καθώς επίσης και τις απαραίτητες πληροφορίες που εμφανίζονται στο κουμπί “More Info” (σύντομη περιγραφή, email, τηλέφωνο και διεύθυνση).

```
function DateAndPeople({ id }) {
  const navigate = useNavigate();
  const now = new Date();
  const [openDate, setOpenDate] = useState(false);
  const [openPeople, setOpenPeople] = useState(false);
  const [people, setPeople] = useState(2);
  const [children, setChildren] = useState(0);
  const [date, setDate] = useState([
    {
      startDate: new Date(now.getFullYear(), now.getMonth(), now.getDate()),
      endDate: new Date(now.getFullYear(), now.getMonth() + 1, 0),
      key: "selection",
    },
  ]);
  const { startDate, endDate } = date[0];

  // calculate nights
  const date1 = new Date(startDate);
  const date2 = new Date(endDate);
  const Difference_In_Time = date2.getTime() - date1.getTime();
  const Difference_In_Days = Difference_In_Time / (1000 * 3600 * 24);

  const saveData = () => {
    navigate(
      `/book/rooms/${id}?startDate=${format(
        date[0].startDate,
        "yyyy-MM-dd"
      )}&endDate=${format(
        date[0].endDate,
        "yyyy-MM-dd"
      )}&people=${people}&children=${children}`
    );
  };
  return (
    <div className="search-bar-comp">
      <div className="search-bar-abs">
        <div className="search-bar">
          <div className="search-date">
            <h2>Ημερομηνία</h2>
            <div className="pick-date">
              <BsCalendar3 fill="#21293a" width="18px" height="18px" />
              <span
                onClick={() => {
                  setOpenDate(!openDate);
```

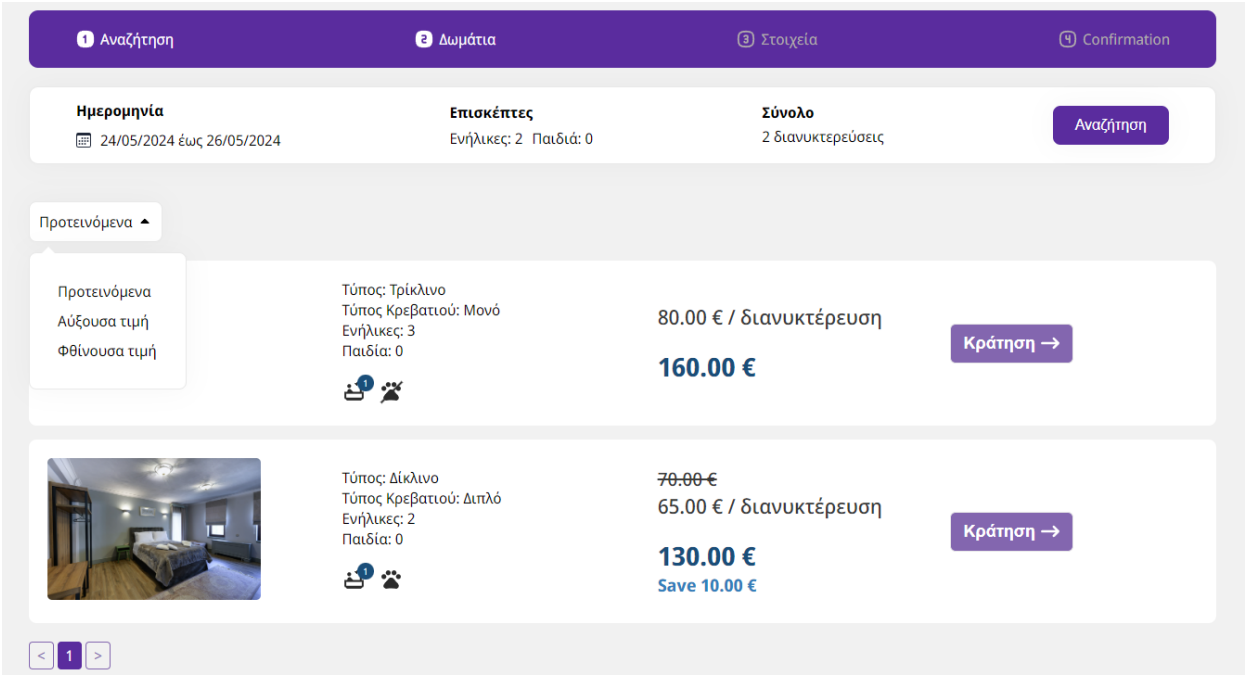
```
            <div className="total">
              <h2>Σύνολο</h2>
              {Difference_In_Days === 1 ? (
                <div>{Difference_In_Days} διανυκτέρευση'</div>
              ) : (
                <div>{Difference_In_Days} διανυκτερεύσεις</div>
              )}
            </div>
          </div>
          <button className="search" onClick={saveData}>
            Αναζήτηση
          </button>
        </div>
      </div>
    </div>
  );
}
```

Σχήμα 28: Σύνταξη συνάρτησης “saveData()”

Το συγκεκριμένο Component αποτελείται από τα πεδία εισαγωγής των στοιχείων που έχουν περιγραφεί άνωθεν και της function “saveData()”. Η οποία καλείται όταν ο χρήστης πατήσει το κουμπί της αναζήτησης τότε ανακατευθύνεται στο “api/book/rooms/id=” έχοντας σαν query το id του ξενοδοχείου, τις ημερομηνίες και τους επισκέπτες που έχει εισάγει, το οποίο αποτελεί και το δεύτερο βήμα.

4.8.2 Βήμα 2ο - Δωμάτια

Όπως φαίνεται στο Σχήμα , η σελίδα των δωματίων αποτελείται από μία λίστα των διαθέσιμων δωματίων ανάλογα με τα στοιχεία εισαγωγής του προηγούμενου βήματος. Υπάρχει επιλογή ταξινόμησης κατά αύξουσα και φθίνουσα τιμή, καθώς επίσης και προτεινόμενων δωματίων. Για κάθε διαθέσιμο δωμάτιο, αναγράφονται οι βασικές πληροφορίες του δωματίου αλλά και η τιμή τους τόσο ανά διανυκτέρευση όσο και στο σύνολό της. Εάν δεν υπάρχουν διαθέσιμα δωμάτια με βάση τις επιλογές του χρήστη, εμφανίζεται το μήνυμα “Δεν βρέθηκαν δωμάτια” και είναι διαθέσιμα τα πεδία του πρώτου βήματος, ώστε να γίνει εκ νέου αναζήτηση.



Σχήμα 29: Σελίδα εμφάνισης διαθέσιμων δωματίων - Βήμα 2ο

```
function StepTwo() {
    const { id } = useParams()
    const queryParams = new URLSearchParams(window.location.search)
    const startDate = queryParams.get('startDate')
    const endDate = queryParams.get('endDate')
    const people = queryParams.get('people')
    const children = queryParams.get('children')
    const navigate = useNavigate()
    const dispatch = useDispatch()
    const { availableRooms, isLoading } = useSelector((state) => state.rooms)

    const date1 = new Date(startDate)
    const date2 = new Date(endDate)
    const Difference_In_Time = date2.getTime() - date1.getTime()
    const Difference_In_Days = Difference_In_Time / (1000 * 3600 * 24)

    useEffect(() => {
        if (!id) {
            navigate('/book/not-found')
        }

        if (!startDate) {
            navigate(`/book/${id}`)
        }
        if (!endDate) {
            navigate(`/book/${id}`)
        }
        if (!people) {
            navigate(`/book/${id}`)
        }
        if (!children) {
            navigate(`/book/${id}`)
        }
    }, [id, startDate, endDate, people, children])

    useEffect(() => {
        const queries = { id, startDate, endDate, adults: people, children }
        dispatch(getAvailableRooms(queries))
            .unwrap()
            .then((res) => {})
            .catch((error) => toast.error(error))
    }, [id, startDate, endDate, people, children])

    return (
        <div className='container step-two'>
            <div>
```

Σχήμα 30: Σύνταξη συνάρτησης “getAvailableRooms()”

Με τη χρήση του Hook useSelector() παίρνουμε τα στοιχεία των δωματίων που βρίσκονται αποθηκευμένα στο Redux και ελέγχουμε την ορθότητά τους. Αν είναι ορθά, γίνονται render τα Components CheckOutSteps, step1, step2, το component StepTwoFilters και το RoomList.

```
function StepTwoFilters({ id }) {
  let [searchParams, setSearchParams] = useSearchParams();
  const now = new Date();
  const [openDate, setOpenDate] = useState(false);
  const [openPeople, setOpenPeople] = useState(false);
  const [people, setPeople] = useState(parseInt(searchParams?.get("people")));
  const [children, setChildren] = useState(
    parseInt(searchParams?.get("children"))
  );
  const [date, setDate] = useState([
    {
      startDate: new Date(searchParams?.get("startDate")),
      endDate: new Date(searchParams?.get("endDate")),
      key: "selection",
    },
  ]);
  const { startDate, endDate } = date[0];

  // calculate nights
  const date1 = new Date(startDate);
  const date2 = new Date(endDate);
  const Difference_In_Time = date2.getTime() - date1.getTime();
  const Difference_In_Days = Difference_In_Time / (1000 * 3600 * 24);

  const saveData = () => {
    const searchParams = {
      startDate: format(date[0].startDate, "yyyy-MM-dd"),
      endDate: format(date[0].endDate, "yyyy-MM-dd"),
      people,
      children,
    };
    setSearchParams(searchParams);
  };
  return (
    <div className="search-bar-comp">
      <div className="search-bar-abs">
        <div className="search-bar">
          <div className="search-date">
            <h2>Ημερομηνία</h2>
            <div className="pick-date">
              <BsCalendar3 fill="#21293a" width="18px" height="18px" />
              <span
                onClick={() => {
                  setOpenDate(!openDate);
                }}
                className="date-range"
              >
```

Σχήμα 31: Σύνταξη συνάρτησης “saveData()”

Το συγκεκριμένο Component, είναι αντίστοιχο του DateAndPeople και αποτελείται εξίσου από τα πεδία εισαγωγής των στοιχείων που έχουν περιγραφεί άνωθεν και της function “saveData()”.

```
function RoomList({ rooms, isLoading, days }) {
   const [openSort, setOpenSort] = useState(false)
   const [sortOption, setSortOption] = useState('Προτεινόμενα')
   const sortOptions = ['Προτεινόμενα', 'Αύξουσα τιμή', 'Φθίνουσα τιμή']

   const dispatch = useDispatch()
   const navigate = useNavigate()

   const sortedRooms = [...rooms]?.sort(
      sortOption === 'Προτεινόμενα'
         ? (a, b) => Number(b?.isFeatured) - Number(a?.isFeatured)
         : sortOption === 'Αύξουσα τιμή'
         ? (a, b) => {
              if (!a.discountPrice && !b.discountPrice) {
                 return a?.regularPrice - b?.regularPrice
              } else if (!a?.discountPrice && b?.discountPrice) {
                 return a?.regularPrice - b?.discountPrice
              } else if (a?.discountPrice && !b?.discountPrice) {
                 return a?.discountPrice - b?.regularPrice
              } else {
                 return a?.discountPrice - b?.discountPrice
              }
           }
         : (a, b) => {
              if (!a?.discountPrice && !b?.discountPrice) {
                 return b?.regularPrice - a?.regularPrice
              } else if (!a?.discountPrice && b?.discountPrice) {
                 return b?.regularPrice - a?.discountPrice
              } else if (a?.discountPrice && !b?.discountPrice) {
                 return b?.discountPrice - a?.regularPrice
              } else {
                 return b?.discountPrice - a?.discountPrice
              }
           }
   )

   // Pagination
   const [itemOffset, setItemOffset] = useState(0)
   const itemsPerPage = 2
   const endOffset = itemOffset + itemsPerPage
   const currentItems = sortedRooms?.slice(itemOffset, endOffset)
   const pageCount = Math.ceil(rooms?.length / itemsPerPage)

   // Invoke when user click to request another page.
   const handlePageClick = (event) => {
      const newOffset = (event.selected * itemsPerPage) % rooms.length
      setItemOffset(newOffset)
   }
```

Σχήμα 32: Διαχείριση sorting και pagination

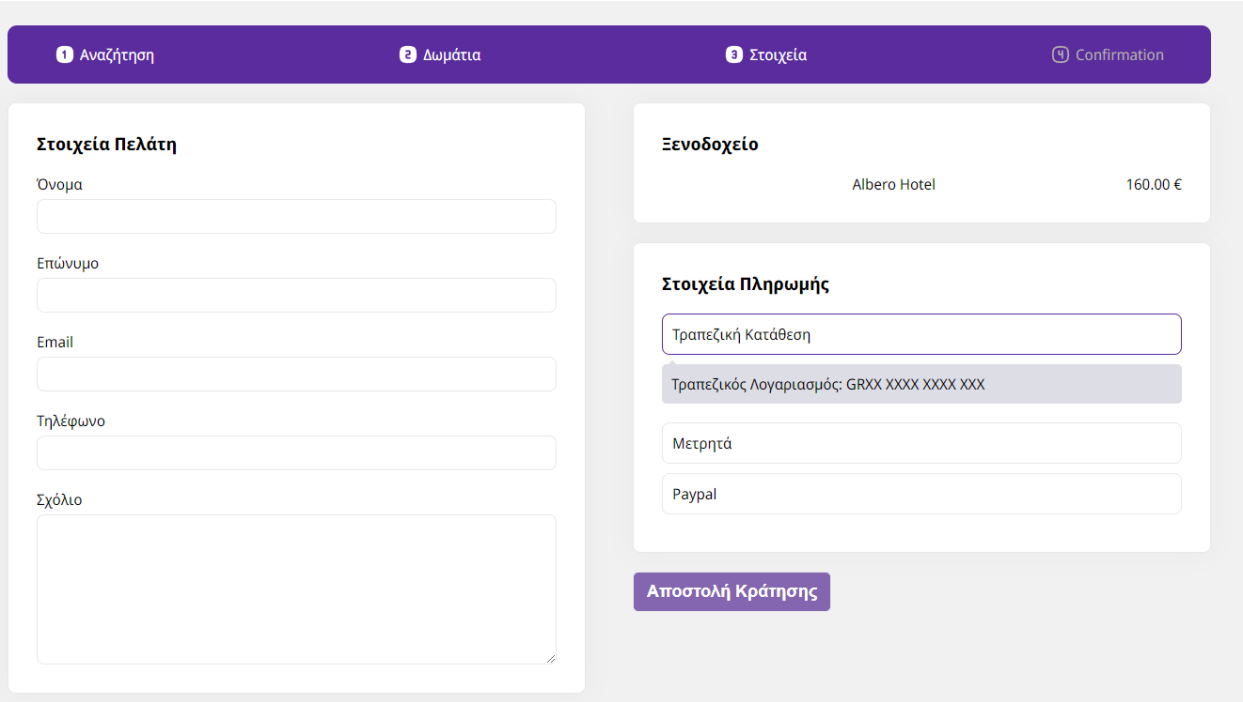
```
const saveRoomInfo = (room) => {
    const total = days * (room.discountPrice ? room.discountPrice : room.regularPrice)

    const queryParams = new URLSearchParams(window.location.search)
    const startDate = queryParams.get('startDate')
    const endDate = queryParams.get('endDate')
    const people = queryParams.get('people')
    const children = queryParams.get('children')

    const data = { startDate, endDate, people, children, room, total }

    dispatch(saveReservation(data))
        .unwrap()
        .then((res) => {
            navigate(`/book/complete-checkout?id=${room.hotel.hotel_id}`)
        })
        .catch((error) => toast.error(error))
}
```

Σχήμα 33: Σύνταξη συνάρτησης “saveReservation()”

Το συγκεκριμένο Component αποτελείται από την λίστα των διαθέσιμων δωματίων, ενα drop-down menu ως φίλτρο που περιγράφεται άνωθεν, του Pagination και της function saveRoomInfo(). Η οποία καλείται όταν ο χρήστης πατήσει το κουμπί “Κράτηση” για το εκάστοτε δωμάτιο, τότε ανακατευθύνεται στο “api/book/complete-checkout?id=” έχοντας σαν query το id του ξενοδοχείου που βρίσκεται το δωμάτιο επιλογής του χρήστη.

4.8.3 Βήμα 3ο - Στοιχεία Κράτησης

Όπως φαίνεται στο Σχήμα , το τρίτο βήμα αποτελείται από μια φόρμα εισαγωγής των στοιχείων του πελάτη, το όνομα του ξενοδοχείου με το συνολικό ποσό πληρωμής, και τους διαθέσιμους τρόπους πληρωμής. Οι διαθέσιμοι τρόποι πληρωμής είναι τρεις, η Τραπεζική κατάθεση, όπου εμφανίζεται ο τραπεζικός λογαριασμός που μπορεί να κάνει κατάθεση ο χρήστης, τα Μετρητα, όπου εμφανίζεται το μήνυμα “Πληρωμή με μετρητά κατά την άφιξη στο ξενοδοχείο” και η Paypal, όπου εμφανίζεται ένα νέο παράθυρο, ώστε ο χρήστης να συνδεθεί στο λογαριασμό του στην Paypal και να ολοκληρώσει την πληρωμή.



Σχήμα 34: Σελίδα υποβολής στοιχείων κράτησης

```
function StepThree() {
   const { reservationData } = useSelector((state) => state.reservations)

   const queryParams = new URLSearchParams(window.location.search)
   const id = queryParams.get('id')

   // if room data is empty and there is no hotel id then go to not-found page
   if (Object.entries(reservationData).length === 0 && !id) return <Navigate to='/book/not-found' />

   // if room data is empty and but there is hotel id go to book page for this hotel
   if (Object.entries(reservationData).length === 0) return <Navigate to={`/book/hotel/${id}`} />

   return (
      <div className='container step-three'>
         <div>
            <CheckOutSteps step1 step2 step3 />
            <div>
               <OrderDetails reservationData={reservationData} />
            </div>
         </div>
         <Footer />
      </div>
   )
}

export default StepThree
```

Σχήμα 35: Εμφάνιση του component “OrderDetails”

Με την χρήση του Hook useSelector() παίρνουμε τα στοιχεία της κράτησης που είναι αποθηκευμένα στο Redux. Έπειτα ελέγχουμε εάν το id του ξενοδοχείου ή του δωματίου είναι κενά και τότε ο χρήστης ανακατευθύνεται αντίστοιχα. Εφόσον δεν είναι κενά γίνεται render το Component “OrderDetails” που είναι υπεύθυνο για την καταχώρηση της κράτησης.

```
function OrderDetails({ reservationData }) {
  const { isLoading } = useSelector((state) => state.reservations);
  const [payment, setPayment] = useState("Τραπεζική Κατάθεση");
  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    phone: "",
    comment: "",
  });

  const dispatch = useDispatch();
  const navigate = useNavigate();

  const reservation = {
    description: `Reservation at Hotel: ${reservationData?.room?.hotel?.hotel_name}`,
    price: reservationData?.total,
  };
  const paymentMethods = ["Τραπεζική Κατάθεση", "Μετρητά", "Paypal"];

  const handleReservationData = (e) => {
    setFormData((prevState) => ({
      ...prevState,
      [e.target.name]: e.target.value,
    }));
  };

  const displayButton = () => {
    if (payment === "Paypal")
      return (
        <div className="paypal-button-container">
          <PaypalCheckoutPayment
            reservation={reservation}
            formData={formData}
            reservationData={reservationData}
          />
        </div>
      );
    else
      return (
        <button
          className="main-button"
          type="submit"
          onClick={submitReservation}
          disabled={isLoading}
        >
          {isLoading ? "Γίνεται Αποστολή..." : "Αποστολή Κράτησης"}
        </button>
      );
```

Σχήμα 36: “OrderDetails” component

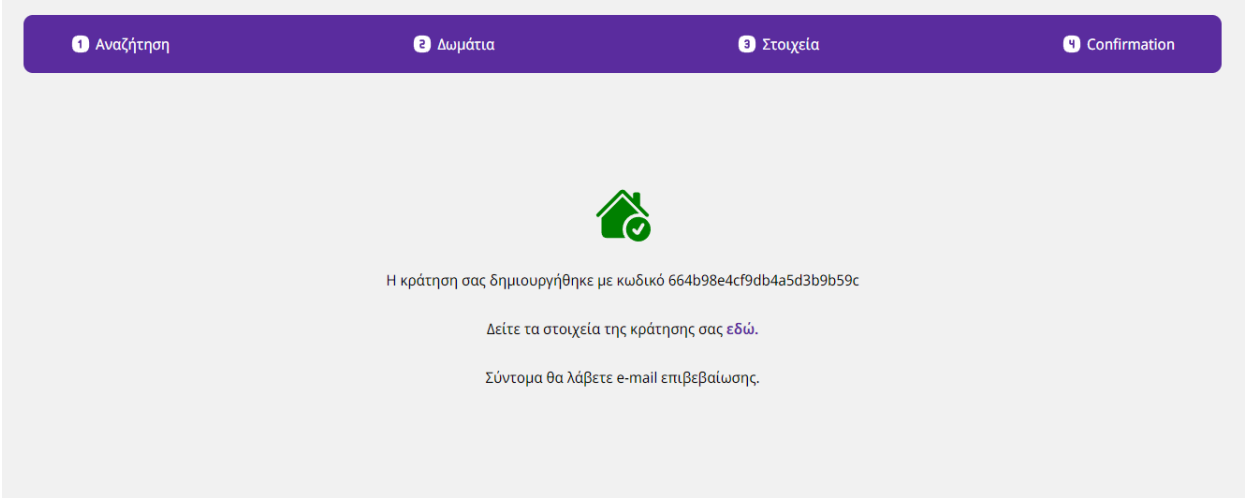
```
const submitReservation = (e) => {
  e.preventDefault();
  const { firstName, lastName, email, phone, comment } = formData;
  dispatch(
    createReservation({
      firstName,
      lastName,
      email,
      phone,
      comment,
      people: {
        adults: reservationData?.people,
        children: reservationData?.children,
      },
      payment: {
        method: payment,
        isPaid: false,
        transactionId: "",
        total: Number(reservationData?.total),
      },
      hotel: {
        hotel_obj: reservationData?.room?.hotel?.hotel_id,
        hotel_id: reservationData?.room?.hotel?.hotel_id,
        hotel_name: reservationData?.room?.hotel?.hotel_name,
        hotel_owner: reservationData?.room?.hotel?.hotel_owner,
      },
      room: {
        room_obj: reservationData?.room,
        room_id: reservationData?.room?._id,
        room_num: reservationData?.room?.roomNumber,
      },
      dates: {
        startDate: new Date(reservationData?.startDate),
        endDate: new Date(reservationData?.endDate),
      },
    })
  )
    .unwrap()
    .then((res) => {
      console.log(res);
      navigate(`/book/hotel/reservation-confirmation?id=${res}`);
    })
    .catch((error) => toast.error(error));
};
```

Σχήμα 37: Σύνταξη συνάρτησης “submitReservation()”

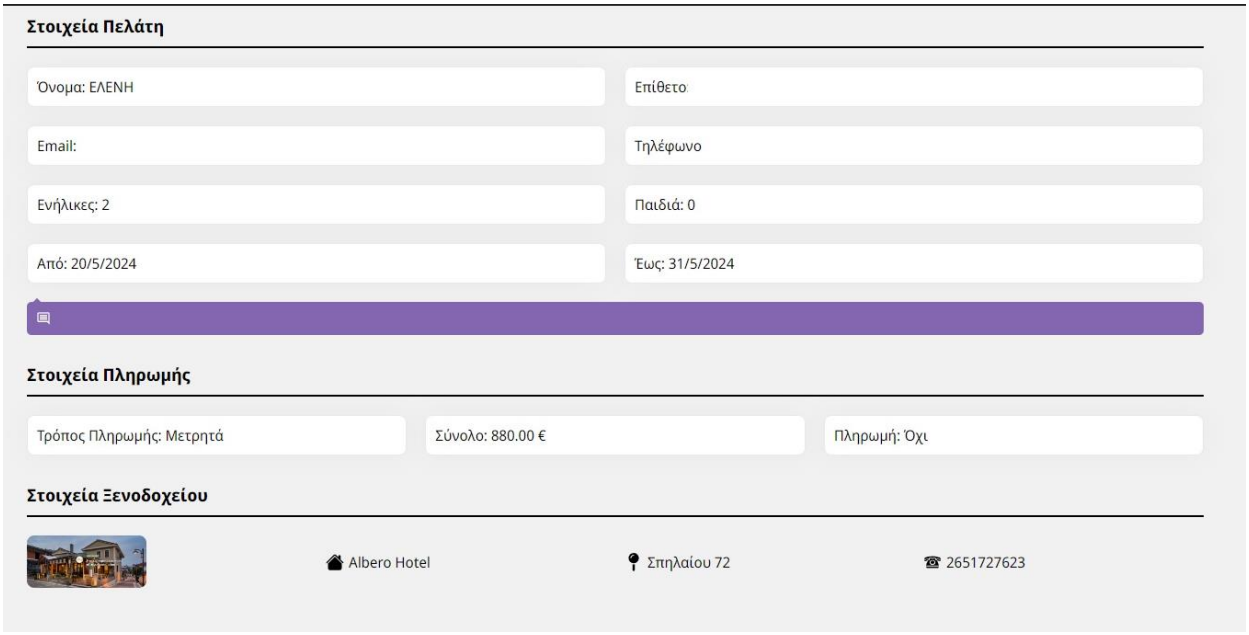
Το συγκεκριμένο Component αποτελείται από μια φόρμα εισαγωγής των στοιχείων που έχουν περιγραφεί άνωθεν, της function displayButton() μέσω της οποίας γίνεται render το Component “PaypalCheckoutPayment” που είναι υπέυθυνο για την εμφάνιση του Paypal dialog και της function “submitReservation()”. Με την βοήθεια του Hook useDispatch() δημιουργείται ένα αίτημα POST στην διεύθυνση “api/reservation/createReservation”. Εάν το αίτημα είναι επιτυχές, τότε εμφανίζεται σχετικό μήνυμα και ο χρήστης ανακατευθύνεται στην σελίδα με τα στοιχεία της κράτησης, εάν υπάρχει σφάλμα τότε εμφανίζεται σχετικό μήνυμα λάθους.

4.8.4 Βήμα 4ο - Επιβεβαίωση Κράτησης

Όπως φαίνεται στο Σχήμα, στην τέταρτη και τελευταία σελίδα της διαδικασίας, εμφανίζεται η επιβεβαίωση της κράτησης με τον μοναδικό κωδικό της, καθώς επίσης και ένας υπερσύνδεσμος ο οποίος μεταφέρει τον χρήστη σε μια νέα σελίδα που παρέχονται όλα τα στοιχεία της κράτησης και του ξενοδοχείου διαμονής του (Σχημα). Τέλος, ενημερώνει τον χρήστη για την λήψη e-mail επιβεβαίωσης.



Σχήμα 38: Σελίδα επιβεβαίωσης κράτησης



Σχήμα 39: Σελίδα εμφάνισης στοιχείων κράτησης

```
function StepFour() {
   const queryParams = new URLSearchParams(window.location.search)
   const id = queryParams.get('id')

   return (
      <div className='container step-four'>
         <div>
            <CheckOutSteps step1 step2 step3 step4 />
            <div className='display-reservation-sucess'>
               <BsFillHouseCheckFill size={60} color='green' className='mt-20' />
               <div>Η κράτηση σας δημιουργήθηκε με κωδικό {id}</div>
               <div>
                  Δείτε τα στοιχεία της κράτησης σας
                  <Link to={`/book/hotel/reservation-details?id=${id}`}> εδώ.</Link>
               </div>
               <div>Σύντομα θα λάβετε e-mail επιβεβαίωσης.</div>
            </div>
         </div>
         <Footer />
      </div>
   )
}

export default StepFour
```

Σχήμα 40:Εμφάνιση στοιχείων κράτησης

Με την χρήση URLSearchParams() παίρνουμε το id της κράτησης και γίνονται render τα στοιχεία που αναφέρονται παραπάνω.

4.8.5 PayPal Integration

Στο σημείο αυτό, αξίζει να σημειωθεί πως η εφαρμογή μας υποστηρίζει πληρωμή μέσω PayPal όπως αναφέρθηκε και παραπάνω.

Η PayPal είναι μια διαδικτυακή πλατφόρμα πληρωμών που δίνει τη δυνατότητα στους χρήστες να στέλνουν και να λαμβάνουν χρήματα παγκοσμίως. Υπάρχει η δυνατότητα με απλά βήματα να συνδέσουν τον τραπεζικό λογαριασμό, την πιστωτική ή τη χρεωστική τους κάρτα αλλά και να δημιουργήσουν έναν λογαριασμό Υπολοίπου PayPal.

Εργάζεται για την προστασία των οικονομικών δεδομένων και των πληρωμών των χρηστών του σε κάθε στάδιο. Κάθε συναλλαγή κρυπτογραφείται ώστε να διασφαλίζεται η ιδιωτικότητα των προσωπικών πληροφοριών. Η εταιρεία δεν κοινοποιεί τα οικονομικά στοιχεία των χρηστών στους πωλητές και παρέχει ασφαλείς επιλογές σύνδεσης, ενώ η παρακολούθηση για εντοπισμό απάτης 24 ώρες το 24ωρο προστατεύει τους λογαριασμούς από μη εξουσιοδοτημένη πρόσβαση. Με την έγκαιρη ανίχνευση απάτης και ειδοποιήσεις σε πραγματικό χρόνο μέσω της εφαρμογής, το PayPal προλαμβάνει πιθανές απειλές. Επίσης, η υπηρεσία Προστασίας Αγορών του PayPal καλύπτει τις επιλέξιμες αγορές, διασφαλίζοντας ότι οι χρήστες θα αποζημιωθούν για την αξία της αγοράς και τα αρχικά έξοδα αποστολής, εάν οι παραγγελίες δεν παραληφθούν ή δεν ανταποκρίνονται στην περιγραφή.

Για τους παραπάνω λόγους και λαμβάνοντας υπόψην το πόσο οι χρήστες εμπιστεύονται τις πληρωμές μέσω PayPal, θεωρήσαμε πως θα είναι σημαντικό να προσφέρουμε αυτή τη μέθοδο πληρωμής,

Προχωρήσαμε λοιπόν στην ενσωμάτωση αυτής της μεθόδου, η οποία επιτεύχθηκε με τη ρύθμιση ενός περιβάλλοντος sandbox PayPal. Μέσω της ιστοσελίδας, δημιουργώντας τον απαραίτητο λογαριασμό,

λάβαμε ένα Client-ID. Στη συνέχεια, εγκαταστήσαμε το PayPal package, το οποίο παρέχει τα απαραίτητα εργαλεία για την προσθήκη του PayPal Checkout στην εφαρμογή μας και περιλαμβάνει δύο components, το PayPalScriptProvider και το PayPalButtons. Δημιουργήσαμε ένα αντικείμενο initialOptions για να καθορίσουμε τις απαραίτητες ρυθμίσεις για το SDK της PayPal, στο οποίο καταχωρήσαμε και το Client-ID που λάβαμε κατά τη δημιουργία του λογαριασμού στο sandbox PayPal. Δημιουργήσαμε επίσης ένα component που είναι υπεύθυνο για την απόδοση των κουμπιών του PayPal και τη διαχείριση της διαδικασίας πληρωμής. Το PayPalButtons component είναι υπεύθυνο για την πραγματοποίηση συναλλαγών, που περιλαμβάνει τη δημιουργία παραγγελίας και την έγκριση της πληρωμής.
Με την ολοκλήρωση αυτών των βημάτων, καταφέραμε να ενσωματώσουμε επιτυχώς τη PayPal στην εφαρμογή μας, προσφέροντας μια ασφαλή εμπειρία πληρωμών στους χρήστες.

Κεφάλαιο 5ο: Υλοποίηση Back–End Μέρους Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων

Έχοντας παρουσιάσει και τις διαθέσιμες μεθόδους και τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη του Back – End μέρους μίας Διαδικτυακής Εφαρμογής, στο κεφάλαιο αυτό θα επιλέξουμε τη καταλληλότερη τεχνολογία για την υλοποίηση του Back – End μέρους της Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων. Στο παρόν, λοιπόν, κεφάλαιο θα δούμε βήμα προς βήμα τη διαδικασία ανάπτυξης του Back – End μέρους της Διαδικτυακής Εφαρμογής μας.

5.1 Επιλογή Τεχνολογίας Ανάπτυξης Back – End

Όπως είδαμε στο Κεφάλαιο 2, οι NodeJS και Java είναι κάποιες από τις τεχνολογίες που χρησιμοποιούνται ευρέως για την ανάπτυξη του Back – End μέρους των εκάστοτε Διαδικτυακών Εφαρμογών.

Για την επιλογή της καταλληλότερης από τις τρεις προς ανάπτυξη του Back – End μέρους της Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων, θα πρέπει να λάβουμε υπόψη τις εξής προϋποθέσεις:

- Να διαθέτει μεγάλη και ενεργή κοινότητα προγραμματιστών που το αναπτύσσουν και εμπλουτίζουν περαιτέρω διαρκώς, η οποία να προσφέρει συνεχώς νέα εργαλεία και επεκτάσεις.
- Να διευκολύνει την πρόσβαση στο επίπεδο των δεδομέων (Data layer)
- Να παρέχει γρήγορη διαχείριση πολλαπλών αιτημάτων ταυτόχρονα
- Να παρέχει εργαλεία σχετικά με την ασφάλεια της Διαδικτυακής Εφαρμογής
- Να είναι ευέλικτο ως προς την κλιμάκωση ώστε να διευκολύνεται η διαχείριση μεγάλου όγκου δεδομένων όταν αυτή είναι αναγκαία
- Η τεκμηρίωση της τεχνολογίας αυτής να είναι αναλυτική και επαρκής όσο προς νέους χρήστες καθώς και για πιο έμπειρους
- Να επιτρέπει την δημιουργία REST API καθώς και άλλων πρωτοκόλλων δοσοληψίας δεδομένων.

Λαμβάνοντάς τες υπόψη, επιλέξαμε το framework Node.js της JavaScript, καθώς:

- Διευκολύνει τον χειρισμό πολλαπλών αιτημάτων πολλαπλών Χρηστών ταυτόχρονα.
- Επιτρέπει την επαναχρησιμοποίηση κώδικα.
- Δεν αποκλείει κανένα αίτημα όταν ανταποκρίνεται σε άλλο.
- Επεξεργάζεται γρήγορα και αποτελεσματικά τις ροές δεδομένων.
- Υποστηρίζει τη δημιουργία REST APIs για τη δημιουργία αποτελεσματικής και απρόσκοπτης επικοινωνίας του Front – End μέρους με το Back – End μέρος μίας Διαδικτυακής Εφαρμογής.
- Διαθέτει μία αρκετά μεγάλη και ενεργή παγκόσμια κοινότητα προγραμματιστών που το αναπτύσσουν διαρκώς.

Δημιουργία Back – End Μέρους

Στη διαδικασία ανάπτυξης του Back-End, επιλέγουμε πλατφόρμες και τεχνολογίες που να ανταποκρίνονται στις ανάγκες και τις απαιτήσεις του έργου. Μια από τις δημοφιλέστερες πλατφόρμες για τη δημιουργία Back-End είναι η Node.js, την οποία και επιλέξαμε ως καταλληλότερη για την ανάπτυξη του Back-End μέρους της πτυχιακής μας, χρησιμοποιείται η πλατφόρμα ανάπτυξης λογισμικού Node της JavaScript και η διαχείριση των πακέτων γίνεται μέσω του NPM (Node Package Manager). Αρχικά, με την εντολή “npm install” δημιουργούμε το αρχείο “package.json”, το οποίο περιέχει τα μεταδεδομένα και

τις εξαρτήσεις (dependencies). Τα πακέτα που χρησιμοποιούνται στην πτυχιακή αυτή εργασία, φαίνονται στο Σχήμα.

5.2.1: Χρησιμοποιηθέντα πακέτα.

```
"dependencies": {
    "bcryptjs": "^2.4.3",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "express-async-errors": "^3.1.1",
    "express-fileupload": "^1.2.1",
    "express-mongo-sanitize": "^2.1.0",
    "express-rate-limit": "^5.4.1",
    "fs-extra": "^11.1.1",
    "helmet": "^4.6.0",
    "http-status-codes": "^2.1.4",
    "joi": "^17.4.0",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^8.4.0",
    "morgan": "^1.10.0",
    "nodemailer": "^6.9.7",
    "validator": "^13.6.0",
    "xss-clean": "^0.1.1"
},
"devDependencies": {
    "nodemon": "^2.0.9"
},
"engines": {
    "node": "14.x"
}
```

Σχήμα 41: Πακέτα

Ενδεικτικά, κάποιες από τις βιβλιοθήκες που χρησιμοποιήθηκαν και απεικονίζονται στο Σχήμα , είναι:
bcrypt.js: Μια δημοφιλής βιβλιοθήκη της JavaScript για τον κατακερματισμό και την επαλήθευση κωδικών πρόσβασης, σχεδιασμένη να δυσκολεύει τους εισβολείς να «σπάσουν» τους κωδικούς πρόσβασης.

dotenv: Μια λειτουργική μονάδα μηδενικής εξάρτησης που μας επιτρέπει να αποθηκεύουμε μεταβλητές περιβάλλοντος από ένα αρχείο .env, χρησιμοποιούμενο για την αποθήκευση ευαίσθητων δεδομένων, όπως κωδικούς πρόσβασης.

express: Ένα πλαίσιο διαδικτυακής εφαρμογής Back-End, χρησιμοποιούμενο για τη δημιουργία RESTful API με τη πλατφόρμα Node της JavaScript.

express-async-handler: Ένα ενδιάμεσο λογισμικό (middleware) για τον χειρισμό εξαιρέσεων σε ασύγχρονες διαδρομές και τη διαβίβασή τους στους χειριστές σφαλμάτων express.

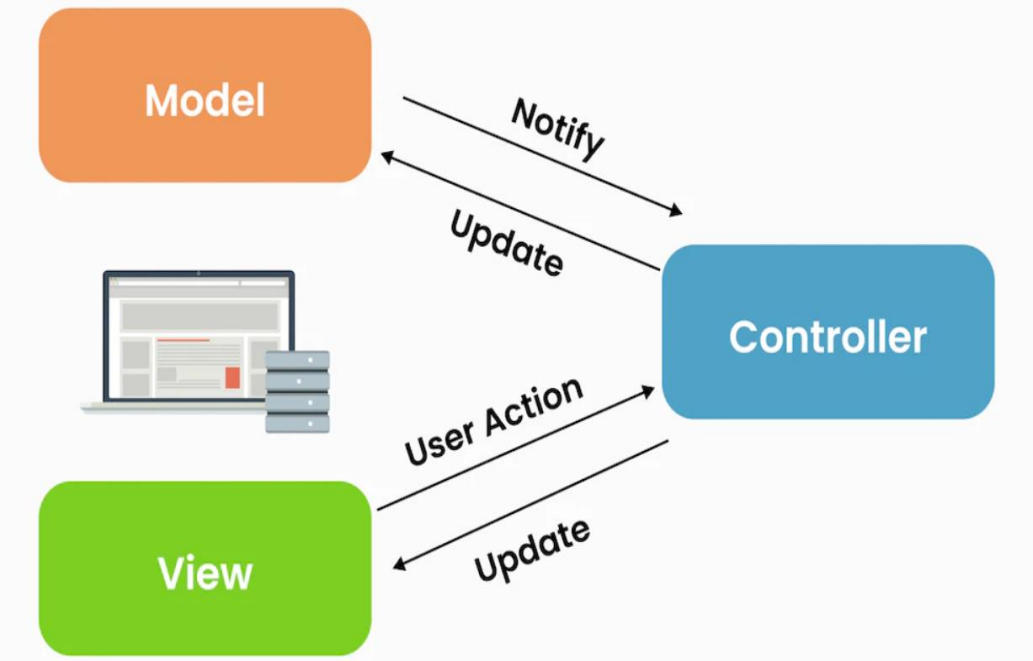
jsonwebtoken: Μια βιβλιοθήκη για τη δημιουργία JSON Web Token (JWT), που χρησιμοποιείται για τη δημιουργία δεδομένων με προαιρετική υπογραφή ή/και προαιρετική κρυπτογράφηση. Όταν ένας χρήστης

κάνει “login” στην εφαρμογή μας, δημιουργούμε ένα “token” – ένα “String” που περιέχει πληροφορίες για τον χρήστη – για λόγους ασφαλείας. Χρησιμοποιούμε ένα ιδιωτικό κλειδί για την κρυπτογράφηση του “token”, ώστε να αποτρέψουμε την πρόσβαση στην εφαρμογή μας από μη εξουσιοδοτημένα άτομα ή την υποκλοπή πληροφοριών.

nodemon: Ένα εργαλείο που παρακολουθεί τον κατάλογο του έργου και επανεκκινεί αυτόματα την εφαρμογή Node.js όταν ανιχνεύει αλλαγές.

5.2.2 Δομή Back – End

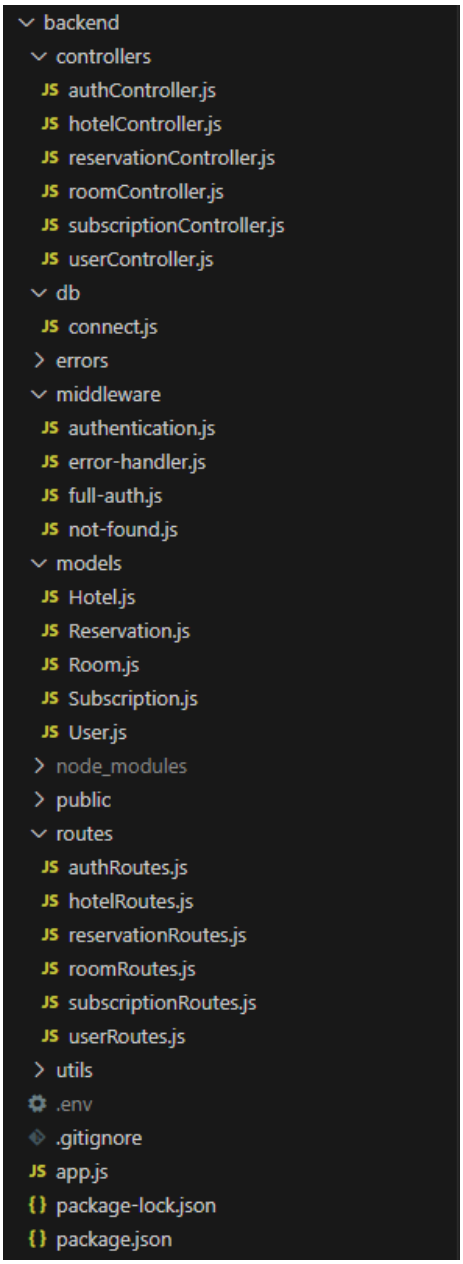
Αφού εγκαταστήσουμε όλα τα απαραίτητα πακέτα που θα χρησιμοποιήσουμε, δημιουργούμε τη δομή της Εφαρμογής μας. Το Back – End μέρος της Διαδικτυακής Εφαρμογής Ξενοδοχειακών Κρατήσεων δημιουργήθηκε ακολουθώντας την Αρχιτεκτονική MVC (Model – View – Controller), η οποία εικονίζεται στο Σχήμα .



Σχήμα 42:Αρχιτεκτονική MVC

Έτσι, όπως μπορούμε να δούμε στο Σχήμα , έχουμε δημιουργήσει:

- Έναν φάκελο “controllers”, στον οποίο ορίζουμε τις συναρτήσεις που ανταποκρίνονται σε κάθε “route”.
- Έναν φάκελο “db”, ο οποίος περιέχει το αρχείο “connect.js¨, που είναι υπεύθυνο για τη σύνδεση με τη βάση μας.
- Έναν φάκελο “errors”, στον οποίο δημιουργούμε μια προσωποποιημένη βιβλιοθήκη μέσω της οποίας παρέχονται τα μηνύματα σφαλμάτων σε όλη την εφαρμογή .
- Έναν φάκελο “middleware”, στον οποίο δημιουργούμε όλες τις ενδιάμεσες συναρτήσεις.
- Έναν φάκελο “models”, ο οποίος περιέχει όλα τα αρχεία με τα οποία ορίζουμε τη δομή (δηλαδή τα πεδία) των “collection” της βάσης μας.
- Έναν φάκελο “routes”, στον οποίο ορίζουμε τα “endpoint” της εφαρμογής μας.
- Και το αρχείο “app.js”, το οποίο είναι υπεύθυνο για την εκκίνηση του Διακομιστή (Server), τη χρήση Ενδιάμεσου Λογισμικού (middleware) και των διαδρομών (Routes).



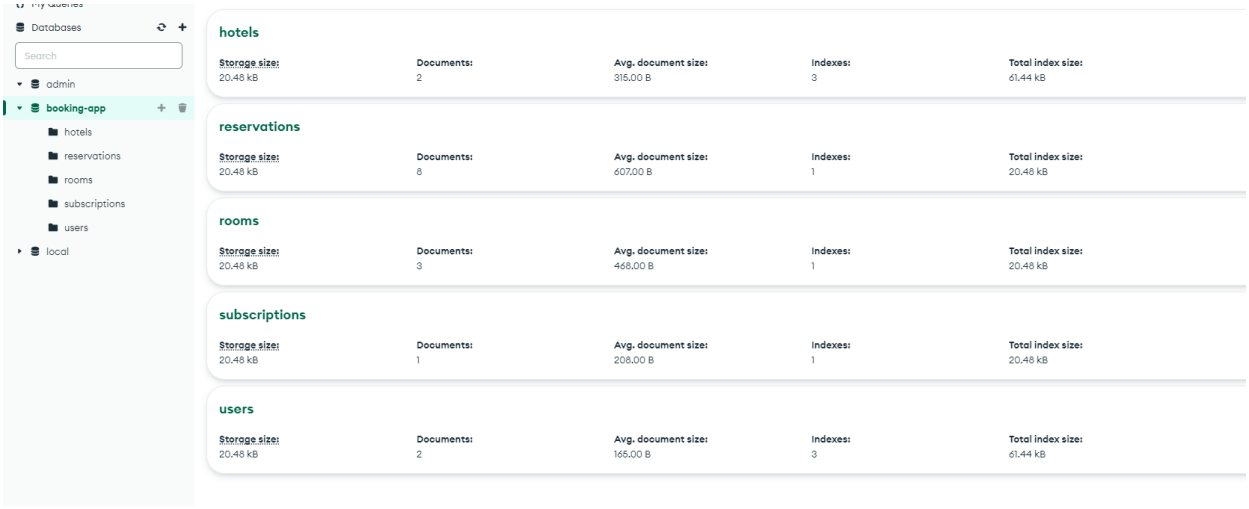
Σχήμα 43: Δομή αρχείων back-end

Μετά την εγκατάσταση όλων των πακέτων, δημιουργούμε το αρχείο “app.js” στον φάκελο “backend”. Για να μπορέσουμε να χρησιμοποιήσουμε το “express” ως Server, το εισάγουμε εκτελώντας την εντολή “const express = require('express')”. ‘Επειτα, δηλώνουμε την “PORT” που θα «τρέχει» ο Server μας. Λαμβάνοντας υπόψην ότι η Βιβλιοθήκη React δε «τρέχει» στην “PORT” 3000, διαλέγουμε μία διαφορετική. Δημιουργούμε, λοιπόν, μία νέα μεταβλητή με όνομα “PORT” και τής δίνουμε τη τιμή που ορίσαμε στο “.env” αρχείο μας. Εάν δεν έχουμε δηλώσει κάποια τιμή, τότε αυτή γίνεται 5000 με την εντολή “const PORT = process.env.PORT || 5000”. Έπειτα, “στήνουμε” το “express” με την εντολή “const app = express()”. Τέλος, για να ξεκινήσει ο Server μας, καλούμε την εντολή “app.listen(PORT, () => console.log(`server started on port ${PORT}`))” στο object έχουμε δημιουργήσει.

5.2.3 Σύνδεση σε Βάση Δεδομένων

Ως βάση δεδομένων χρησιμοποιήθηκαν οι MongoDB, mongoose και MongoDB Compass. Για να δημιουργήσουμε μία βάση δεδομένων MongoDB, πρέπει να επισκεφθούμε τη σελίδα “https://www.mongodb.com/”. Σε αυτήν, αφού συνδεθούμε με τα στοιχεία μας, θα χρειαστεί να δημιουργήσουμε ένα “Cluster”. Αφήνουμε τις default επιλογές του free tier και αλλάζουμε το όνομα του

“Cluster” σε “BOOKING-APP”. Στη συνέχεια, δημιουργούμε ένα νέο project με όνομα “booking-app”.



Σχήμα 44: Διαχείριση Βάσης Δεδομένων

Η σύνδεση της Εφαρμογής μας με τη Βάση Δεδομένων γίνεται στο αρχείο “./db/connect.js”.

```
const mongoose = require('mongoose')

const connectDB = (url) => {
    mongoose.set('strictQuery', true)
    return mongoose.connect(url)
}

module.exports = connectDB
```

Σχήμα 45: Σύνδεση Εφαρμογής και Βάσης Δεδομένων.

Το MongoDB προσφέρει Cloud Database. Έτσι, με τη χρήση του πακέτου “mongoose”, καλούμε τη συνάρτηση σύνδεσης, χρησιμοποιώντας τη διεύθυνση που αποθηκεύσαμε νωρίτερα. Για να καλέσουμε τη συνάρτησή μας όταν τρέχει ο Server μας, τη κάνουμε import στο αρχείο “app.js” και έπειτα τη καλούμε.

```
const start = async () => {
  try {
    await connectDB(process.env.MONGO_URL);
    app.listen(port, console.log(`listening to port ${port}`));
  } catch (error) {
    console.log(error);
  }
};
```

Σχήμα 46: Σύνταξη της συνάρτησης σύνδεσης.

5.2.4 Συλλογές (Models)

Εφόσον επιτευχθεί η σύνδεση με τη Βάση Δεδομένων, δημιουργούμε τις συλλογές (models) που χρειαζόμαστε. Για κάθε συλλογή δημιουργούμε και ένα μοντέλο (model). Πιο συγκεκριμένα, στον φάκελο “models”, δημιουργούμε πέντε μοντέλα:

1. Το μοντέλο “Hotel” για το κάθε ξενοδοχείο στο οποίο αποθηκεύονται τα εξής πεδία:
    - Το όνομα του ξενοδοχείου (**name**),

- Μια περιγραφή για το ξενοδοχείο (**description**),
- Το email επικοινωνίας του ξενοδοχείου (**email**),
- Την διεύθυνση του (**address**),
- Έναν αριθμό τράπεζας (**bankAccount**),
- Το τηλέφωνο επικοινωνίας (**phone**),
- Ένα αντικείμενο τύπου owner για τα στοιχεία του ιδιοκτήτη (**owner**),
- Τα URLs για τις φωτογραφίες του ξενοδοχείου (**imgs**)

2. Το μοντέλο "Reservation" για την κάθε κράτηση στο οποίο αποθηκεύονται τα εξής πεδία:
    - Το όνομα του πελάτη που πραγματοποιεί την κράτηση (**firstName**),
    - Το επώνυμο του πελάτη που πραγματοποιεί την κράτηση (**lastName**),
    - Το email επικοινωνίας του πελάτη που πραγματοποιεί την κράτηση (**email**),
    - Το τηλέφωνο επικοινωνίας του πελάτη (phone),
    - Ένα σχόλιο για την κράτηση (**comment**),
    - Το πλήθος των ανθρώπων που θα μείνουν στο δωμάτιο (**people**),
    - Τον τρόπο πληρωμής που έχει επιλεχθεί (**payment**),
    - Ένα αντικείμενο τύπου hotel για τα στοιχεία του ξενοδοχείου (**hotel**),
    - Ένα αντικείμενο τύπου room για τα δεδομένα του δωματίου που έχει επιλεχθεί (**room**),
    - Τις ημερομηνίες της κράτησης (**dates**),
    - Και μια τιμή **isCancelled** η οποία προσδιορίζει την εγκυρότητα της κράτησης
3. Το μοντέλο "Room" για κάθε δωμάτιο ξενοδοχείου στο οποίο αποθηκεύονται τα εξής πεδία:
    - Ο αριθμός του δωματίου (**roomNumber**),
    - Η περιγραφή του δωματίου (**description**),
    - Ο τύπος του δωματίου, για παράδειγμα μονόκλινο (**roomType**),
    - Ο τύπος του κρεβατιού (**bedType**)
    - Ο αριθμός των μπάνιων (**bathroom**),
    - Μια τιμή που προσδιορίζει αν επιτρέπονται κατοικίδια σε αυτό (**petsAllowed**),
    - Μια τιμή που προσδιορίζει αν το δωμάτιο ανήκει στα προτεινόμενα (**isFeatured**),
    - Την κανονική τιμή (**regularPrice**),
    - Την τιμή έκπτωσης (**discountPrice**),
    - Έναν αριθμό που προσδιορίζει την χωρητικότητα του δωματίου (**people**),
    - Ένα αντικείμενο τύπου **hotel** για τα δεδομένα του ξενοδοχείου στο οποίο ανήκει
    - Τα URLs για τις φωτογραφίες του δωματίου (**imgs**)
4. Το μοντέλο "User" για κάθε είδος χρήστη στο οποίο αποθηκεύονται τα εξής πεδία:
    - Το όνομα του χρήστη (**name**),
    - Το email επικοινωνίας του (**email**),
    - Ο κωδικός πρόσβασης του χρήστη (**password**)
    - Το πεδίου που προσδιορίζει τον ρόλο του, admin/owner/employee (**role**),
    - Ένα αντικείμενο τύπου **hotel** για τα δεδομένα του ξενοδοχείου στο οποίο ανήκει
5. Το μοντέλο "Subscription" για τις συνδρομές στο οποίο αποθηκεύονται τα εξής πεδία:
    - Το μοναδικό αναγνωριστικό του ιδιοκτήτη (**ownerId**),
    - Το όνομα του ιδιοκτήτη (**name**),
    - Την ημερομηνία έναρξης της συνδρομής (**subscriptionDate**),
    - Την ημερομηνία λήξης την συνδρομής (**subscriptionExpr**),
    - Το μοναδικό αναγνωριστικό της πληρωμής (**paymentId**)

Με τη χρήση του πακέτου "mongoose" δημιουργείται το κάθε "model". Αρχικά, δημιουργούμε μια μεταβλητή που περιέχει το "Schema", όπως φαίνεται στο Σχήμα 4.10. Η δομή των εγγράφων σε αυτήν την

συλλογή ορίζεται από κάθε “Schema” που αντιστοιχίζεται σε μια συλλογή MongoDB. Το εκάστοτε κλειδί στον κώδικα του “Schema” εκπροσωπεί και από μια ιδιότητα των εγγράφων μας και μεταφέρεται στον αντίστοιχο τύπο “Schema”. Για να μπορέσουμε να μετατρέψουμε σε μοντέλο οποιοδήποτε “Schema”, με σκοπό να εργαστούμε πάνω σε αυτό, θα πρέπει να το περάσουμε στη συνάρτηση “mongoose.model()” ως “mongoose.model(modelName, Schema)”.

```
const mongoose = require("mongoose");
const validator = require("validator");
const bcrypt = require("bcryptjs");

const UserSchema = mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please provide a name"],
    unique: true,
    minlength: 3,
    maxlength: 30,
  },
  email: {
    type: String,
    required: [true, "Please provide an email"],
    unique: true,
    validate: {
      validator: validator.isEmail,
      message: "Please provide a valid email",
    },
  },
  password: {
    type: String,
    required: [true, "Please provide a password"],
    minlength: 6,
  },
  role: {
    type: String,
    enum: ["admin", "owner", "employee"],
    default: "owner",
  },
  hotel: {
    hotel_id: { type: mongoose.Schema.Types.ObjectId, ref: "Hotel" },
    hotel_name: { type: String },
    hotel_owner: { type: String },
  },
});

UserSchema.pre("save", async function () {
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

UserSchema.methods.comparePassword = async function (canditatePassword) {
  const isMatch = await bcrypt.compare(canditatePassword, this.password);
  return isMatch;
};
module.exports = mongoose.model("User", UserSchema);
```

Σχήμα 47: Δημιουργία User model

5.2.5 Διαχείριση Σφαλμάτων

Στο πλαίσιο μιας διαδικτυακής εφαρμογής, η διαχείριση σφαλμάτων μπορεί να περιλαμβάνει τη χρήση προσαρμοσμένων κλάσεων σφαλμάτων για να αναπαραστήσουν συγκεκριμένα είδη σφαλμάτων με σαφή και κατανοητό τρόπο. Αυτό βελτιώνει την ανιχνευσιμότητα και τη διαχείριση σφαλμάτων, επιτρέποντας στην εφαρμογή να ανταποκρίνεται κατάλληλα σε διαφορετικές καταστάσεις σφάλματος και να παρέχει ακριβή μηνύματα στους χρήστες.

Στο πλαίσιο της παρούσας εργασίας υλοποιήθηκε μια σειρά από προσαρμοσμένες κλάσεις σφαλμάτων (BadRequestError, NotFoundError, UnauthenticatedError, UnauthorizedError), που κληρονομούν από την βασική κλάση CustomAPIError, η οποία αντλεί δεδομένα από την ενσωματωμένη κλάση Error της JavaScript. Κάθε μία από αυτές τις κλάσεις αντιπροσωπεύει ένα συγκεκριμένο τύπο HTTP σφάλματος, προσθέτοντας έναν αντίστοιχο status code. Αυτές οι κλάσεις σφαλμάτων χρησιμοποιούνται για την καλύτερη διαχείριση σφαλμάτων στην εφαρμογή, επιτρέποντας την δημιουργία σαφών και κατανοητών μηνυμάτων σφάλματος που διευκολύνουν την ανίχνευση και επίλυση προβλημάτων, ενώ παράλληλα βελτιώνουν την εμπειρία του χρήστη και την ασφάλεια της εφαρμογής.

```
const CustomAPIError = require('./custom-api')
const UnauthenticatedError = require('./unauthenticated')
const NotFoundError = require('./not-found')
const BadRequestError = require('./bad-request')
const UnauthorizedError = require('./unauthorized')
module.exports = {
    CustomAPIError,
    UnauthenticatedError,
    NotFoundError,
    BadRequestError,
    UnauthorizedError,
}
```

Σχήμα 48:Σύνταξη “CustomApiError”

5.2.6 Authentication & Authorization

Για να εξασφαλίσουμε την ασφαλή χρήση της Εφαρμογής μας από τους Χρήστες, υλοποιήσαμε JSON Web Tokens (JWT). Χρησιμοποιώντας τα Tokens, έχουμε τη δυνατότητα να εξακριβώσουμε αν ένας χρήστης έχει συνδεθεί και αν το “Token” του είναι έγκυρο – δηλαδή αν έχει λήξει ή όχι.

Για τον έλεγχο των JSON Web Tokens, δημιουργούμε ένα ακόμη “middleware”, το οποίο θα ελέγχει την εγκυρότητά τους. Έτσι, στον φάκελο “middlware” δημιουργούμε ένα νέο αρχείο “full-auth.js”.Για να επιτευχθεί η πρόσβαση σε “routes” που είναι μόνο για χρήστες που έχουν συνδεθεί ή για χρήστες που έχουν συγκεκριμένο ρόλο στην εφαρμογή μας, από το Front- End μέρος της, πρέπει να στέλνουμε και το token που δημιουργείται όταν ο χρήστης κάνει “login”. Το Token θα στέλνεται ως “header” με όνομα “authorization” και η τιμή του θα έχει την μορφή “Bearer”.

Δημιουργήσαμε μία άσυγχρονη συνάρτηση “authenticateUser()”, η οποία ελέγχει εάν υπάρχει έγκυρο token είτε στην κεφαλίδα `Authorization` είτε στα cookies του αιτήματος. Αν βρεθεί το token, ελέγχει την εγκυρότητά του και αν είναι έγκυρο, προσθέτει τις πληροφορίες του χρήστη (userId και ρόλο) στο αντικείμενο `req` για να είναι διαθέσιμες στις επόμενες λειτουργίες. Αν δεν βρεθεί ή δεν είναι έγκυρο το token, πετάει ένα σφάλμα μη εξουσιοδοτημένης πρόσβασης. Στη συνέχεια δημιουργήσαμε μια ακόμη ασύχρονη συνάρτηση “authorizeRoles()”, όπου ελέγχει αν ο ρόλος του χρήστη, που είναι αποθηκευμένος στο αντικείμενο `req` από το προηγούμενο middleware, ανήκει στους επιτρεπόμενους ρόλους που παρέχονται ως παράμετροι. Αν ο ρόλος του χρήστη δεν είναι μέσα στους επιτρεπόμενους, πετάει ένα σφάλμα μη εξουσιοδοτημένης πρόσβασης.

Αυτές οι λειτουργίες εξασφαλίζουν ότι μόνο οι εξουσιοδοτημένοι και πιστοποιημένοι χρήστες μπορούν να προσπελάσουν συγκεκριμένες διαδρομές στην εφαρμογή.

5.2.7 Routes & Controllers

Το κεφάλαιο "Routes & Controllers" αναλύει τις βασικές έννοιες και λειτουργίες που σχετίζονται με τη δρομολόγηση και τον έλεγχο της ροής δεδομένων σε μια διαδικτυακή εφαρμογή. Οι Routes (Διαδρομές) καθορίζουν πώς τα αιτήματα HTTP αντιστοιχίζονται σε συγκεκριμένες λειτουργίες του διακομιστή, επιτρέποντας την πλοήγηση και την αλληλεπίδραση με την εφαρμογή. Για κάθε πόρο που θα χρειαστούμε, δημιουργούμε ξεχωριστές διαδρομές ώστε να διαχειρίζονται τις μεθόδους “GET”, “PUT”, “DELETE”, “UPDATE” που θα χρησιμοποιήσουμε. Όπως μπορούμε να δούμε στο Σχήμα 4.15, για να μπορέσουμε να χρησιμοποιήσουμε τα Routes που δημιουργούμε στο αρχείο “app.js”, ορίζουμε σε ποιο “path” «ακούει» το κάθε Route. Οι Controllers (Ελεγκτές) είναι υπεύθυνοι για την επεξεργασία αυτών των αιτήσεων, χειρίζοντας την επιχειρηματική λογική και την επικοινωνία με τη βάση δεδομένων. Μαζί, οι διαδρομές και οι ελεγκτές δημιουργούν τη ραχοκοκαλιά της αρχιτεκτονικής μιας εφαρμογής, επιτρέποντας την οργανωμένη και αποδοτική διαχείριση των αιτημάτων των χρηστών και των δεδομένων.

Για κάθε πόρο που θα χρειαστούμε, δημιουργούμε ξεχωριστές διαδρομές (Routes), ώστε να διαχειρίζεται τις μεθόδους “GET”, “PUT”, “DELETE”, “UPDATE” που θα χρησιμοποιήσουμε. Συνολικά έχουμε τις εξής έξι λειτουργίες:

- Για την διαχείριση του JWT (authRoutes.js),
- Καταχώρηση και επεξεργασία ξενοδοχείων (hotelRoutes.js)
- Διαχείριση και καταχώρηση κρατήσεων (reservationRoutes.js),
- Διαχείριση και καταχώρηση δωματίων (roomRoutes.js),
- Διαχείριση συνδρομών (subscriptionRoutes.js),
- Διαχείριση χρηστών (userRoutes.js).

Στην Εφαρμογή μας δημιουργούμε και χρησιμοποιούμε έξι (6) controllers:

- Auth Controller: Είναι η μέθοδος καταχώρισης πληροφοριών χρήστη που θέλουμε να γίνει μέσω της εφαρμογής μας, η οποία αποτελείται από τις εξής συναρτήσεις: “register()”, “login()”, “logout()” και “updatePassword()”.Την κυριοτερη συνάρτηση αποτελέι η “register()” η οποία αρχικά παίρνει τις τιμές name, email και password από το body του αιτήματος (req.body) και ελέγχει αν τα πεδία name, email και password είναι συμπληρωμένα, διαφορετικά εμφανίζει το κατάλληλο σφάλμα.Έπειτα ελέγχει αν το email είναι ήδη καταχωρημένο στη βάση δεδομένων και εφόσον δεν είναι δημιουργεί έναν νέο χρήστη στη βάση δεδομένων.Τέλος, δημιουργεί ένα JSON Web Token (JWT) και το επισυνάπτει ως cookie στο response και επιστρέφει τον νέο χρήστη.
- Hotel Controller: Είναι η μέθοδος διαχείρισης πληροφοριών ξενοδοχείου στην εφαρμογή μας, η οποία αποτελείται από τις εξής συναρτήσεις: “createHotel()”, “uploadHotelImages()”, “updateHotel()”, “getAllHotels()”, “getMyHotels()”, “getSingleHotel()” και “deleteHotel()”.Την κυριότερη συνάρτηση αποτελεί η “createHotel()” η οποία αρχικά παίρνει τις τιμές name, description, email, address, phone και imgs από το body του αιτήματος (req.body) και ελέγχει αν τα πεδία name, description, email, address και phone είναι συμπληρωμένα, διαφορετικά εμφανίζει το κατάλληλο σφάλμα. Έπειτα ελέγχει αν το όνομα του ξενοδοχείου είναι ήδη καταχωρημένο στη βάση δεδομένων και, εφόσον δεν είναι, δημιουργεί έναν νέο ξενοδοχείο στη βάση δεδομένων. Τέλος, επισυνάπτει το token του χρήστη ως cookie στην απάντηση και επιστρέφει το νέο ξενοδοχείο.

- Reservation Controller: Αυτός ο controller είναι υπεύθυνος για τη διαχείριση των κρατήσεων στην εφαρμογή και αποτελείται από τις εξής συναρτήσεις “getReservations()”, “getMyReservations()”, “createReservation()”, “getSingleReservation()”, “updateReservation()”, “cancelReservation()” και την “deleteReservation()”. Την κυριότερη αποτελεί η “createReservation()” που είναι υπεύθυνη για τη δημιουργία μιας νέας κράτησης στο σύστημα.Αρχικά, λαμβάνει τις πληροφορίες της κράτησης από το body του αιτήματος και πραγματοποιεί αρκετούς ελέγχους για να εξασφαλίσει την ορθότητα των παρεχόμενων δεδομένων. Δηλαδή, ελέγχει αν όλα τα απαραίτητα πεδία όπως το όνομα, το επώνυμο, το email, το τηλέφωνο, οι επισκέπτες, η πληρωμή, το ξενοδοχείο, το δωμάτιο και οι ημερομηνίες έχουν συμπληρωθεί. Αν δεν έχουν συμπληρωθεί, επιστρέφει αντίστοιχο σφάλμα. Ελέγχει αν το δωμάτιο είναι διαθέσιμο για τις συγκεκριμένες ημερομηνίες, αν το δωμάτιο είναι ήδη κλεισμένο για αυτές τις ημερομηνίες, επιστρέφει σφάλμα "Room is already booked", δημιουργεί την κράτηση στη βάση δεδομένων με βάση τις παρεχόμενες πληροφορίες, όπως το όνομα, το επώνυμο, το email, το τηλέφωνο, οι επισκέπτες, η πληρωμή, το ξενοδοχείο, το δωμάτιο και οι ημερομηνίες.Και τέλος αποστέλλει ένα email επιβεβαίωσης στον χρήστη για την επιτυχή δημιουργία της κράτησης και επιστρέφει το ID της νέας κράτησης ως απάντηση.

- Room Controller: Είναι η μέθοδος διαχείρισης δωματίων ενός ξενοδοχείου που θέλουμε να επιτύχουμε μέσω την εφαρμογής μας, οι οποία αποτελείται από τις εξής συναρτήσεις: “getRooms()”, “getMyRooms()”, “createRoom()”, “getSingleRoom()”, “updateRoom()”, “deleteRoom()”, “uploadRoomImages()” και “getAvailableRooms()”. Την κυριότερη συνάρτηση αποτελεί η “createRoom()” η οποία αναλαμβάνει τη δημιουργία ενός νέου δωματίου στη βάση δεδομένων. Αρχικά, λαμβάνει τις πληροφορίες του δωματίου από το αίτημα του χρήστη, όπως αριθμός δωματίου, περιγραφή, τύπος κλπ. Στη συνέχεια, ελέγχει αν όλα τα απαραίτητα πεδία έχουν συμπληρωθεί και αν δεν λείπει κάποια πληροφορία που απαιτείται. Έπειτα, ελέγχει αν το δωμάτιο που προσπαθεί να δημιουργήσει ο χρήστης υπάρχει ήδη στη βάση δεδομένων. Αν όλα αυτά είναι επιτυχή, δημιουργεί το νέο δωμάτιο και επιστρέφει τις πληροφορίες του στον χρήστη μαζί με την κατάλληλη επικύρωση και τον κωδικό κατάστασης HTTP 201 (Created). Σε περίπτωση που δεν πληρούνται κάποιες από τις προϋποθέσεις, η μέθοδος επιστρέφει τον κατάλληλο κωδικό κατάστασης HTTP 400 (Bad Request) και ένα μήνυμα σφάλματος που περιγράφει το πρόβλημα.
- Subscription Controller: Είναι η μέθοδος διαχείρισης των συνδρομών του συστήματος και αποτελείται από τις εξής συναρτήσεις: “getSubscriptions()”, “createSubscription()”, “getSubscription()”, “getMySubscription()”, “updateSubscription()”, “renewSubscription()”, “cancelSubscription()” και “deleteSubscription()”. Το βασικότερο της στοιχείο είναι η συνάρτηση “createSubscription()”, η οποία αναλαμβάνει τη δημιουργία μιας νέας συνδρομής. Αρχικά, εξάγει τον τίτλο της συνδρομής και το id του χρήστη από το cookie του αιτήματος. Στη συνέχεια, ελέγχει αν ο χρήστης έχει ήδη συνδρομή, και αν ναι, επιστρέφει ένα σφάλμα. Αν όχι, δημιουργεί μια νέα συνδρομή με τις πληροφορίες που παρέχονται και τις αποθηκεύει στη βάση δεδομένων. Τέλος, επιστρέφει τις πληροφορίες της νέας συνδρομής στον χρήστη.
- User Controller: Είναι η μέθοδος διαχείρισης των χρηστών της παρούσας εφαρμογής και αποτελείται από τις εξής συναρτήσεις: “getAllUsers()”,“getSingleUser()”, “showCurrentUser()”, “updateUser()”, “updateUserPassword()”, “registerEmployee()”, “getMyEmployees()” και “deleteUser()”. Μια από τις βασικές συναρτήσεις είναι η “registerEmployee()”, η οποία χρησιμοποιείται για την εγγραφή ενός νέου χρήστη τύπου εργαζόμενου. Σε αυτή τη μέθοδο, εξάγονται τα στοιχεία όπως το όνομα, το email και ο κωδικός από το body του αιτήματος και γίνονται οι ανάλογοι έλεγχοι για την ορθή συμπλήρωση και την απαιτούμενη μορφή τους. Στη συνέχεια, ελέγχεται εάν το email του νέου χρήστη υπάρχει ήδη στη βάση δεδομένων. Έπειτα, εντοπίζεται σε ποιο ξενοδοχείο πρόκειται να εργαστεί ο νέος εργαζόμενος με βάση το hotelId που παρέχεται. Αν το ξενοδοχείο δεν υπάρχει, επιστρέφεται ένα σφάλμα. Στη συνέχεια, ο νέος χρήστης δημιουργείται στη βάση δεδομένων με τον ρόλο "employee" και τα σχετικά στοιχεία του ξενοδοχείου στο οποίο εργάζεται. Τέλος, επιστρέφεται ένα μήνυμα επιτυχίας μαζί με το όνομα του ξενοδοχείου στο οποίο έχει γίνει η εγγραφή.

```
const registerEmployee = async (req, res) => {
    const { name, email, password, hotelId } = req.body

    // check if name is provided
    if (!name) {
        throw new CustomError.BadRequestError('Please provide a name')
    }

    // check if email is provided
    if (!email) {
        throw new CustomError.BadRequestError('Please provide an email')
    }

    // check if password is provided
    if (!password) {
        throw new CustomError.BadRequestError('Please provide a password')
    }

    // check if user already exists
    const emailAlreadyExists = await User.findOne({ email })
    if (emailAlreadyExists) {
        throw new CustomError.BadRequestError('Email already exists')
    }

    // Find in which hotel he is
    const hotel = await Hotel.findById(hotelId)

    if (!hotel) {
        throw new CustomError.BadRequestError('Hotel doesnt exists')
    }

    // create user
    await User.create({
        name,
        email,
        password,
        role: 'employee',
        hotel: { hotel_id: hotel, hotel_name: hotel.name, hotel_owner: hotel.owner.owner_name },
    })

    res.status(StatusCodes.CREATED).json({ msg: `User created for hotel ${hotel.name}` })
}
```

Σχήμα 49: Σύνταξη “User Controller”

5.2.8 Αναλυτική περιγραφή API

Το API του server έχει ως prefix το “api/” και αρχικοποιούμε πέντε είδη routes, auth,users,hotels,rooms, reservations, subscriptions. Τα οποία έχουμε χωρίσει με βάση τις ενέργειες που μπορούν να γίνουν ανάλογα με το model που σχετίζονται και την ομαδοποίηση ενεργειών που συνδέονται εννοιολογικά (πχ login/logout).

5.2.8.1 User API

Την διαδρομή “api/users” ακολουθούν μέθοδοι οι οποίες σχετίζονται με την ανάκτηση, την αποθήκευση, την επεξεργασία και την διαγραφή δεδομένων που αφορούν το μοντέλο “Users”. Οι συγκεκριμένες διαδρομές χρειάζονται authorization, δηλαδή αφορούν μόνο τους χρήστες που έχουν κάνει σύνδεση στην εφαρμογή και τους έχει καταχωρηθεί ένα valid token συνεπώς είναι οι χρήστες που έχουν κάποιον ρόλο (owner, admin).

Endpoint	Http Method	Params	Body	Response	Errors
api/users/	GET	user,hotel, role (optional)	-	List of users	404
api/users/:id	GET	id	-	User details	404,

					401
api/users/showMe	GET	-	-	Current user details	-
api/users/registerEmployee	POST	-	name,email, password, hotelId	Employee created	400
api/users/getMyEmployees	GET	Search criteria	-	List of employees	404
api/users/deleteUser	DELETE	id	-	User successfully deleted	404, 401
api/users/updateUser	PATCH	-	name,email	User updated	-
api/users/updateUserPassword	PATCH	-	oldPassword, newPassword	Password updated	-

Πίνακας 1.Users API

Σε αυτό το σημείο θα θέλαμε να περιγράψουμε τον σκοπό κάθε endpoint. Αρχικά, μέσω του “api/users/” ανακτούμε μια λίστα με τους χρήστες της εφαρμογής και μπορεί να δεχτεί προαιρετικά παραμέτρους όπως ένα όνομα (name), έναν ρόλο (role) ή και ένα όνομα ξενοδοχείου (hotel name) για να λειτουργήσουν ως φίλτρα στα δεδομένα της λίστας που θα εμφανιστεί. Παρεμφερή λειτουργία έχει και το endpoint “api/users/:id” το οποίο είναι υπεύθυνο για την ανάκτηση των δεδομένων ενός χρήστη με βάση το “id” του. Όπως επίσης και το endpoint ”api/users/showMe” με το οποίο γίνεται η ανάκτηση των δεδομένων του συνδεδεμένου χρήστη. Έπειτα το “api/users/registerEmployee” χρησιμοποιείται για την καταχώρηση ενός “employee” για ένα ξενοδοχείο και απαιτούνται τα εξής πεδία name, email, password, hotelId. Επιπλέον για την εμφάνιση της λίστας των employees ενός “owner” χρησιμοποιείται το ”api/users/getMyEmployees”. Το “api/users/deleteUser” αφορά την διαγραφή ενός χρήστη από την εφαρμογή και δέχεται ως παράμετρο υποχρεωτικά το “id” του χρήστη. Τέλος, δυο endpoints που σχετίζονται με την επεξεργασία δεδομένων του χρήστη είναι το “api/users/updateUser” μέσω του οποίου επιτυγχάνεται η επεξεργασία των στοιχείων του χρήστη (πχ email) και το api/users/updateUserPassword” το οποίο ενημερώνει τον κωδικό πρόσβασης του χρήστη.

5.2.8.2 Hotels API

Την διαδρομή “api/hotels” ακολουθούν μέθοδοι οι οποίες σχετίζονται με την ανάκτηση, την αποθήκευση, την επεξεργασία και την διαγραφή δεδομένων που αφορούν το μοντέλο “Hotels”. Οι συγκεκριμένες διαδρομές χρειάζονται authorization, δηλαδή αφορούν μόνο τους χρήστες που έχουν κάνει σύνδεση στην εφαρμογή και τους έχει καταχωρηθεί ένα valid token συνεπώς είναι οι χρήστες που έχουν κάποιον ρόλο (owner, admin).

Endpoint	**Http Method**	**Params**	**Body**	**Response**	**Errors**
api/hotels/	GET	name, email, address	-	List of hotel objects	-

api/hotels/createHotel	POST	-	name, description, ,email, address, phone, imgs, bankAccount	Created hotel object	400
api/hotels/uploadHotelmages	POST	hotelId	images	Image URL	400
api/hotels/getMyHotels	GET	Search criteria	-	List of the hotels owned by the user	404
api/hotels/getSingleHotel	GET	hotelId	-	Single hotel object	404
api/hotels/updateHotel	PATCH	-	hotelId, createdBy [fields to update]	Updated hotel object	404, 401
api/hotels/deleteHotel	DELETE	hotelId	-	Deleted hotel object	404, 401

Πίνακας 2. Hotels API

Το Hotels API παρέχει κάποια endpoints για την διαχείριση διαφόρων λειτουργιών σχετικά με το ξενοδοχείο. Το "api/hotels/" επιτρέπει στον "admin" να ανακτήσει μια λίστα με όλα τα ξενοδοχεία, επιτρέποντας χρήση φίλτρου με βάση είτε το name, είτε το email αλλά είτε και το address του ξενοδοχείου. Έπειτα το "api/hotels/createHotel" παρέχει την δυνατότητα στον "owner" να δημιουργήσει ένα ξενοδοχείο εισάγοντας τα απαραίτητα στοιχεία όπως name, description, email, address, phone, imgs, bankAccount. Στη συνέχεια το endpoint "api/hotels/uploadHotelImages" είναι υπεύθυνο για την μεταμόρφωση εικόνων για ένα συγκεκριμένο ξενοδοχείο. Ακόμη, μέσω του "api/hotels/getMyHotel" ο "owner" ανακτά μια λίστα με τα ξενοδοχεία του και του δίνεται η δυνατότητα της χρήσης φίλτρου με βάση το όνομα του ξενοδοχείου. Έτσι, ως επόμενο βήμα για την εμφάνιση των λεπτομερειών ενός συγκεκριμένου ξενοδοχείου χρησιμοποιείται το "api/hotels/getSingleHotel". Το endpoint "api/hotels/updateHotel" αφορά την ενημέρωση των στοιχείων του ξενοδοχείου είτε από τον "owner" είτε από τον "employee" του συγκεκριμένου ξενοδοχείου. Τέλος, το endpoint "api/hotels/deleteHotel" επιτρέπει στους "owners" ή τους "employees" να διαγράψουν ένα ξενοδοχείο με βάση το "id" του ξενοδοχείου.

5.2.8.3 Rooms API

Την διαδρομή "api/rooms" ακολουθούν μέθοδοι οι οποίες σχετίζονται με την ανάκτηση, την αποθήκευση, την επεξεργασία και την διαγραφή δεδομένων που αφορούν το μοντέλο "Rooms". Οι παρακάτω διαδρομές (endpoints) χωρίζονται σε δύο είδη, σε αυτές που η ταυτοποίηση του χρήστη δεν είναι απαραίτητη ("getAvailableRooms", "getAllHotelRooms") και σε αυτές που αφορούν μόνο εξουσιοδοτημένους χρήστες με ορισμένο ρόλο.

Endpoint	**Http Method**	**Params**	**Body**	**Response**	**Errors**
api/rooms/	GET	roomNumber	-	Array of room objects	
api/rooms /getAllHotelRooms	GET	hotelName	-	Array of room objects	
api/rooms /getMyRooms	GET	roomNumber hotelName	-	Array of rooms owned by the	404

				user or their hotel	
api/rooms /createRoom	POST		roomNumber, description, roomType, bedType, bathroom, petsAllowed, regularPrice, discountPrice, hotelId, isFeatured, people	Created room object	400
api/rooms /updateRoom	PATCH		id, roomNumber, description, roomType, bedType, bathroom, petsAllowed, regularPrice, discountPrice, isFeatured, people, imgs, hotelId	Updated room object	404
api/rooms /uploadRoomImage	POST	hotelId, roomId	images	Array of image URLs	400
api/rooms /getAvailableRooms	GET		id, startDate, endDate, adults, children	Array of available room objects	404
api/rooms/:id	GET DELETE	id		Room object	401, 404

Πίνακας 3. Rooms API

Αναλυτικότερα, το πρώτο endpoint (διαδρομή) “api/rooms/” χρησιμοποιείται για την ανάκτηση όλων των δωματίων και μπορεί να γίνει χρήση των παραμέτρων προαιρετικά ως φίλτρα. Το “api/rooms/getAllHotelRooms” εξυπηρετεί την ανάκτηση των δωματίων ενός ξενοδοχείου μόνο με βάση το όνομα που παρέχει ο χρήστης. Επίσης, στην διαδρομή “api/rooms/getMyRooms” γίνεται η ανάκτηση μιας λίστας δωματίων που ανήκουν στο ξενοδοχείο του συνδεδεμένου χρήστη. Στη συνέχεια η δημιουργία ενός δωματίου αλλά και η προσθήκη φωτογραφιών σε αυτό γίνονται μέσω του “api/rooms/createRoom” και του “api/rooms/uploadRoomImages” όπως και η ενημέρωση των στοιχείων ενός συγκεκριμένου δωματίου γίνεται μέσω του “api/rooms/updateRoom”. Το endpoint “api/rooms/:id” εξυπηρετεί είτε την ανάκτηση των δεδομένων ενός συγκεκριμένου δωματίου είτε την διαγραφή αυτού του δωματίου ανάλογα με το είδος της Http μεθόδου. Τέλος, υπεύθυνο για την ανάκτηση των διαθέσιμων δωματίων ανάλογα με ένα εύρος ημερομηνιών και το πλήθος τον ανθρώπων είναι το “api/rooms/getAvailableRooms”.

5.2.8.4 Reservations API

Την διαδρομή “api/reservations” ακολουθούν μέθοδοι οι οποίες σχετίζονται με την ανάκτηση, την αποθήκευση, την επεξεργασία και την διαγραφή δεδομένων που αφορούν το μοντέλο “Reservations”. Οι παρακάτω διαδρομές (endpoints) χωρίζονται σε δύο είδη, σε αυτές που η ταυτοποίηση του χρήστη δεν είναι απαραίτητη (“createReservations”, “/:id”) και σε όλες τις υπόλοιπες που αφορούν μόνο εξουσιοδοτημένους χρήστες με ορισμένο ρόλο.

Endpoint	Http Method	Params	Body	Response	Errors

/api/reservations/	GET	startDate, endDate, hotel, room	-	Array of reservation objects	-
/api/reservations /getMyReservations	GET	startDate, endDate, hotel, room	-	Array of reservation objects	-
/api/reservations /createReservation	POST	-	firstName, lastName, email, phone, comment, people, payment, hotel, room, dates	Created: Reservation ID	400
/api/reservations/:id	GET	id	-	Reservation object	404
/api/reservations /updateReservation	PATCH	-	id, transactionId	Updated reservation object	401
/api/reservations /cancelReservation	PATCH	id	-	Canceled reservation	401
/api/reservations/:id	DELETE	id	-	Deleted reservation object	401

Πίνακας 4.Reservations API

Έχοντας κατανοήσει τα δεδομένα του παραπάνω πίνακα θα αναλύσουμε τον σκοπό κάθε endpoint (διαδρομής). Αρχικά με το endpoint “api/reservations/” γίνεται ανάκτηση όλων των κρατήσεων με προαιρετική χρήση των πεδίων της ημερομηνίας έναρξης, λήξης, το όνομα του ξενοδοχείου και το δωμάτιο ως φίλτρων. Όπως επίσης με το “api/reservations/getMyReservations” γίνεται ανάκτηση, από τον ιδιοκτήτη του ξενοδοχείου ή από τον εξουσιοδοτημένο υπάλληλο, μιας λίστας των κρατήσεων που αφορούν τα δικά τους ξενοδοχεία. Η δημιουργία μιας κράτησης από μη εξουσιοδοτημένο χρήστη γίνεται μέσω του “api/reservations/createReservation” αποστέλλοντας και τα απαραίτητα πεδία που αναφέρονται στον πίνακα. Ακόμη, τα endpoints “api/reservations/updateReservation” και “api/reservations/cancelReservation” έχουν ως σκοπό την ενημέρωση και την διαγραφή μιας συγκεκριμένης κράτησης. Τέλος, το “api/reservations/:id’ αποσκοπεί είτε στην ανάκτηση των δεδομένων της συγκεκριμένης κράτησης είτε στην διαγραφή αυτής της κράτησης με βάση την http μέθοδο του αιτήματος.

5.2.8.5 Subscriptions API

Την διαδρομή “api/subscriptions” ακολουθούν μέθοδοι οι οποίες σχετίζονται με την ανάκτηση, την αποθήκευση, την επεξεργασία και την διαγραφή δεδομένων που αφορούν το μοντέλο “Subscriptions”. Οι συγκεκριμένες διαδρομές χρειάζονται authorization, δηλαδή αφορούν μόνο τους χρήστες που έχουν κάνει σύνδεση στην εφαρμογή και τους έχει καταχωρηθεί ένα valid token συνεπώς είναι οι χρήστες που έχουν κάποιον ρόλο (owner, admin).

Endpoints	**Http Method**	**Parms**	**Body**	**Response**	**Errors**

api/subscriptions/	GET	name	-	Array of subscription objects	-
api/subscriptions/createSubscription	POST	paymentId	-	Created subscription object	400
api/subscriptions/getMySubscription	GET	-	-	User's subscription object	404
api/subscriptions/renewSubscription	POST	id, paymentId	-	Update subscription object	400
api/subscriptions/cancelSubscription	PATCH	id	-	Canceled subscription	400
api/subscriptions/:id	GET	id	-	Subscription object	404
api/subscriptions/:id	DELETE	id	-	Deleted subscription object	404
api/subscriptions/:id	PATCH	id	-	Update dsubscription object	400

Πίνακας 5. Subscriptions API

Πιο αναλυτικά, η διαδρομή “api/subscriptions/” ανακτά μια λίστα από τις συνδρομές. Σκοπός των διαδρομών “api/subscriptions/createSubscription” και “api/subscriptions/renewSubcsription” είναι η δημιουργία συνδρομής για τον χρήστη και η ανανέωση αυτής αντίστοιχα. Η διαδρομή “api/subscriptions/getMySubscription” είναι υπεύθυνη για την ανάκτηση των δεδομένων της συνδρομής του εκάστοτε συνδεδεμένου χρήστη. Επίσης, η διαδρομή “api/subscriptions/cancelSubscription” έχει ως αποτέλεσμα την ακύρωση της συγκεκριμένης συνδρομής. Τέλος, η διαδρομή “api/subscriptions/:id” στην οποία η παράμετρος “id” είναι υποχρεωτική έχει ως στόχο είτε την ανάκτηση είτε την διαγραφή αλλά είτε την ενημέρωση της συγκεκριμένης συνδρομής.

5.2.8.6 Auth API

Την διαδρομή “api/auth” την ακολουθούν μέθοδοι οι οποίες σχετίζονται με την ταυτοποίηση του χρήστη στην εφαρμογή μας.

Endpoints	Http Method	Parms	Body	Response	Errors
/register	POST	-	name, email, password	Registered user object with JWT cookie	400
/login	POST	-	email, password	Created: User object with JWT token and cookie	400
/logout	GET	-	-	{ msg: 'User logged out' }	-
/updatePassword	PATCH	-	id, password	Updated password field on User object	401

Πίνακας 6. Autth API

Συγκεκριμένα, το endpoint “api/auth/register” χρησιμοποιείται για την εγγραφή ενός χρήστη και την ταυτοποίηση του. Μετά, το endpoint “api/auth/login” είναι υπεύθυνο για την ταυτοποίηση του χρήστη με την χρήση του JWT και αντίστοιχα το “api/auth/logout” αφαιρεί το cookie που χρησιμοποιούμε για την ταυτοποίηση του χρήστη και εξυπηρετεί δηλαδή στην αποσύνδεση του. Τέλος, το endpoint “api/auth/updatePassword” χρησιμοποιείται για την ενημέρωση του κωδικού πρόσβασης ενός συνδεδεμένου χρήστη.

Κεφάλαιο 6ο:Συμπεράσματα από την Υλοποίηση της Εφαρμογής και προτάσεις βελτίωσης.

Η ανάπτυξη της εφαρμογής κρατήσεων χρησιμοποιώντας τις τεχνολογίες React και Node.js απέδειξε την αποτελεσματικότητα και την ευελιξία αυτών των εργαλείων στην κατασκευή σύγχρονων διαδικτυακών εφαρμογών. Τα συμπεράσματα από την υλοποίηση της συγκεκριμένης εφαρμογής περιλαμβάνουν:

Η χρήση της React για το Front-End επέτρεψε την κατασκευή μιας γρήγορης και διαδραστικής διεπαφής χρήστη. Η δυνατότητα δυναμικής ανανέωσης των δεδομένων χωρίς πλήρη επαναφόρτωση της σελίδας βελτίωσε σημαντικά την εμπειρία του χρήστη. Η δομή των component στη React διευκόλυνε την ανάπτυξη, την επαναχρησιμοποίηση κώδικα και τη συντήρηση της εφαρμογής. Η δυνατότητα δημιουργίας επαναχρησιμοποιήσιμων και modular components κατέστησε τον κώδικα πιο οργανωμένο και ευκολότερο στη διαχείριση.

Η Node.js, με τη χρήση των πακέτων της (NPM), επιτάχυνε την ανάπτυξη του Back-End. Η ευκολία στην εγκατάσταση και χρήση βιβλιοθηκών και εργαλείων μείωσε τον χρόνο ανάπτυξης. Επίσης, η χρήση προσαρμοσμένων κλάσεων σφαλμάτων στο Node.js, σε συνδυασμό με την ασύγχρονη διαχείριση αιτημάτων, βελτίωσε την ασφάλεια της εφαρμογής. Η διαχείριση σφαλμάτων έγινε πιο αποτελεσματική, επιτρέποντας την παροχή σαφών και κατανοητών μηνυμάτων στους χρήστες.

Όσον αφορά την αρχιτεκτονική της εφαρμογής, βασισμένη σε React και Node.js, παρείχε τη δυνατότητα εύκολης επέκτασης με νέες λειτουργίες και υπηρεσίες. Η ευελιξία αυτών των τεχνολογιών επιτρέπει την προσαρμογή της εφαρμογής στις μεταβαλλόμενες ανάγκες των ξενοδοχείων και των πελατών.

Συνολικά, η επιλογή των React και Node.js για την ανάπτυξη της εφαρμογής κρατήσεων αποδείχθηκε ιδιαίτερα ωφέλιμη, επιτρέποντας την κατασκευή μιας ευέλικτης, αποδοτικής και επεκτάσιμης εφαρμογής που ανταποκρίνεται στις σύγχρονες απαιτήσεις των χρηστών και των επιχειρήσεων.

Μετά από έρευνα και αξιολόγηση αντίστοιχων εφαρμογών επιλέξαμε η εφαρμογή μας να έχει αυτόν τον χαρακτήρα και να συνδυάζει αυτές τις λειτουργίες καθώς εντοπίσαμε κάποια βασικά πλεονεκτήματα σε σχέση με τις πλατφόρμες αναζήτησης σε διαφορετικά ξενοδοχεία. Κύριο πλεονέκτημα αποτελεί η εξάλειψη της πρακτικής των προμηθειών ανά κράτηση, που συνηθίζει να ανέρχεται στο 15%-20% στο ποσό κάθε κράτησης. Λειτουργώντας με ένα σύστημα συνδρομών λοιπόν η εφαρμογή αποσκοπεί στο να βοηθήσει μικρές επιχειρήσεις να γίνουν εξίσου γνωστές και ανταγωνιστικές και να μην επιβαρύνει τα ήδη περιορισμένα περιθώρια κέρδους τους. Επιπλέον, η δική μας πλατφόρμα δεν επιβάλλει συγκεκριμένους όρους και προϋποθέσεις δίνοντας έτσι στους ιδιοκτήτες ανεξαρτησία και ευελιξία, σχετικα με τους όρους χρήσης και τις πολιτικές κρατήσεων και ακυρώσεων. Ένα ακόμη θετικό χαρακτηριστικό, είναι η άμεση επαφή του χρήστη με τον υπεύθυνο εφόσον διαχειρίζεται κατευθείαν ο ίδιος τις κρατήσεις του ξενοδοχείου. Αυτό έχει ως αποτέλεσμα ο υπεύθυνος να έχει απευθειας πληροφορίες για τα ειδικά αιτήματα που προκύπτουν και του δίνει το πλεονέκτημα να έχει γνώση των προτιμήσεων πελατών του και να μπορεί όποτε κρίνει κατάλληλο να θέσει σε λειτουργία κάποια εξατομικευμένη υπηρεσία ή προσφορά.

Προσφέροντας μεγαλύτερο έλεγχο, ευελιξία και οικονομία, η εφαρμογή μας αποτελεί μια ισχυρή λύση για κάθε ξενοδόχο που επιθυμεί να βελτιώσει την εμπειρία των πελατών του και να αυξήσει τα έσοδά του. Η εφαρμογή μας, με την απευθείας ενσωμάτωσή της στην ιστοσελίδα του ξενοδοχείου, προσφέρει μια ολοκληρωμένη και αποτελεσματική προσέγγιση στη διαχείριση κρατήσεων, καθιστώντας την ένα απαραίτητο εργαλείο για την σύγχρονη ξενοδοχειακή βιομηχανία.

Ωστόσο, η συνεχής βελτίωση της εφαρμογής κρατήσεων είναι απαραίτητη για να διατηρείται επίκαιρη και να καλύπτει τις αυξανόμενες ανάγκες των χρηστών. Υπάρχουν διάφορες πτυχές που μπορούν να

βελτιωθούν.

Μια αρκετά ουσιώδης βελτίωση είναι η αυξημένη προσβασιμότητα, η οποία θα διασφαλίζει ότι άτομα με αναπηρίες μπορούν να χρησιμοποιούν εύκολα και αποτελεσματικά την εφαρμογή σύμφωνα με τις βέλτιστες πρακτικές προσβασιμότητας, όπως αυτές ορίζονται από τις Οδηγίες για την Προσβασιμότητα του Περιεχομένου Ιστού (Web Content Accessibility Guidelines - WCAG). Η εφαρμογή αυτών των βέλτιστων πρακτικών προσβασιμότητας θα διασφαλίσει ότι η εφαρμογή κρατήσεων είναι εύχρηστη και φιλική προς όλους τους χρήστες, ανεξάρτητα από τις ιδιαιτερότητες που μπορεί να τους χαρακτηρίζουν. Αυτό όχι μόνο βελτιώνει την εμπειρία χρήστη, αλλά επίσης ενισχύει την κοινωνική ευθύνη και τη συμμόρφωση με τους νομικούς κανονισμούς.

Ακόμη μια πρόταση που θα συμβάλει στην αναβάθμιση της εφαρμογής είναι ένα Σύστημα Ειδοποιήσεων το οποίο θα ενημερώνει τους χρήστες μέσω γραπτών μηνυμάτων και ηλεκτρονικού ταχυδρομείου για την επερχόμενη κράτηση τους ώστε να μπορούν να ειδοποιήσουν εγκαίρως για κάποια ενδεχόμενη μεταβολή ή ακύρωση της κράτησης τους.

Τέλος, η βελτίωση που θα μπορούσε να εξυπηρετεί εξίσου τους χρήστες αλλά και τους διαχειριστές της εφαρμογής είναι η ύπαρξη της λειτουργίας υποστήριξης πελατών για την διαχείριση αιτημάτων όλο το 24ωρο. Αυτό ενδεχομένως θα μπορούσε να επιτευχθεί με κάποια τεχνολογία Chat bot σε πρώτο στάδιο και μετέπειτα με την απασχόληση κάποιου υπαλλήλου για τα αιτήματα που κρίνεται απαραίτητο.

Οι παραπάνω προτάσεις μπορούν να συμβάλουν στη δημιουργία μιας ακόμα πιο ισχυρής, ευέλικτης και αποδοτικής εφαρμογής για τις ανάγκες των ξενοδοχείων και των πελατών τους.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Hotel Property Management Systems: Products and Features. [Online]. Διαθέσιμο μέσω: https://www.altexsoft.com/blog/hotel-property-management-systems-products-and-features/ [Πρόσβαση: 09/09/2024].

[2] Software Architecture Documentation in Practice: Documenting Architectural Layer. [Online]. Διαθέσιμο μέσω: https://cdn-prod-pdfsimpli-wpcontent.azureedge.net/pdfseoforms/pdf-20180219t134432z-001/pdf/software-architecture-document-2.pdf [Πρόσβαση: 09/09/2024].

[3] Guide to Web Application Architecture. [Online]. Διαθέσιμο μέσω:https://www.intellectsoft.net/blog/web-application-architecture/ [Πρόσβαση: 09/09/2024].

[4] Java SE Application Design With MVC. [Online].Διαθέσιμο μέσω:https://www.oracle.com/technical-resources/articles/javase/mvc.html [Πρόσβαση: 09/09/2024].

[5] Optimized Development of Web Front-end Development Technology. [Online]. Διαθέσιμο μέσω: https://iopscience.iop.org/article/10.1088/1742-6596/1693/1/012057/pdf [Πρόσβαση: 09/09/2024].

[6] HTML: HyperText Markup Language. [Online]. Διαθέσιμο μέσω: https://developer.mozilla.org/en-US/docs/Web/HTML [Πρόσβαση: 09/09/2024].

[7] HTML basics. [Online]. Διαθέσιμο μέσω: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics [Πρόσβαση: 09/09/2024].

[8] JavaScript® for Web Developers, 2nd Edition, Nicholas C. Zakas. [Online]. Διαθέσιμο μέσω: https://nuleren.be/edocumenten/professional-javascript-for-web-developers.pdf [Πρόσβαση: 09/09/2024].

[9] What is Vue?. [Online]. Διαθέσιμο μέσω: https://vuejs.org/guide/introduction [Πρόσβαση: 09/09/2024].

[10] 10 Things You Need To Know About the Vue.js Frontend Framework. [Online]. Διαθέσιμο μέσω: https://kinsta.com/blog/vue-js/ [Πρόσβαση: 09/09/2024].

[11] What are the Advantages of Vue js Framework in Web Development?. [Online]. Διαθέσιμο μέσω: https://www.monocubed.com/blog/advantages-of-vue-js/ [Πρόσβαση: 09/09/2024].

[12] The Good and the Bad of Vue.js Framework Programming. [Online]. Διαθέσιμο μέσω: https://www.altexsoft.com/blog/pros-and-cons-of-vue-js/ [Πρόσβαση: 09/09/2024].

[13] 10 Top Backend Technologies You Must Know [2024]. [Online]. Διαθέσιμο μέσω: https://herovired.com/learning-hub/blogs/back-end-technology/ [Πρόσβαση: 09/09/2024].

[14] Frontend vs Backend. [Online]. Διαθέσιμο μέσω: https://www.geeksforgeeks.org/frontend-vs-backend [Πρόσβαση: 09/09/2024].

[15] What is Java technology and why do I need it?. [Online]. Διαθέσιμο μέσω: https://www.java.com/en/download/help/whatis_java.html [Πρόσβαση: 09/09/2024].

[16] What is Java?. [Online]. Available: https://aws.amazon.com/what-is/java/ [Πρόσβαση: 09/09/2024].

[17] The Role of APIs in Digital Transformation. [Online]. Διαθέσιμο μέσω: https://www.linkedin.com/pulse/role-apis-digital-transformation-anbarasan-rathinam/ [Πρόσβαση: 09/09/2024].

[18] An Introduction to APIs . [Online]. Διαθέσιμο μέσω:

https://cdn.zapier.com/storage/learn_ebooks/e06a35cfcf092ec6dd22670383d9fd12.pdf
[Πρόσβαση: 09/09/2024].

[19] What's an API? Benefits, Examples, and Types. [Online]. Διαθέσιμο μέσω: https://www.coursera.org/articles/what-is-an- api? [Πρόσβαση: 09/09/2024].

[20] A Brief Guide to Everything You Need to Know About APIs. [Online]. Διαθέσιμο μέσω: https://hackernoon.com/a-brief-guide-to-everything-you-need-to-know-about-apis
[Πρόσβαση: 09/09/2024].

[21] Database Defined. [Online]. Διαθέσιμο μέσω: https://www.oracle.com/database/what-is-database/ [Πρόσβαση: 09/09/2024].

[22] What Is Structured Query Language (SQL)?. [Online]. Διαθέσιμο μέσω: https://www.oracle.com/database/what-is-database/ [Πρόσβαση: 09/09/2024].

[23] MySQL: Understanding What It Is and How It's Used. [Online]. Διαθέσιμο μέσω: https://www.oracle.com/mysql/what-is-mysql/ [Πρόσβαση: 09/09/2024].

[24] What is PostgreSQL?. [Online]. Διαθέσιμο μέσω: https://www.postgresql.org/about/ [Πρόσβαση: 09/09/2024].

[25] All About Oracle Database. [Online]. Διαθέσιμο μέσω: https://www.devart.com/dbforge/oracle/all-about-oracle-database/ [Πρόσβαση: 09/09/2024].

[26] What is Microsoft SQL Server? [Online]. Διαθέσιμο μέσω: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server [Πρόσβαση: 09/09/2024].

[27] What Is MongoDB?. [Online]. Διαθέσιμο μέσω: https://www.mongodb.com/company/what-is-mongodb [Πρόσβαση: 09/09/2024].

[28] What is CouchDB?. [Online]. Διαθέσιμο μέσω: https://www.ibm.com/topics/couchdb [Πρόσβαση: 09/09/2024].

[29] What is Apache Cassandra Registered?. [Online]. Διαθέσιμο μέσω: https://cassandra.apache.org/_/index.html [Πρόσβαση: 09/09/2024].

[30] Introduction to Redis. [Online]. Διαθέσιμο μέσω: https://redis.io/about/ [Πρόσβαση: 09/09/2024].

[31] Authentication and Authorization Mechanism for Cloud Security. [Online]. Διαθέσιμο μέσω: https://www.researchgate.net/profile/Dr-Chandra-Jadala/publication/335842661Authentication_and_Authorization_Mechanism_for_Cloud_Security/links/5d7fe326a6fdcc66b001a77f [Πρόσβαση: 09/09/2024].