



Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Ανάπτυξη Web Εφαρμογής με Θέμα τον Τουρισμό και Χρήση MERN Stack (MongoDB, Express, React, Node) Web Tourist Application Development using MERN Stack (MongoDB, Express, React, Node)
Όνοματεπώνυμο Φοιτητή	Ντάνυ Μαχζούμπ
Πατρώνυμο	Χαϊσάμ
Αριθμός Μητρώου	ΜΠΠΛ/16035
Επιβλέπων	Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης **Φεβρουάριος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)

(υπογραφή)

(υπογραφή)

Ευθύμιος Αλέπης
Αναπληρωτής Καθηγητής

Μαρία Βίρβου
Καθηγήτρια

Κωνσταντίνος Πατσάκης
Αναπληρωτής Καθηγητής

Περίληψη/Abstract

Ελληνικά

Θέμα της Μεταπτυχιακής Διατριβής είναι η δημιουργία μιας ιστοσελίδας με την ονομασία **TripSpot**, που αφορά τον τουρισμό. Σκοπός της ιστοσελίδας είναι οι χρήστες να μπορούν να ανακαλύψουν ή να προτείνουν διάφορα τουριστικά μέρη του κόσμου. Επιπρόσθετα, μπορούν να διαβάσουν και να γράψουν κριτικές για αυτά και να αλληλεπιδρούν μέσω μηνυμάτων μεταξύ των υπόλοιπων χρηστών. Όλα τα παραπάνω, κάνουν πιο ευχάριστη την εμπειρία του χρήστη στην ιστοσελίδα και τον βοηθούν να μπορέσει να οργανώσει ένα ταξίδι του. Η τεχνολογία που χρησιμοποιήθηκε για την δημιουργία της ιστοσελίδας είναι η MERN Stack (MongoDB, Express, React, Node).

English

The subject of this Master Thesis is the creation of a website, named TripSpot which is about tourism. The main goal of the website is that the users can discover or suggest a wide variety of tourist places all around the world. Furthermore, the user could read and write comments about the places and also they can interact with each other using messages. With these features, the experience of the users becomes more pleasant and helps them to organise an upcoming trip. The technology that was used for creating the website is MERN Stack (MongoDB, Express, React, Node).

Εισαγωγή

Η ιστοσελίδα έχει την ονομασία **TripSpot** και το σκεπτικό δημιουργίας της βασίζεται στην ανάγκη των χρηστών να οργανώσουν ένα επερχόμενο ταξίδι τους αλλά και να βοηθήσουν άλλους χρήστες που έχουν την ίδια ανάγκη. Ανάλογα τις προτιμήσεις τους μπορούν να αναζητήσουν διάφορες κατηγορίες τουριστικών προορισμών όπως για παράδειγμα παραλίες, μνημεία, μουσεία κ.ά. Επιπλέον, μέσω του διαδραστικού χάρτη που περιέχει η συγκεκριμένη ιστοσελίδα ο χρήστης μπορεί να βρει ένα μέρος τον οποίο τον ενδιαφέρει μέσω χιλιομετρικής απόστασης ή με την προσθήκη της χώρα/πόλης που θέλει να επισκεφτεί. Αυτό δίνει ένα μεγάλο προτέρημα στην εφαρμογή διότι προσαρμόζεται πλήρως με τις ανάγκες του χρήστη.

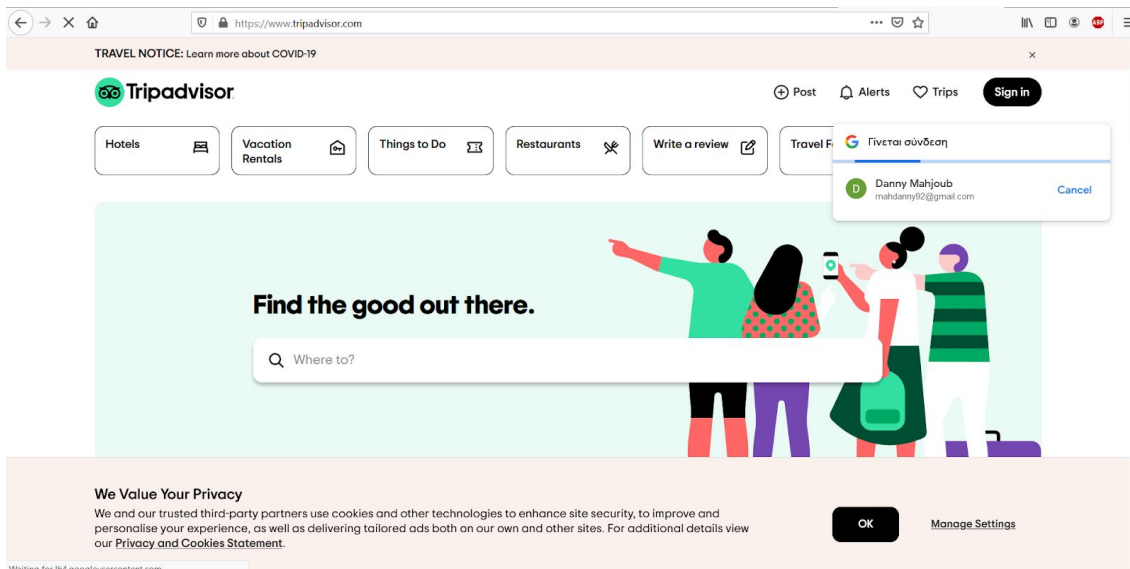
Μία ακόμη λειτουργία που κάνει την ιστοσελίδα να ξεχωρίζει είναι πως ο χρήστης μπορεί να προτείνει κάποιο μέρος το οποίο δεν είναι καταχωρημένο. Με αυτόν τον τρόπο η βάση της ιστοσελίδας μεγαλώνει διαρκώς με την προσθήκη νέων τοποθεσιών και κρατάει συνεχώς το ενδιαφέρον των χρηστών. Όμως, για την εγκυρότητα των νέων τοποθεσιών πρέπει ένας διαχειριστής να επεξεργαστεί και να αποδεχτεί την κάθε νέα πρόταση, πράγμα το οποίο κάνει ακόμη πιο ασφαλή την εικόνα των δεδομένων.

Επιπρόσθετα, οι χρήστες μπορούν να αλληλεπιδρούν μεταξύ τους όχι μόνο μέσω των προσωπικών μηνυμάτων, αλλά και μέσω της προσθήκης σχολίων (comments) περιγράφοντας την εμπειρία τους για μια συγκεκριμένη τοποθεσία, ανεβάζοντας νέες φωτογραφίες και δίνοντας της μια βαθμολογία (stars rating).

Επιπλέον, με την λειτουργία των αγαπημένων (favorites) οι χρήστες μπορούν να προσθέτουν ή να διαγράφουν τοποθεσίες που πλέον δεν τους ενδιαφέρουν από την λίστα τους. Τέλος, μπορούν να αναζητήσουν ποιες τοποθεσίες τους ή σχόλια τους έχουν γίνει αποδεκτά από τους διαχειριστές της σελίδας ώστε να έχουν εικόνα για την δραστηριότητα τους στην ιστοσελίδα.

Παρόμοιες Εφαρμογές

Tripadvisor



Η ιστοσελίδα που έχει τα πιο πολλά κοινά χαρακτηριστικά με την ιστοσελίδα που δημιουργήθηκε για την αυτή την Μεταπτυχιακή Διατριβή είναι η www.tripadvisor.com. Το κυριότερο κοινό τους στοιχείο είναι πως και οι δυο ιστότοποι προσφέρουν την ευκαιρία στους χρήστες να ανακαλύψουν τοποθεσίες που τους ενδιαφέρουν και να οργανώσουν ένα ταξίδι τους. Επίσης, και στις δύο σελίδες οι χρήστες μπορούν να σχολιάσουν τοποθεσίες, να ανεβάσουν φωτογραφίες και να τις βαθμολογήσουν.

Από την άλλη πλευρά, το Tripadvisor έχει μια μεγαλύτερη γκάμα με μέρη τα οποία μπορείς να αναζητήσει κάποιος. Για παράδειγμα, ένας χρήστης μπορεί να ψάξει από ένα ξενοδοχείο μέχρι και μια αεροπορική εταιρία. Το TripSpot όμως επικεντρώνεται στα αξιοθέατα που μπορεί να ανακαλύψει ένας χρήστης και αυτό το καθιστά πιο συγκεκριμένο στις ανάγκες που θέλει να εξυπηρετήσει. Το Tripadvisor έχει μια πιο διαφημιστική προσέγγιση προς τους επισκέπτες του.

Επιπλέον, στην δική μας ιστοσελίδα οι χρήστες έχουν την επιλογή να στείλουν μεταξύ τους μηνύματα έτσι ώστε να ανταλλάζουν πιο άμεσα απόψεις και κριτικές για τα μέρη που θέλουν να επισκεφθούν. Ακόμα, οι χρήστες μπορούν να προτείνουν συνεχώς καινούργιες τοποθεσίες, κάτι το οποίο για να το κάνει κάποιος στο Tripadvisor θα πρέπει να δώσει ένα χρηματικό αντίκτυπο διότι θεωρείται διαφήμιση προς την προτεινόμενη τοποθεσία.

Εν κατακλείδι, μπορεί αυτοί οι δυο ιστότοποι να μοιάζουν αρκετά μεταξύ τους, αλλά προσεγγίζουν τελείως διαφορετικά τους χρήστες. Οπότε, η κάθε ιστοσελίδα εξυπηρετεί διαφορετικές ανάγκες άρα ανάλογα με τον σκοπό που έχει ένας χρήστης θα επισκεφτεί και την αντίστοιχη ιστοσελίδα.

Παρουσίαση και χρήση της ιστοσελίδας (User's Manual)

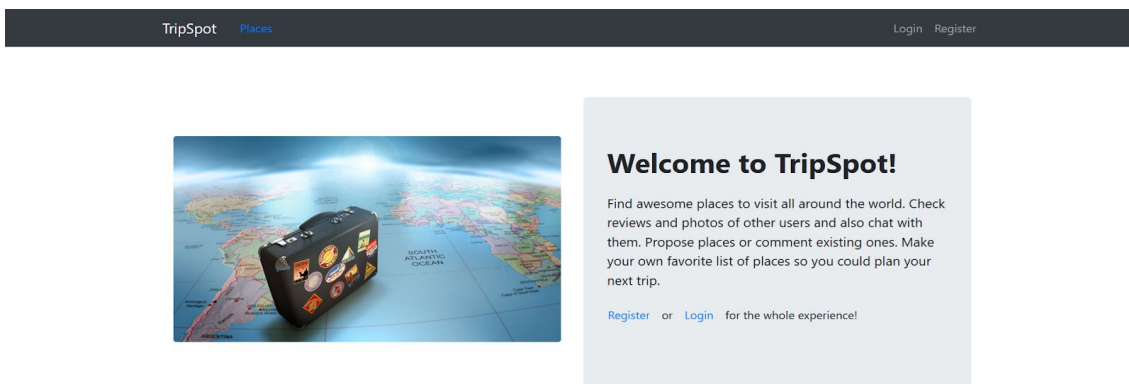
Η ιστοσελίδα προσφέρει τρία είδη διαφορετικών χρηστών. Τον απλό επισκέπτη, τον συνδεδεμένο χρήστη και τους διαχειριστές. Παρακάτω θα παρουσιαστούν οι λειτουργίες του ιστότοπου μέσω screenshots και για ευκολία η παρουσίαση θα γίνει σε τρεις ενότητες χωρισμένες ανάλογα με το είδος του χρήστη. Επιπλέον, είναι σημαντικό να αναφερθεί ότι οπουδήποτε υπάρχουν λίστες με δεδομένα υπάρχει σελιδοποίηση (pagination) ώστε τα δεδομένα να είναι πιο ταξινομημένα και να είναι πιο εύκολη η χρήση της ιστοσελίδας. Τέλος, οπουδήποτε γίνεται κάποια λάθος εισαγωγή από τον χρήστη ή προσπαθήσει να κάνει είσοδο κάπου που δεν έχει άδεια γίνεται η εμφάνιση των ανάλογων μηνυμάτων/errors.

Απλός επισκέπτης (Visitor)

Εφόσον ο χρήστης δεν έχει λογαριασμό ή δεν είναι συνδεδεμένος από πριν τότε του εμφανίζονται οι βασικές λειτουργίες της ιστοσελίδας.

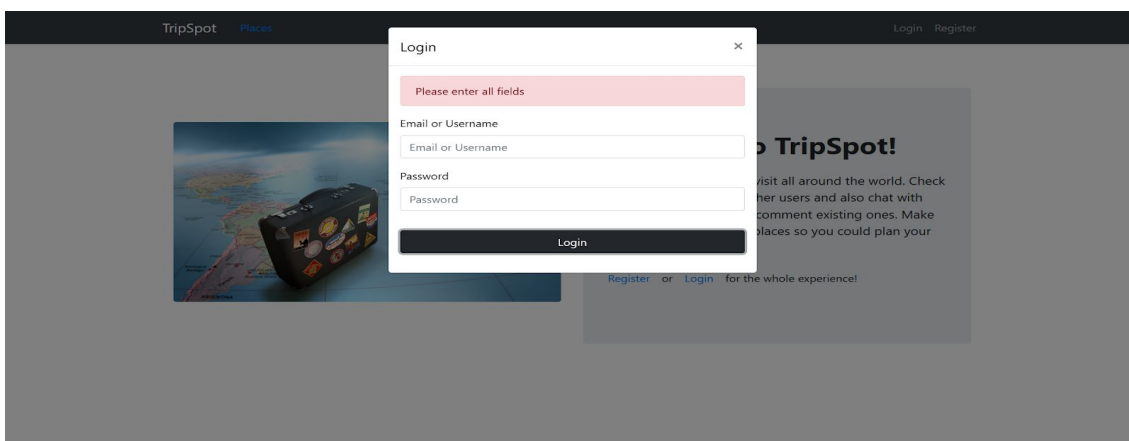
Αρχική Σελίδα (Home Page)

Η εμφάνιση της αρχικής σελίδας, η οποία παροτρύνει τον επισκέπτη να συνδεθεί ή να κάνει εγγραφή.



Είσοδος Χρήστη (Login)

Είσοδος του χρήστη με την εμφάνιση ανάλογων μηνυμάτων σε περίπτωση κάποιου λάθους(χρήστης που δεν υπάρχει, άλλος κωδικός κ.α.)





Ανάπτυξη Web Εφαρμογής με Θέμα τον Τουρισμό και Χρήση MERN Stack (MongoDB, Express, React, Node)

Εγγραφή Χρήστη (Register)

Εγγραφή του χρήστη με την εμφάνιση ανάλογων μηνυμάτων σε περίπτωση κάποιου λάθους (ήδη υπαρχών χρήστης ή email, λάθος τύπος κωδικού κ.α.)

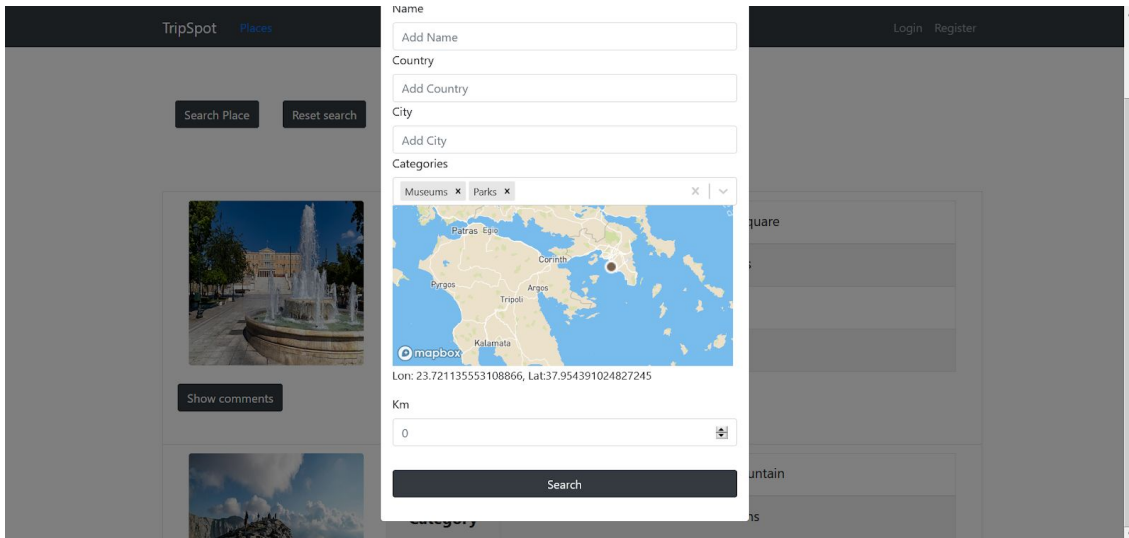
Τοποθεσίες (Places)

Εδώ ο χρήστης μπορεί να δει τις τοποθεσίες που υπάρχουν, να αναζητήσει συγκεκριμένες μέσω του κουμπιού Search, να πατήσει το κουμπί επαναφοράς Reset για να επανέλθουν στην αρχική κατάσταση τα κριτήρια αναζήτησης και να επιλέξει μια τοποθεσία ώστε να μπορέσει να δει τα σχόλια που έχουν κάνει άλλοι χρήστες για αυτή.

	Name	Syntagma Square
	Category	Squares
	Country	Greece
	City	Athens
Show comments		
	Name	Olympus Mountain
	Category	Mountains

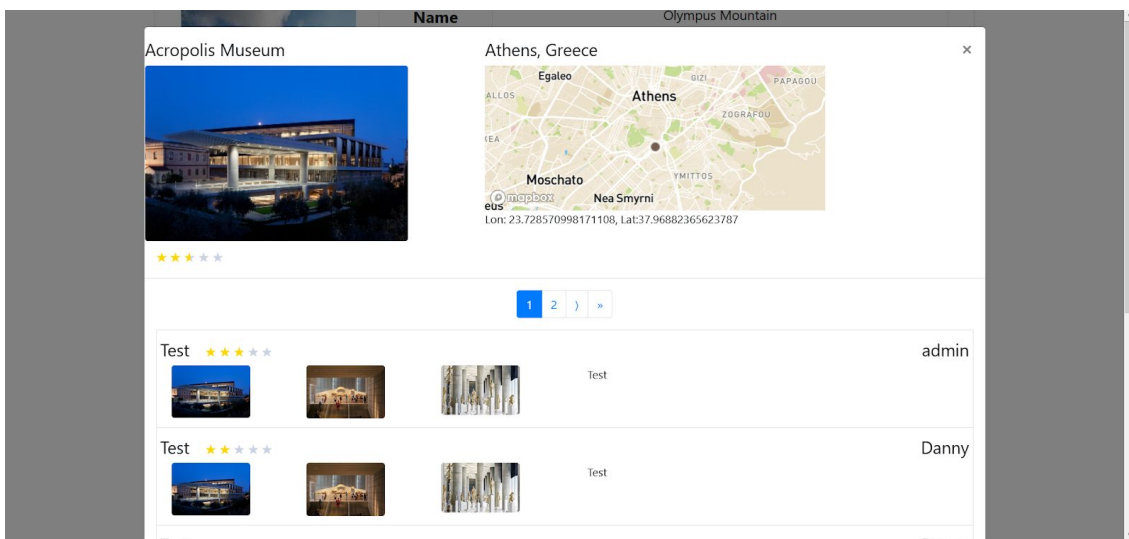
Κουμπί Αναζήτησης (Search)

Όταν ο χρήστης πατήσει το κουμπί αναζήτησης μπορεί να αναζητήσει τοποθεσίες βάσει συγκεκριμένων κριτηρίων όπως το όνομα ενός μέρους, την χώρα ή την πόλη που βρίσκεται. Ακόμα, μπορεί να διαλέξει πολλαπλές επιλογές κατηγοριών των τοποθεσιών. Και τέλος, μπορεί να σύρει οπουδήποτε επιθυμεί την κουκίδα στο χάρτη και να επιλέξει την ακτίνα χιλιομέτρων που θέλει να του εμφανιστούν τοποθεσίες.



Εμφάνιση Λεπτομερειών και Σχολίων Τοποθεσίας (Comments)

Όταν ο χρήστης θα πατήσει το κουμπί Show comments, μπορεί πλέον να δει παραπάνω πληροφορίες για μια τοποθεσία όπως που βρίσκεται ακριβώς στον χάρτη. Επιπλέον, μπορεί να δει τα σχόλια που έχουν ανεβάσει άλλοι χρήστες (φωτογραφίες, περιγραφή κριτικής, βαθμολογία και το όνομα χρήστη που έκανε την κριτική). Εδώ να τονίσουμε ότι επειδή ο χρήστης είναι ένας απλός επισκέπτης δεν έχει την δυνατότητα να κάνει comment ή να προσθέσει την περιοχή στην λίστα με τα αγαπημένα του.

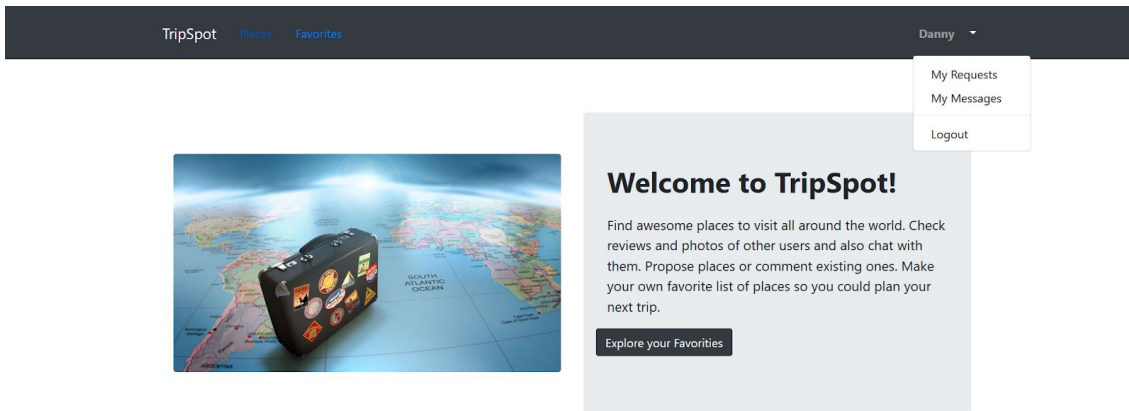


Συνδεδεμένος χρήστης (Register User)

Εάν ο χρήστης κάνει επιτυχή είσοδο στην σελίδα πλέον έχει το δικαίωμα να χρησιμοποιήσει παραπάνω λειτουργίες της ιστοσελίδας.

Αρχική Σελίδα (Home Page)

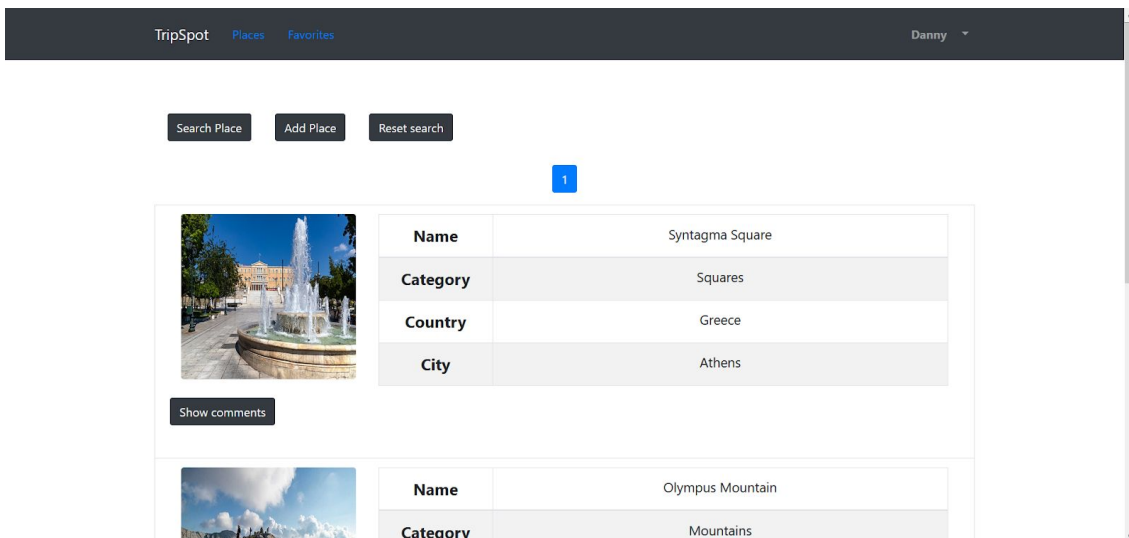
Η διαφορά της αρχικής σελίδας του συνδεδεμένου χρήστη είναι ότι τον παροτρύνει να εξερευνήσει τις αγαπημένες του τοποθεσίες. Επιπρόσθετα, στην θέση του Register/Login in εμφανίζεται το όνομα του και αν κλικάρει σε αυτό του εμφανίζονται οι επιλογές να δει τα requests του, τα μηνύματα του ή να κάνει αποσύνδεση.



localhost:3000/Places

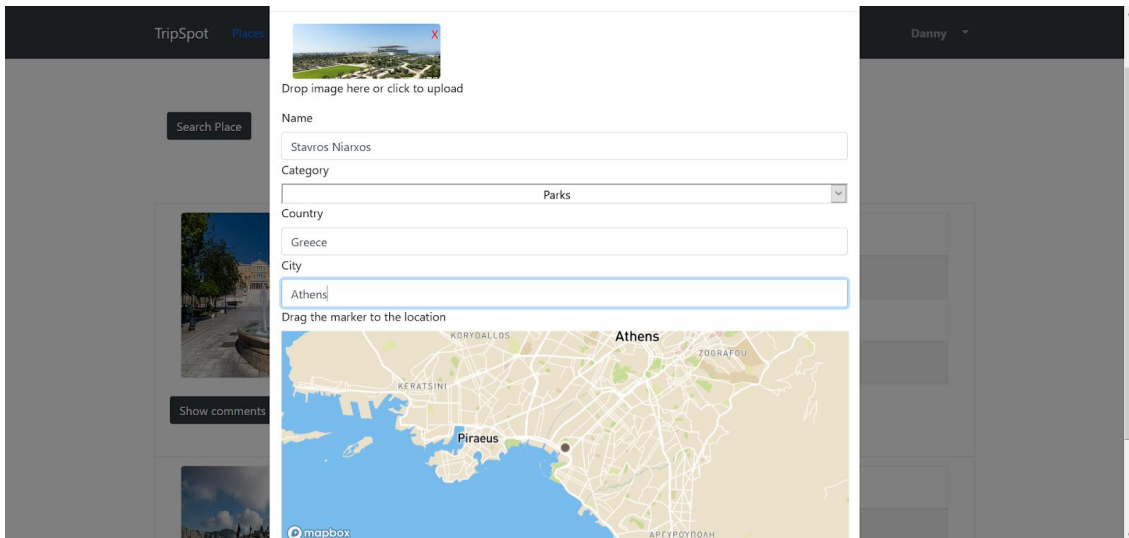
Τοποθεσίες (Places)

Πλέον, υπάρχει το κουμπί προσθήκης τοποθεσίας.



Κουμπί Προσθήκης Τοποθεσίας (Add Place)

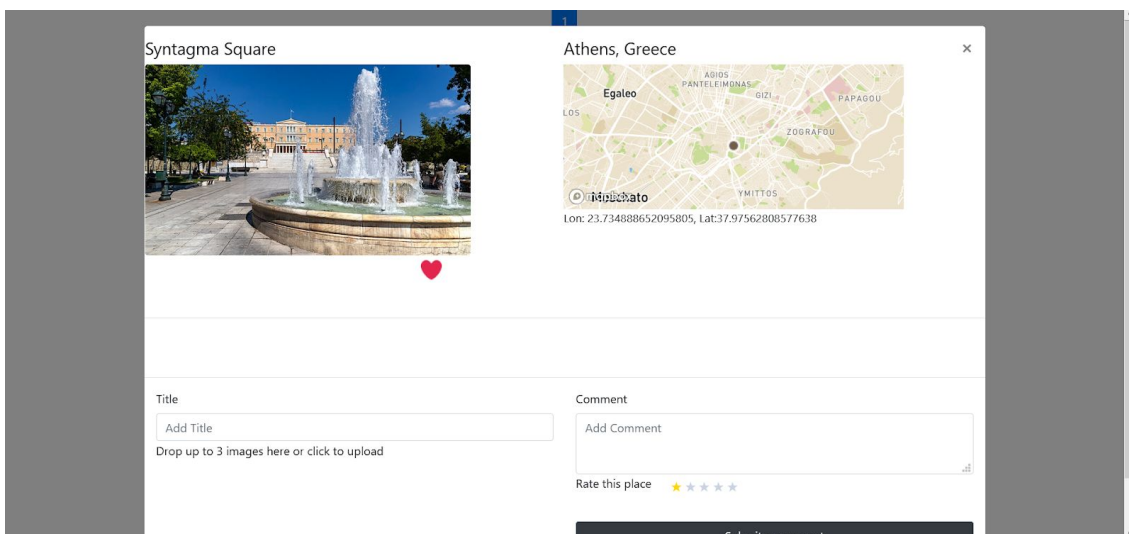
Ο χρήστης μπορεί να προτείνει μια τοποθεσία που στην συνέχεια ένας χρήστης admin θα την εγκρίνει. Πρέπει να ανεβάσει μια φωτογραφία του μέρους, να προσθέσει το όνομα του την χώρα και την πόλη που βρίσκεται αλλά και να σύρει την κουκίδα του χάρτη που βρίσκεται ακριβώς η τοποθεσία. Τέλος, γίνεται εμφάνιση ανάλογων μηνυμάτων σε περίπτωση κάποιου λάθους (μη προσθήκη φωτογραφίας, μέρος που υπάρχει ήδη κ.α.)



Εμφάνιση Λεπτομερειών και Σχολίων Τοποθεσίας (Comments)

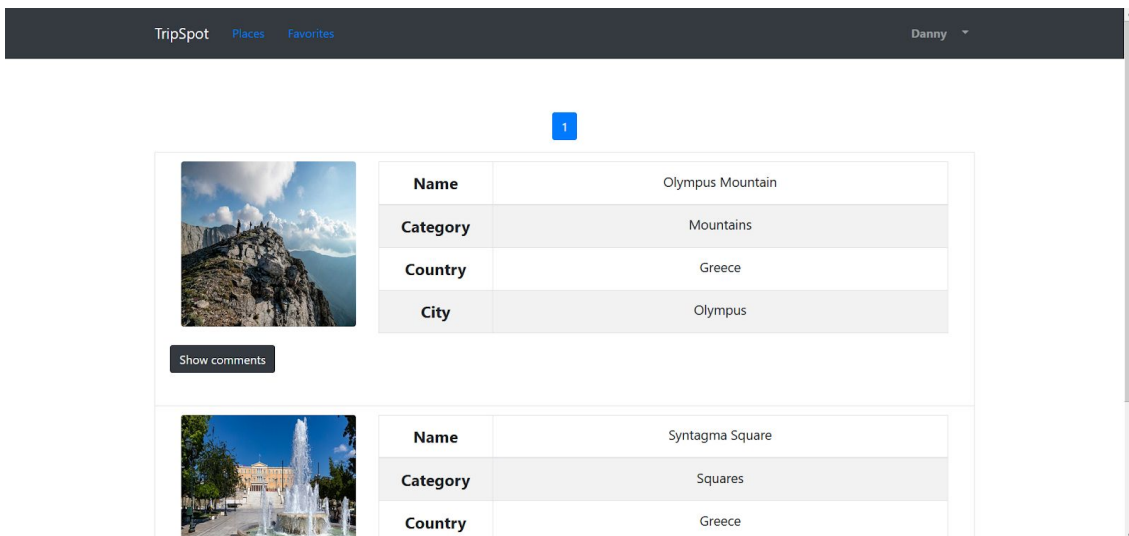
Ο συνδεδεμένος χρήστης μπορεί μέσω το εικονιδίου της καρδούλας να προσθέσει ή να αφαιρέσει μια τοποθεσία από την λίστα με τα αγαπημένα του.

Επιπρόσθετα, μπορεί να κάνει σχόλιο για την συγκεκριμένη τοποθεσία που στην συνέχεια ένας χρήστης admin θα την εγκρίνει. Πρέπει να ανεβάσει έως τρεις φωτογραφίες, να βάλει τίτλο, να περιγράψει την κριτική του και να βαθμολογήσει με αστεράκια. Τέλος, γίνεται εμφάνιση ανάλογων μηνυμάτων σε περίπτωση κάποιου λάθους (μη προσθήκη φωτογραφίας ή τίτλου κ.α.)



Αγαπημένα (Favorites)

Εδώ ο χρήστης μπορεί να περιηγηθεί και να επεξεργαστεί την λίστα με τις αγαπημένες του τοποθεσίες.



The screenshot shows the 'Favorites' section of the TripSpot application. At the top, there is a navigation bar with 'TripSpot', 'Places', and 'Favorites' links, and a user profile 'Danny'. Below the navigation, there is a blue box with the number '1'. The main content area displays two favorite locations:

- Olympus Mountain:** Includes a thumbnail image of a mountain peak, a 'Show comments' button, and a table with the following details:

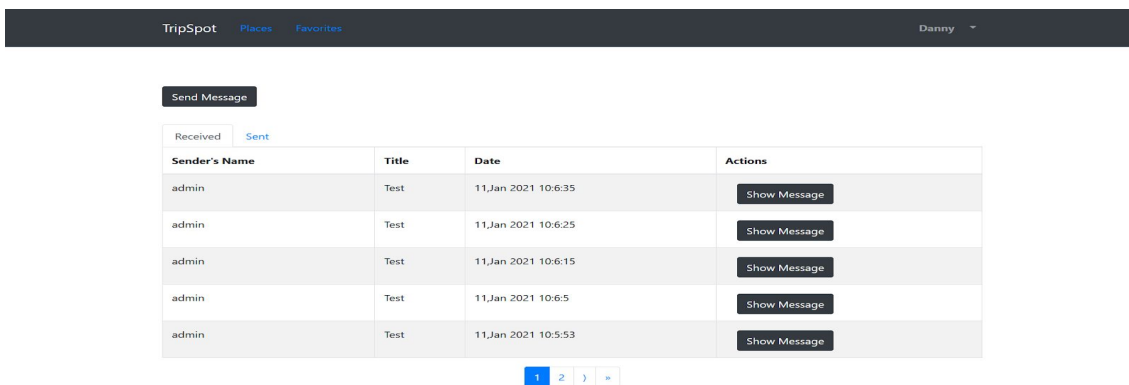
Name	Olympus Mountain
Category	Mountains
Country	Greece
City	Olympus
- Syntagma Square:** Includes a thumbnail image of a square with a fountain, and a table with the following details:

Name	Syntagma Square
Category	Squares
Country	Greece

Προσωπικά Μηνύματα(My Messages)

Ο χρήστης μπορεί να δει μια λίστα με τα μηνύματα που έχει στείλει και έχει λάβει με χρονολογική σειρά. Ακόμα, μπορεί να στείλει και προσωπικά μηνύματα.

Τα εισερχόμενα (Received)

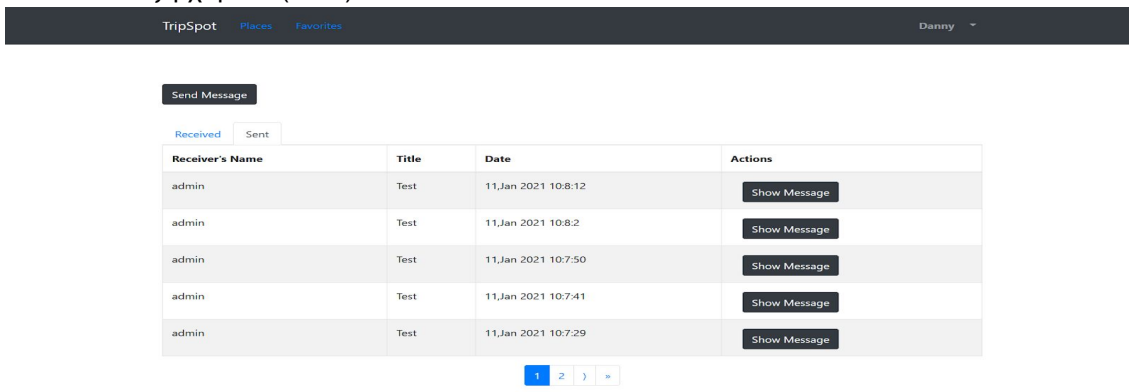


The screenshot shows the 'Received' messages section. At the top, there is a 'Send Message' button and tabs for 'Received' and 'Sent'. Below the tabs is a table with the following data:

Sender's Name	Title	Date	Actions
admin	Test	11,Jan 2021 10:6:35	Show Message
admin	Test	11,Jan 2021 10:6:25	Show Message
admin	Test	11,Jan 2021 10:6:15	Show Message
admin	Test	11,Jan 2021 10:6:5	Show Message
admin	Test	11,Jan 2021 10:5:53	Show Message

At the bottom of the table, there is a pagination control showing '1', '2', and navigation arrows.

Τα εξερχόμενα (Sent)

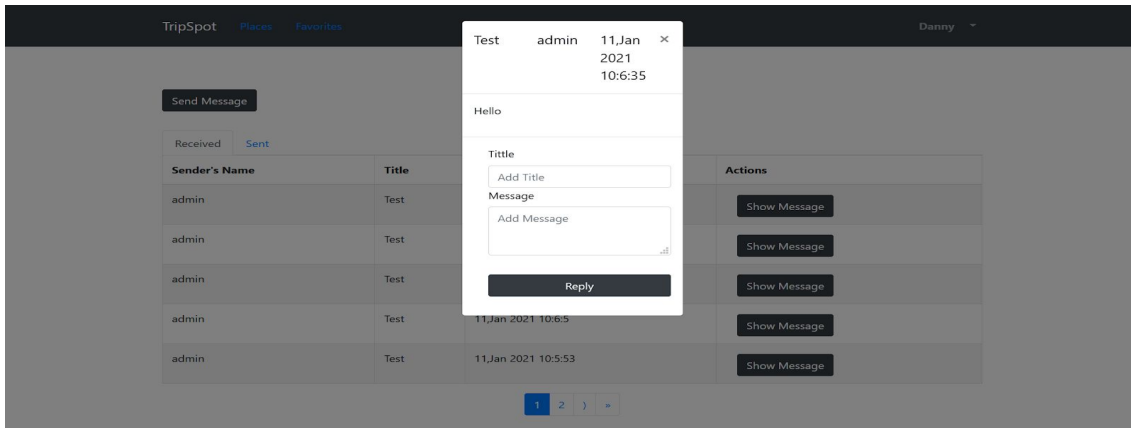


The screenshot shows the 'Sent' messages section. At the top, there is a 'Send Message' button and tabs for 'Received' and 'Sent'. Below the tabs is a table with the following data:

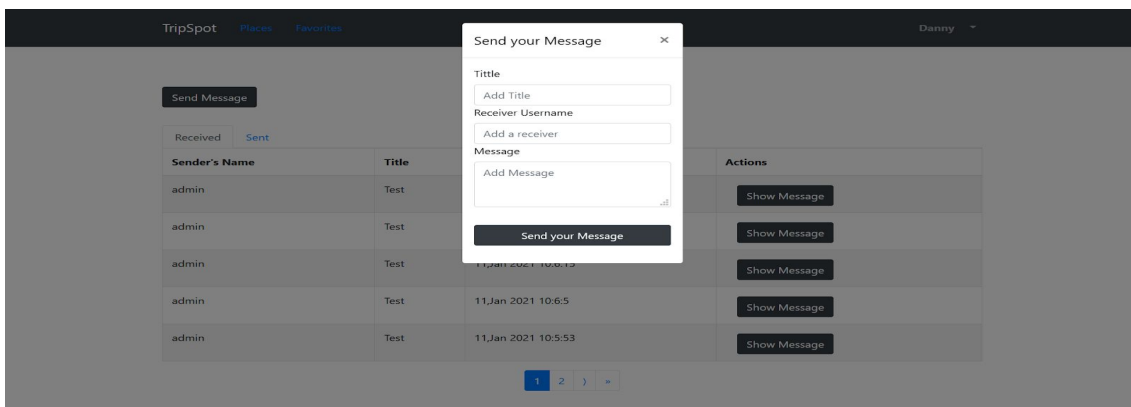
Receiver's Name	Title	Date	Actions
admin	Test	11,Jan 2021 10:8:12	Show Message
admin	Test	11,Jan 2021 10:8:2	Show Message
admin	Test	11,Jan 2021 10:7:50	Show Message
admin	Test	11,Jan 2021 10:7:41	Show Message
admin	Test	11,Jan 2021 10:7:29	Show Message

At the bottom of the table, there is a pagination control showing '1', '2', and navigation arrows.

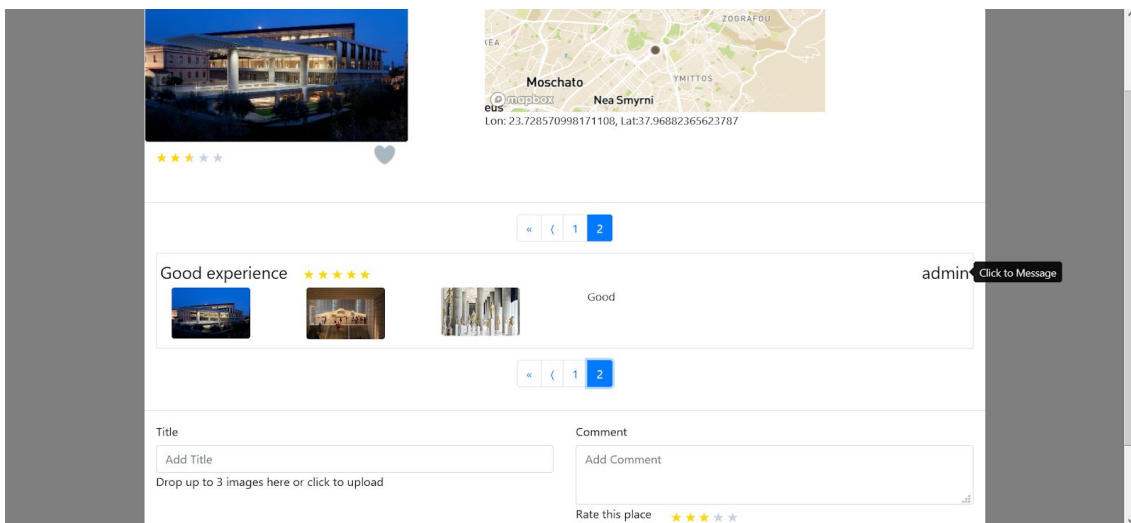
Το κουμπί εμφάνιση μηνύματος (Show Message). Σε περίπτωση που είμαστε στα εισερχόμενα μας εμφανίζει την επιλογή απάντησης (Reply), στα εξερχόμενα δεν υπάρχει αυτή η επιλογή.



Το κουμπί αποστολή μηνύματος (Send Message)

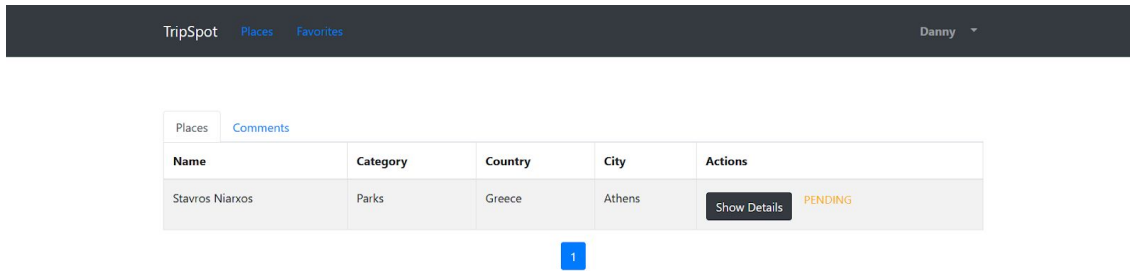


Εάν, ο χρήστης είναι στην λειτουργία που εμφανίζονται οι κριτικές και πάει πάνω από ένα όνομα χρήστη που έχει κάνει ένα σχόλιο θα του εμφανιστεί η επιλογή ότι αν κλικάρει το όνομα του άλλου χρήστη μπορεί να του στείλει προσωπικό μήνυμα και θα του εμφανιστεί η αποστολή μηνύματος με το αντίστοιχο όνομα του παραλήπτη ήδη συμπληρωμένο.



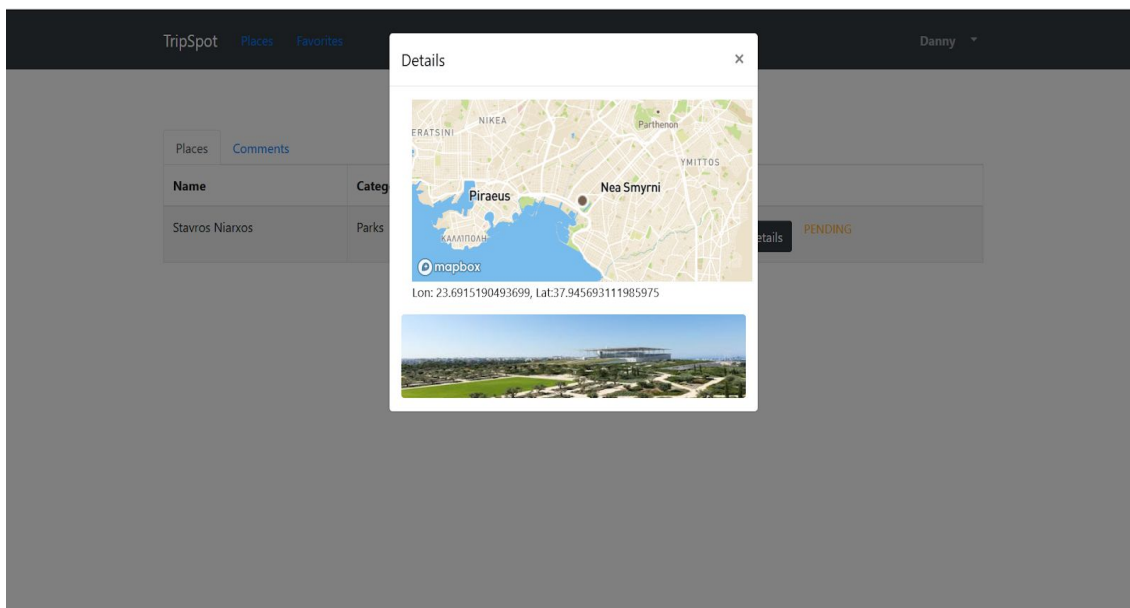
Οι Αιτήσεις μου (My requests)

Ο χρήστης μπορεί να δει ποιές προτάσεις του για καινούργιες τοποθεσίες ή ποιές κριτικές του έχουν γίνει αποδεκτές και ποιές ακόμα εκκρεμούν. Παρακάτω φαίνεται μια πρόταση για τοποθεσία ή οποία ακόμα εκκρεμεί αποδοχής από τον διαχειριστή.

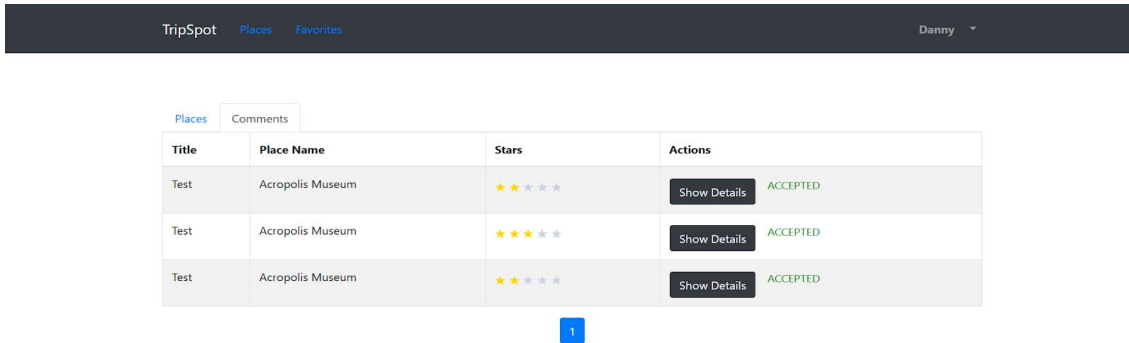


Name	Category	Country	City	Actions
Stavros Niarxos	Parks	Greece	Athens	Show Details PENDING

Με το κουμπί των λεπτομερειών (Show Details) μας εμφανίζεται η φωτογραφία και ο χάρτης για κάθε πρόταση τοποθεσίας.



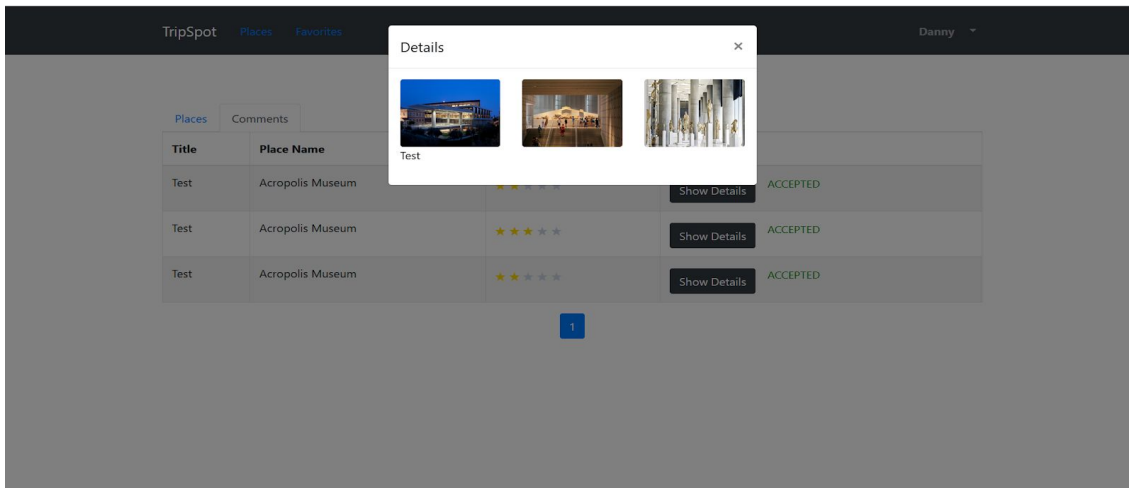
Παρακάτω φαίνονται οι κριτικές που έχει κάνει ο χρήστης και έχουν γίνει αποδεκτές από τον διαχειριστή. Δεν εκκρεμεί καμία κριτική προς το παρόν προς αποδοχή.



The screenshot shows the TripSpot application interface. At the top, there is a navigation bar with 'TripSpot', 'Places', and 'Favorites' links, and a user profile 'Danny'. Below the navigation bar, there are two tabs: 'Places' and 'Comments'. The 'Comments' tab is active, displaying a table of reviews. The table has four columns: 'Title', 'Place Name', 'Stars', and 'Actions'. There are three rows of reviews, all for 'Acropolis Museum' with a rating of 4 stars. Each row has a 'Show Details' button and a green 'ACCEPTED' status. Below the table, there is a blue button with the number '1'.

Title	Place Name	Stars	Actions
Test	Acropolis Museum	★★★★☆	Show Details ACCEPTED
Test	Acropolis Museum	★★★★☆	Show Details ACCEPTED
Test	Acropolis Museum	★★★★☆	Show Details ACCEPTED

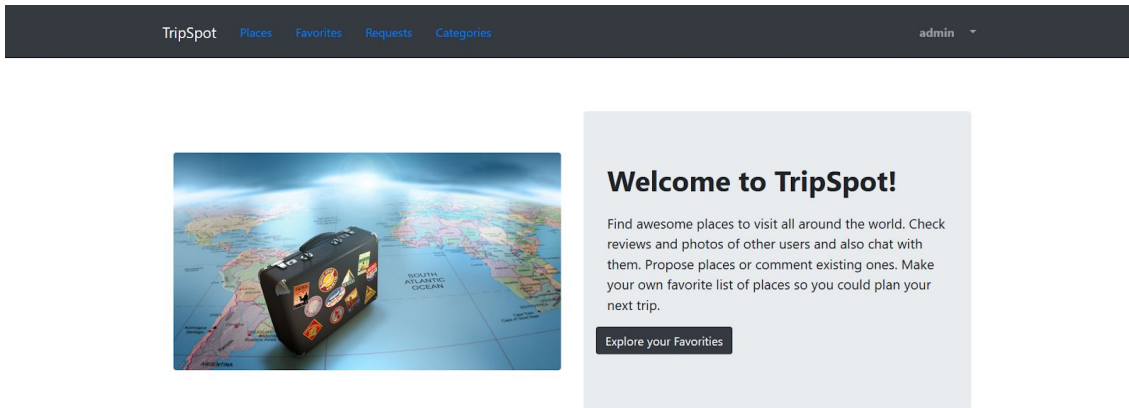
Με το κουμπί των λεπτομερειών (Show Details) μας εμφανίζονται οι φωτογραφίες και η περιγραφή της επιλεγμένης κριτικής.



Χρήστης Διαχειριστής (Admin User)

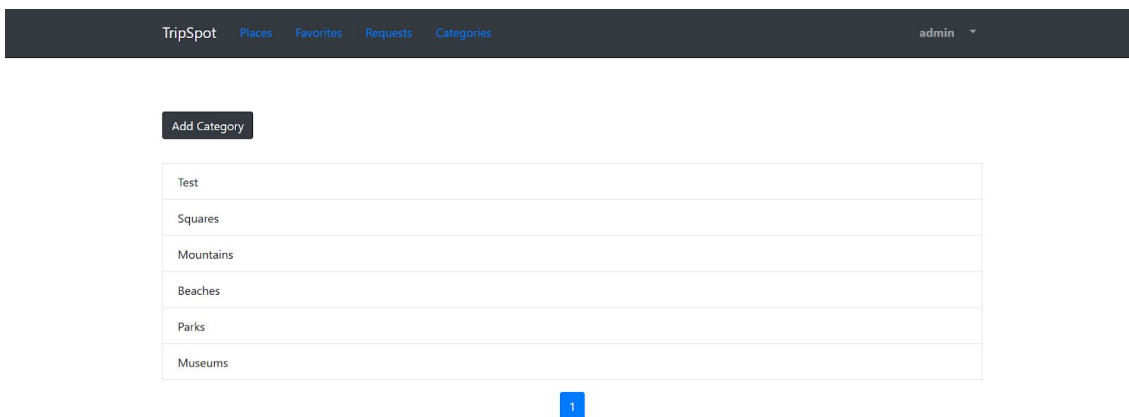
Για να γίνει ένας χρήστης διαχειριστής πρέπει να του δοθεί το δικαίωμα από κάποιον που έχει πρόσβαση στην βάση δεδομένων. Από την στιγμή που γίνεται admin έχει όλες τις δυνατότητες ενός συνδεδεμένου χρήστη αλλά και κάποιες επιπλέον. Παρακάτω θα παρουσιαστούν μόνο οι επιπλέον λειτουργίες καθώς όπως προαναφέρθηκε οι υπόλοιπες είναι ακριβώς ίδιες με του συνδεδεμένου χρήστη.

Εδώ φαίνεται στην μπάρα περιήγησης ότι έχουν προστεθεί τα αιτήματα (Requests) και οι κατηγορίες (Categories).

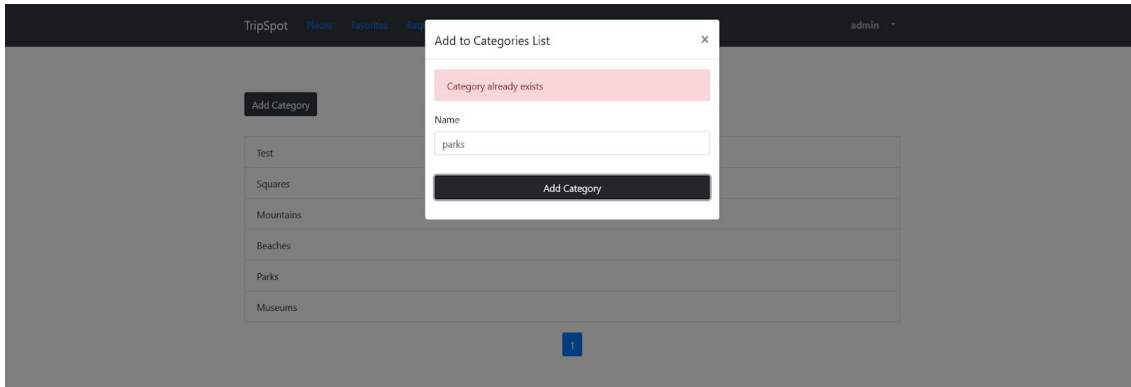


Κατηγορίες (Categories)

Ο διαχειριστής βλέπει όλες τις κατηγορίες τοποθεσιών που υπάρχουν.



Επιπλέον μπορεί να προσθέσει καινούργιες. Αν κάποια κατηγορία υπάρχει θα του εμφανιστεί το αντίστοιχο μήνυμα.



Αιτήματα (Requests)

Ο διαχειριστής μπορεί να δει τοποθεσίες που έχουν προτείνει χρήστες και σχόλια τα οποία έχουν κάνει, αλλά που ακόμα δεν τα έχει αποδεχτεί ή απορρίψει. Με το κουμπί των λεπτομερειών (Show Details) εμφανίζονται οι ανάλογες λεπτομέρειες. Με το κουμπί αποδοχή (Accept) εγκρίνει μια τοποθεσία ή ένα σχόλιο και με το κουμπί διαγραφής (Delete) τα απορρίπτει. Και στις δύο περιπτώσεις η τοποθεσία ή το σχόλιο εξαφανίζονται μετά από την λίστα των αιτημάτων και εμφανίζονται τα ανάλογα μηνύματα στον χρήστη που τα έχει κάνει.



Places		Comments		
Name	Category	Country	City	Actions
Stavros Niarxos	Parks	Greece	Athens	<a>Show Details <a>Accept <a>Delete

1



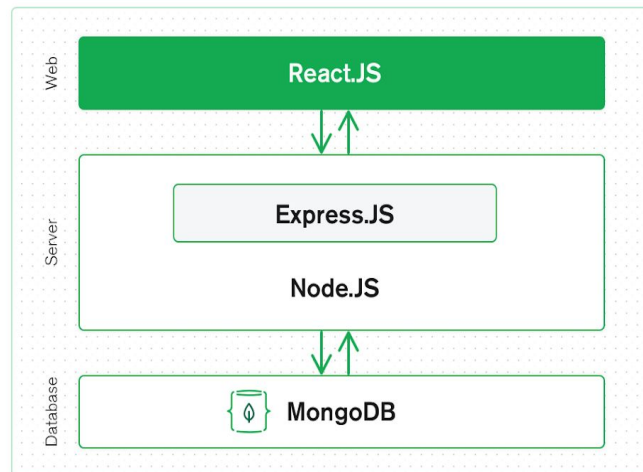
Places		Comments		
Title	Place Name	Stars	Actions	
Test	Acropolis Museum	★ ★ ★ ★ ★	<a>Show Details <a>Accept <a>Delete	

1

Αρχιτεκτονική Συστήματος

Η τεχνολογία που χρησιμοποιήθηκε για την δημιουργία της ιστοσελίδας είναι η MERN Stack (MongoDB, Express(.js), React(.js), Node(.js)). MongoDB είναι μια document NoSQL βάση δεδομένων, Express είναι ένα Node.js web framework, React είναι ένα client-side JavaScript framework και Node είναι ο βασικός JavaScript web server.

Με λίγα λόγια αυτή η τεχνολογία μας επιτρέπει να δημιουργήσουμε μια ιστοσελίδα με αρχιτεκτονική τριών επιπέδων (frontend, backend, database) χρησιμοποιώντας μόνο JavaScript και JSON.



React.JS Front End

Η React είναι το JavaScript framework που δημιουργεί δυναμικές client-side εφαρμογές σε HTML. Με την React δημιουργήθηκε όλο το frontend μέσω απλών στοιχείων (Components), τα οποία συνδέονται με τα δεδομένα που υπάρχουν στον backend server και μετά αυτά εμφανίζονται ως HTML.

Τα Components μέσω του Redux store τους επιτρέπεται να διαβάζουν δεδομένα και στέλνουν ενέργειες (Actions) στο store ώστε να αναβαθμίσουν τα δεδομένα αυτά. Για κάθε collection που έχουμε στην βάση δεδομένων μας έχουμε και το ανάλογο actions.js αρχείο το οποίο ορίζει τα actions του κάθε collection και το ανάλογο reducer.js αρχείο το οποίο συνδέει το frontend με το backend και ορίζει την κατάσταση που βρίσκεται το frontend όταν εκτελείται ένα action.

Ένα ενδεικτικό παράδειγμα για το πως συνδέονται τα Components με το Redux.

```
CategoriesList.js  ↳ ×
Thesis JavaScript Content Files  -  {} "CategoriesList"
73
74  CategoriesList.propTypes = {
75    getCategories: PropTypes.func.isRequired,
76    category: PropTypes.object.isRequired,
77    isAuthenticated: PropTypes.string,
78    isAdmin: PropTypes.bool.isRequired
79  }
80
81  const mapStateToProps = state => ({
82    category: state.category,
83    isAuthenticated: state.auth.isAuthenticated,
84    isAdmin: state.auth.isAdmin
85  });
86
87
88
89  export default connect(mapStateToProps, { getCategories })(CategoriesList);
```

Παράδειγμα του κώδικα ενός Reducer.

```
categoryReducer.js  ↳ ×  categoryActions.js
Thesis JavaScript Content Files  -  {} "categoryReducer"
1  import { GET_CATEGORIES, ADD_CATEGORY, CATEGORIES_LOADING, CATEGORY_FAIL, CATEGORIES_FAIL } from '../actions/types';
2
3  const initialState = {
4    categories: [],
5    loading: false
6  }
7
8  export default function(state = initialState, action) {
9    switch (action.type) {
10     case GET_CATEGORIES:
11       return {
12         ...state,
13         categories: action.payload,
14         loading: false
15       }
16     case ADD_CATEGORY:
17       return {
18         ...state,
19         categories: [action.payload, ...state.categories]
20       }
21     case CATEGORIES_LOADING:
22       return {
23         ...state,
24         loading: true
25       }
26     case CATEGORIES_FAIL:
27       return {
28         ...state,
29         categories: [],
30         loading: false
31       }
32     case CATEGORY_FAIL:
33     default:
34       return state;
35   }
```

Παράδειγμα του κώδικα κάποιων Actions.

```

categoryActions.js  categoryReducer.js
Thesis JavaScript Content Files  {} "categoryActions"
1  import { GET_CATEGORIES, ADD_CATEGORY, CATEGORIES_LOADING, CATEGORY_FAIL, CATEGORIES_FAIL } from './types';
2  import axios from 'axios';
3  import { tokenConfig } from './authActions';
4  import { returnErrors } from './errorActions';
5
6  export const getCategories = () => dispatch => {
7    dispatch(setCategoriesLoading());
8    axios
9      .get('/api/categories')
10     .then(res =>
11       dispatch({
12         type: GET_CATEGORIES,
13         payload: res.data
14       }))
15     .catch(err => {
16       dispatch(returnErrors(err.response.data, err.response.status, 'CATEGORIES_FAIL'));
17       dispatch({
18         type: CATEGORIES_FAIL
19       })
20     });
21  });
22
23
24  export const addCategory = (category) => (dispatch, getState) => {
25
26    if (!category.name) {
27      dispatch(returnErrors({ msg: "Please enter category's name" }, 400, 'CATEGORY_FAIL'));
28      dispatch({
29        type: CATEGORY_FAIL
30      })
31      return;
32    }
33
34    axios
35      .post('/api/categories', category, tokenConfig(getState))
36      .then(res =>
37        dispatch({
38          type: ADD_CATEGORY,
39          payload: res.data
40        }))
41      .catch(err => {
42        dispatch(returnErrors(err.response.data, err.response.status, 'CATEGORY_FAIL'));
43        dispatch({

```

Βασικές συναρτήσεις που χρησιμοποιούμε μέσα στα Components μας είναι οι `Render()`, `componentDidMount()`, `componentDidUpdate()`. Με την `Render()` στην ουσία δείχνουμε τι θα εμφανιστεί στον χρήστη μας. Με `componentDidMount()` ορίζουμε τι ενέργειες θα γίνουν όταν θα φορτώσει ένα Component. Και τέλος, με την `componentDidUpdate()` ορίζουμε τι ενέργειες θα γίνουν όταν γίνει κάποια αλλαγή στο Component, όπως για παράδειγμα το πάτημα ενός κουμπιού όταν καλούμε μια συνάρτηση.

Η `css` που χρησιμοποιήθηκε για την εμφάνιση της ιστοσελίδας δεν βρίσκεται σε κάποιο ξεχωριστό αρχείο αλλά στον κώδικα του κάθε Component.

Express.js and Node.js Server

Με την Node.js γράψαμε τον κώδικα όλου του backend. Και μέσω της Express.js μπορεί να τρέχει η JavaScript σε μηχανήμα αντί για web browser.

Στα αρχεία που βρίσκονται στο φάκελο models είναι τα collections και εκεί ορίζουμε τα στοιχεία του κάθε collection (π.χ name,id).

Στα αρχεία που βρίσκονται στο φάκελο api ορίζουμε τα routes μας κάνοντας κάποια POST, GET και DELETE.

Παρακάτω φαίνονται οι φάκελοι και τα αρχεία που αναφέρθηκαν παραπάνω, όπως και ένα ενδεικτικό παράδειγμα με POST,GET και DELETE.

The screenshot shows a Visual Studio Code editor with a project named 'Thesis' open. The Solution Explorer on the left shows the project structure, with the 'api' folder under 'routes' selected. The main editor displays the code for 'places.js', which defines three routes: GET, POST, and DELETE. The GET route is for '/admin' and returns a list of places. The POST route is for '/accept/:id' and updates a place. The DELETE route is for '/delete/:id' and removes a place. The code includes authentication checks and error handling.

```

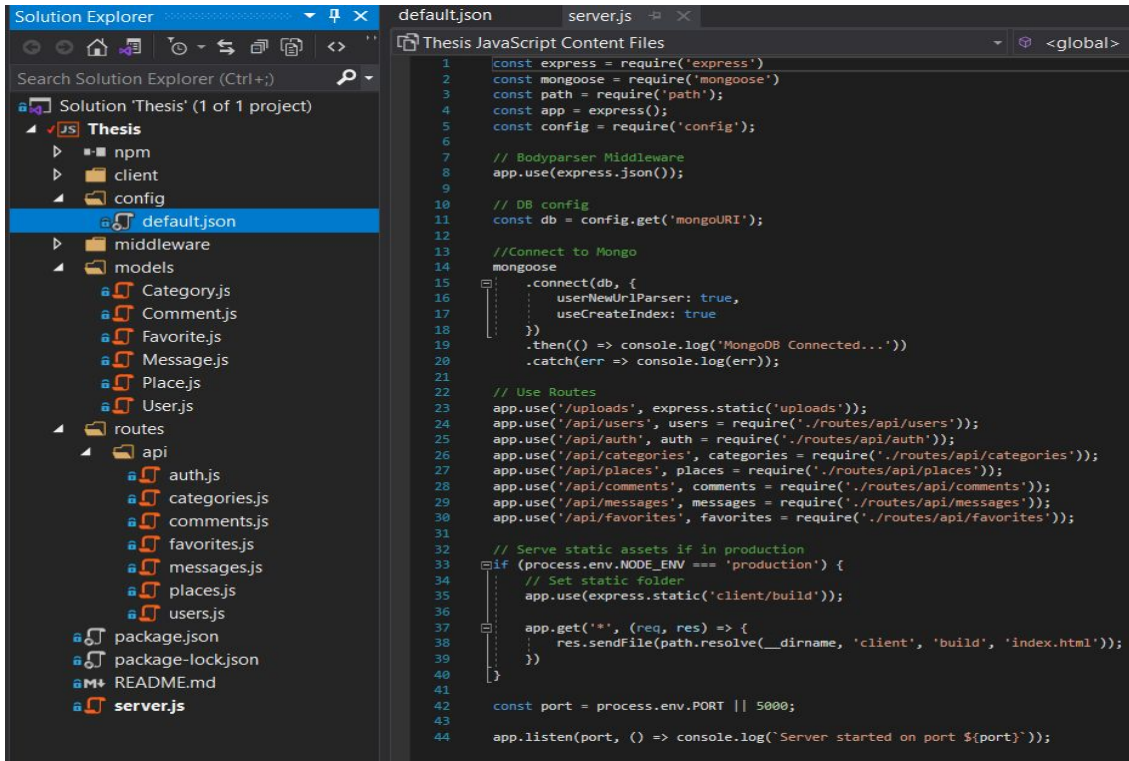
79
80
81 // @route GET api/places/admin
82 // @desc pending places
83 // @access Private
84 router.get('/admin', auth, (req, res) => {
85   if (!req.user.admin) return res.status(400).json({ msg: "You are not an admin,you do not have access!" })
86   Place.find( { accepted: false } )
87     .sort({ date: 1 })
88     .then(places => res.json(places));
89 });
90
91
92 // @route POST api/places/accept/:id
93 // @desc ACCEPT a place
94 // @access Private
95 router.post('/accept/:id', auth, (req, res) => {
96   if (!req.user.admin) return res.status(400).json({ msg: "You are not an admin,you do not have access!" })
97   Place.findById(req.params.id)
98     .then(place => {
99     if (!place) return res.status(400).json({ msg: 'Wrong place ID' })
100    place.accepted = true; place.save().then(() => res.json({ success: true })))
101  })
102  .catch(err => res.status(404).json({ success: false })))
103 });
104
105 // @route DELETE api/places/delete/:id
106 // @desc DELETE a place
107 // @access Private
108 router.delete('/delete/:id', auth, (req, res) => {
109   if (!req.user.admin) return res.status(400).json({ msg: "You are not an admin,you do not have access!" })
110   Place.findById(req.params.id)
111     .then(place => {
112     if (!place) return res.status(400).json({ msg: 'Wrong place ID' })
113     place.remove().then(() => {
114       imgs = place.images;
115       imgs.forEach((img) => {
116         const index = img.search("uploads");
117         var del = img.substring(index);
118         fs.unlink(del, (err) => {
119           if (err) return res.status(400).json('Error:' + err)
120         });
121       });
122     });
123     res.json({ success: true })
124   })
125   .catch(err => res.status(404).json({ success: false })))
126 });

```

Below the code editor, the Error List is visible, showing a Developer PowerShell icon.

MongoDB Database

Η βάση μας είναι η MongoDB και είναι μια NoSQL βάση δεδομένων. Δημιουργήσαμε ένα account στην σελίδα <https://www.mongodb.com/> και εκεί δημιουργήσαμε μια καινούργια βάση δεδομένων με το όνομα [TouristWebSite](#). Οπότε η βάση μας είναι ανεβασμένη online και όχι τοπικά. Μέσω ενός κλειδιού που πήραμε από την σελίδα της MongoDB συνδέσαμε την βάση με το project.



The screenshot shows a VS Code editor with a project named 'Thesis' and a file named 'server.js'. The project structure includes folders for 'npm', 'client', 'config', 'default.json', 'middleware', 'models', 'routes', and 'api'. The 'server.js' file contains the following code:

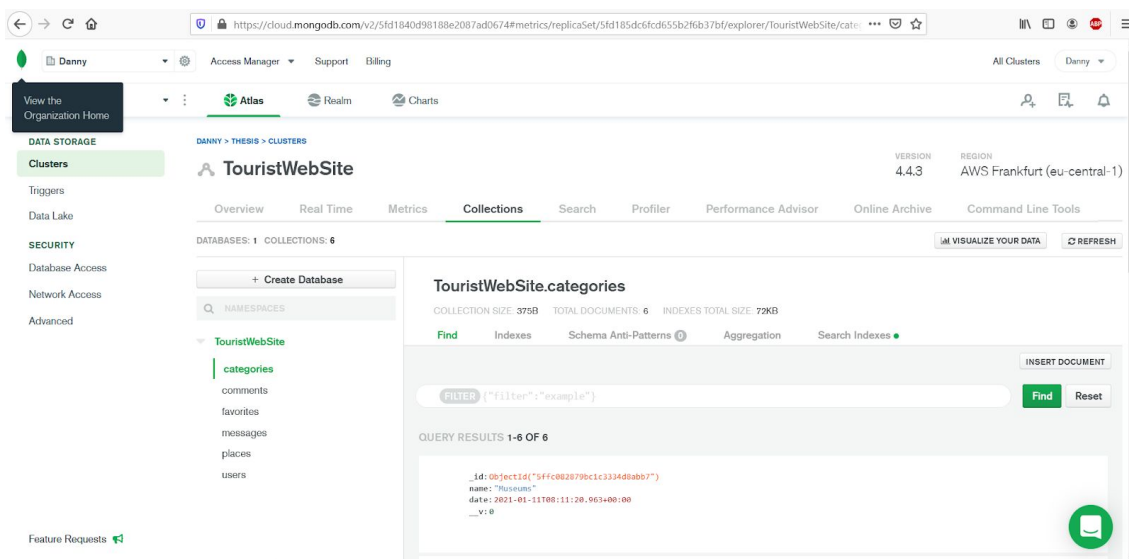
```

1  const express = require('express')
2  const mongoose = require('mongoose')
3  const path = require('path')
4  const app = express()
5  const config = require('config')
6
7  // Bodyparser Middleware
8  app.use(express.json())
9
10 // DB config
11 const db = config.get('mongoURI')
12
13 //Connect to Mongo
14 mongoose
15   .connect(db, {
16     userNewUrlParser: true,
17     useCreateIndex: true
18   })
19   .then(() => console.log('MongoDB Connected...'))
20   .catch(err => console.log(err))
21
22 // Use Routes
23 app.use('/uploads', express.static('uploads'));
24 app.use('/api/users', users = require('./routes/api/users'));
25 app.use('/api/auth', auth = require('./routes/api/auth'));
26 app.use('/api/categories', categories = require('./routes/api/categories'));
27 app.use('/api/places', places = require('./routes/api/places'));
28 app.use('/api/comments', comments = require('./routes/api/comments'));
29 app.use('/api/messages', messages = require('./routes/api/messages'));
30 app.use('/api/favorites', favorites = require('./routes/api/favorites'));
31
32 // Serve static assets if in production
33 if (process.env.NODE_ENV === 'production') {
34   // Set static folder
35   app.use(express.static('client/build'));
36
37   app.get('*', (req, res) => {
38     res.sendFile(path.resolve(__dirname, 'client', 'build', 'index.html'));
39   })
40 }
41
42 const port = process.env.PORT || 5000;
43
44 app.listen(port, () => console.log(`Server started on port ${port}`));

```

Εδώ φαίνεται ο κώδικας που γίνεται η σύνδεση. Στο αρχείο default.json έχουμε αποθηκευμένο μέσα το κλειδί που αναφέρθηκε παραπάνω.

Παρακάτω φαίνεται πως είναι η βάση μας στο Cloud της MongoDB.



Ανάπτυξη Web Εφαρμογής με Θέμα τον Τουρισμό και Χρήση MERN Stack (MongoDB, Express, React, Node)

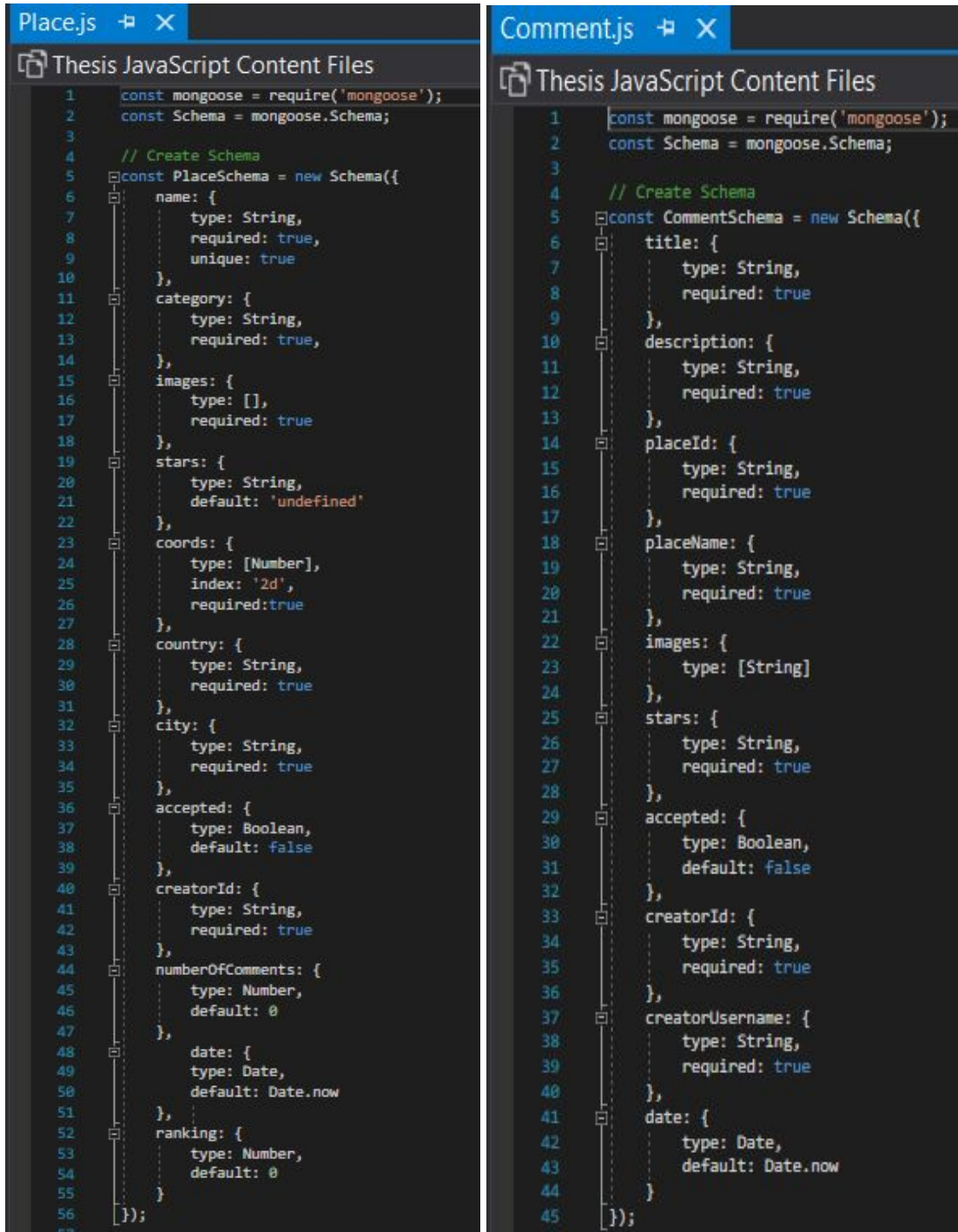
Και εδώ φαίνονται όλα τα Collection μαζί με τα πεδία τους γραμμένα κώδικα Node.JS.

```
User.js
1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3
4 // Create Schema
5 const UserSchema = new Schema({
6   username: {
7     type: String,
8     required: true,
9     unique: true
10  },
11  email: {
12    type: String,
13    required: true,
14    unique: true
15  },
16  password: {
17    type: String,
18    required: true
19  },
20  name: {
21    type: String,
22    required: true
23  },
24  surname: {
25    type: String,
26    required: true
27  },
28  admin: {
29    type: Boolean,
30    default: false
31  },
32  date: {
33    type: Date,
34    default: Date.now
35  }
36 });
37
```

```
Message.js
1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3
4 // Create Schema
5 const MessageSchema = new Schema({
6   title: {
7     type: String,
8     required: true,
9   },
10  message: {
11    type: String,
12    required: true,
13  },
14  receiverUsername: {
15    type: String,
16    required: true
17  },
18  senderUsername: {
19    type: String,
20    required: true
21  },
22  receiverId: {
23    type: String,
24    required: true
25  },
26  senderId: {
27    type: String,
28    required: true
29  },
30  date: {
31    type: Date,
32    default: Date.now
33  }
34 });
35
```

```
Favorite.js
1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3
4 // Create Schema
5 const FavoriteSchema = new Schema({
6   placeId: {
7     type: String,
8     required: true
9   },
10  userId: {
11    type: String,
12    required: true
13  },
14  date: {
15    type: Date,
16    default: Date.now
17  }
18 });
19
```

```
Category.js
1 const mongoose = require('mongoose');
2 const Schema = mongoose.Schema;
3
4 // Create Schema
5 const CategorySchema = new Schema({
6   name: {
7     type: String,
8     required: true,
9     unique: true
10  },
11  date: {
12    type: Date,
13    default: Date.now
14  }
15 });
16
```



```
Place.js
Thesis JavaScript Content Files
1  const mongoose = require('mongoose');
2  const Schema = mongoose.Schema;
3
4  // Create Schema
5  const PlaceSchema = new Schema({
6    name: {
7      type: String,
8      required: true,
9      unique: true
10   },
11   category: {
12     type: String,
13     required: true,
14   },
15   images: {
16     type: [],
17     required: true
18   },
19   stars: {
20     type: String,
21     default: 'undefined'
22   },
23   coords: {
24     type: [Number],
25     index: '2d',
26     required: true
27   },
28   country: {
29     type: String,
30     required: true
31   },
32   city: {
33     type: String,
34     required: true
35   },
36   accepted: {
37     type: Boolean,
38     default: false
39   },
40   creatorId: {
41     type: String,
42     required: true
43   },
44   numberOfComments: {
45     type: Number,
46     default: 0
47   },
48   date: {
49     type: Date,
50     default: Date.now
51   },
52   ranking: {
53     type: Number,
54     default: 0
55   }
56 });
57
```

```
Comment.js
Thesis JavaScript Content Files
1  const mongoose = require('mongoose');
2  const Schema = mongoose.Schema;
3
4  // Create Schema
5  const CommentSchema = new Schema({
6    title: {
7      type: String,
8      required: true
9    },
10   description: {
11     type: String,
12     required: true
13   },
14   placeId: {
15     type: String,
16     required: true
17   },
18   placeName: {
19     type: String,
20     required: true
21   },
22   images: {
23     type: [String]
24   },
25   stars: {
26     type: String,
27     required: true
28   },
29   accepted: {
30     type: Boolean,
31     default: false
32   },
33   creatorId: {
34     type: String,
35     required: true
36   },
37   creatorUsername: {
38     type: String,
39     required: true
40   },
41   date: {
42     type: Date,
43     default: Date.now
44   }
45 });
46
```

Authentication και Token

Για το Authentication των χρηστών χρησιμοποιείται JSON web Token. Επιπλέον, για την κρυπτογράφηση των password των χρηστών χρησιμοποιείται το <https://www.npmjs.com/package/bcryptjs>.

Διαχείριση Errors και Προσβασιμότητα

Για τα Errors έχουν δημιουργηθεί τα ανάλογα Actions και ο ανάλογος Reducer. Γίνονται έλεγχοι και στο frontend για να εμφανίζονται τα αντίστοιχα μηνύματα στον χρήστη αλλά και στο backend που κάνει την ιστοσελίδα μας πιο ασφαλή σε επιθέσεις.

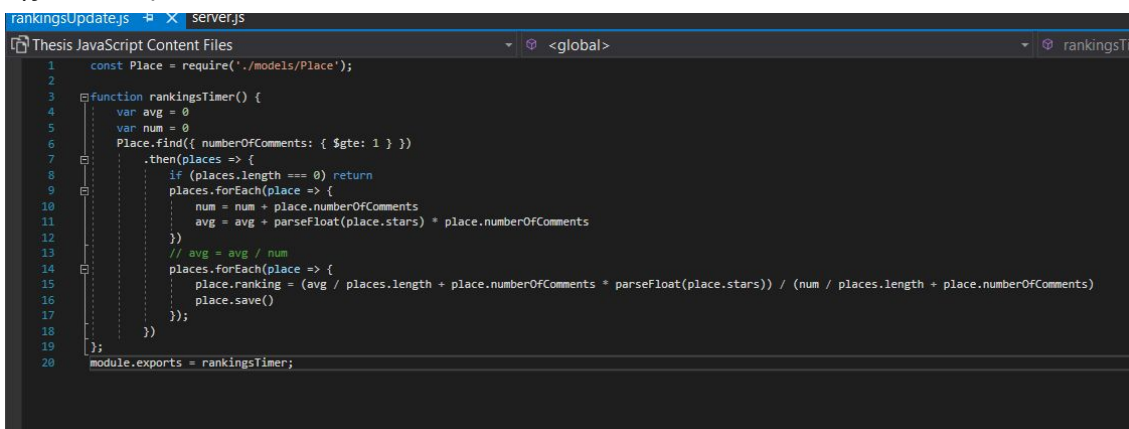
Σε οποιοδήποτε path της ιστοσελίδας γίνονται οι ανάλογοι έλεγχοι εάν ο χρήστης έχει το δικαίωμα προσβασιμότητας. Για παράδειγμα αν ένας συνδεδεμένος χρήστης προσπαθήσει να μπει σε ένα path που έχουν πρόσβαση μόνο οι Admin τότε δεν θα του επιτραπεί η είσοδος και θα εμφανιστεί και το ανάλογο μήνυμα.

Ranking Αλγόριθμος

Για την σειρά εμφάνισης των τοποθεσιών παίρνουμε υπόψη την μέση βαθμολογία τους αλλά και των αριθμό των comments που έχουν. Για αυτό το σκοπό δημιουργήσαμε μια μαθηματική έκφραση η οποία βασίζεται στο Bayesian Regret.

Προσθέτουμε ισάριθμα ψεύτικα (dummies) ratings σε κάθε τοποθεσία τα οποία έχουν ίση βαθμολογία με τον μέσο όρο του συνόλου των ratings όλων των κριτικών. Τα dummies είναι ισάριθμα με το σύνολο των πραγματικών κριτικών. Ύστερα υπολογίζοντας σε ένα νέο rating για κάθε τοποθεσία μειώνεται η βαρύτητα της βαθμολογίας της κάθε κριτικής, ειδικότερα σε μικρό αριθμό κριτικών. Επιπλέον, με αυτόν τον τρόπο αυξάνεται η βαρύτητα του αριθμού των κριτικών για κάθε τοποθεσία (popularity-based). Για παράδειγμα αν μια τοποθεσία έχει μέσο όρο βαθμολογίας 5 αστέρια αλλά μόνο με δυο κριτικές θα εμφανιστεί χαμηλότερα στην λίστα μας από μια τοποθεσία που έχει 4 αστέρια αλλά με είκοσι κριτικές.

Παρακάτω φαίνεται η υλοποίηση αυτής της μαθηματικής έκφρασης. Επιπρόσθετα, μέσω ενός timer στον server μας τρέχουμε αυτή την μαθηματική έκφραση σε χρόνο τον οποίο τον ορίζουμε εμείς ανάλογα με τις ανάγκες μας, χρησιμοποιώντας την συνάρτηση setInterval() της JavaScript.



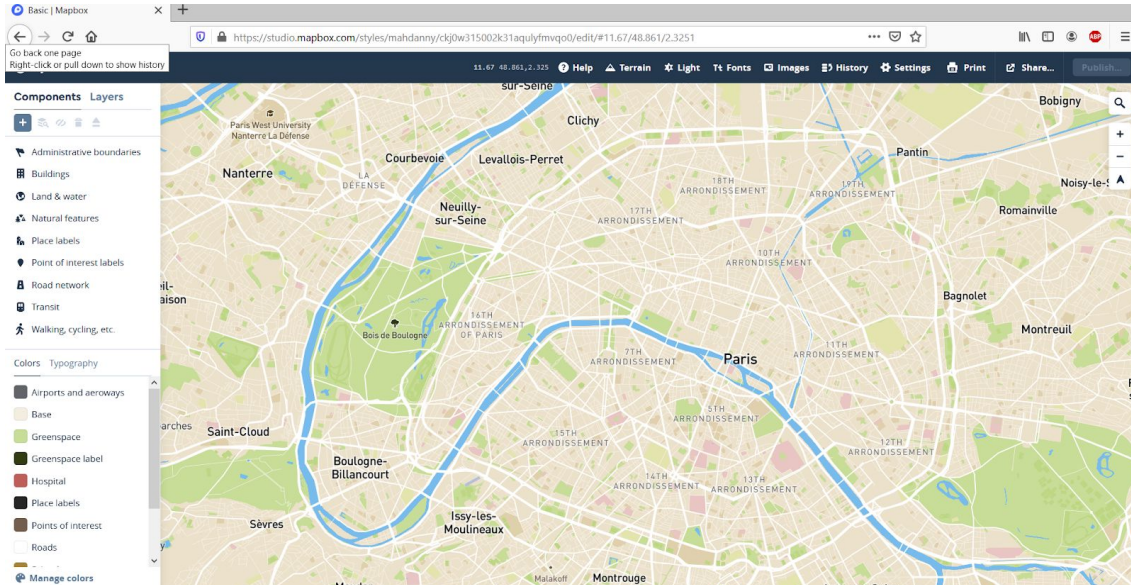
```

rankingsUpdate.js  X  server.js
Thesis JavaScript Content Files  <global>  rankingsTim
1  const Place = require('./models/Place');
2
3  function rankingsTimer() {
4    var avg = 0
5    var num = 0
6    Place.find({ numberOfComments: { $gte: 1 } })
7      .then(places => {
8        if (places.length === 0) return
9        places.forEach(place => {
10         num = num + place.numberOfComments
11         avg = avg + parseFloat(place.stars) * place.numberOfComments
12       })
13       // avg = avg / num
14       places.forEach(place => {
15         place.ranking = (avg / places.length + place.numberOfComments * parseFloat(place.stars)) / (num / places.length + place.numberOfComments)
16         place.save()
17       });
18     });
19   });
20   module.exports = rankingsTimer;

```


Χάρτης

Για τον χάρτη χρησιμοποιείται το <https://www.mapbox.com/>. Σε αυτή την ιστοσελίδα δημιουργήσαμε τον δικό μας custom χάρτη και τον συνδέσαμε μέσω ενός κλειδιού με το project μας και έπειτα τον εμφανίζομαι σαν ένα απλό Component. Παρακάτω φαίνεται το customization και ο κώδικας που γράφτηκε για τον χάρτη.



```

Map.js
Thesis JavaScript Content Files
render
icon-image

1 import React, { Component } from 'react';
2 import ReactMapboxGL, { Layer, Feature } from 'react-mapbox-gl';
3 import style from './mapStyle/style.json';
4
5 const MapGL = ReactMapboxGL({
6   accessToken: "pk.eyJ1IjoibWZ0ZGZubnk1CjhiY2tqMHRlZ3AxMw53dTJ1bW14OHQ1a2d1eCJ9.y4qe8LsrRQyCknTu7oFAjQ",
7   attributionControl: false
8 });
9
10 export default class Map extends Component {
11   constructor(props) {
12     super(props);
13     this.state = {
14       coords: this.props.coords
15     };
16   }
17   componentDidUpdate(prevProps) {
18     if (this.props.coords !== prevProps.coords) {
19       this.setState({
20         coords: this.props.coords
21       });
22     }
23   }
24   onDragEnd=(e)=> {
25     this.props.onChangeCoords(e.lngLat)
26   }
27   render(){
28     if(String(this.state.coords[0])=== "undefined") return ((null));
29     return (
30       <div>
31         <MapGL containerStyle={this.props.sz}
32           style={style} center={[this.state.coords[0], this.state.coords[1]]>
33           <Layer type="symbol" id="marker" layout={{ 'icon-image': 'marker-15', 'icon-size': 3}}
34             {this.props.draggable === 'true' ? <Feature onDragEnd={this.onDragEnd} draggable="true" coordinates={[this.state.coords[0], this.state.coords[1]]} />
35             : <Feature coordinates={[this.state.coords[0], this.state.coords[1]]} />
36           </Layer>
37         </MapGL>
38         <p style={{ fontSize: "1vw" }}>{`Lon: ` + this.state.coords[0] + `, Lat: ` + this.state.coords[1]} </p>
39       </div>
40     );
41   }
42 }
  
```

Εικόνες

Παράδειγμα κώδικα για την αποθήκευση εικόνων στο backend.

```

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, './uploads/');
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + "_" + file.originalname);
  }
});

const fileFilter = (req, file, cb) => {
  if (file.mimetype === "image/x-png" || file.mimetype === "image/png" ||
    file.mimetype === "image/jpg" || file.mimetype === "image/jpeg")
    cb(null, true);
  else cb(new Error("Only image files are allowed!"), false);
}

const upload = multer({
  storage: storage,
  limits: {
    fileSize: 1024 * 1024,
    files: 3
  },
  fileFilter: fileFilter
});

```

Παράδειγμα κώδικα για τις εικόνες στο frontend.

```

124 //Remove Image
125 removeImage = (image) => {
126   if (String(image.name) === "undefined") this.setState({
127     del: this.state.del.concat([image])
128   });
129   this.setState({
130     images: this.state.images.filter(img => img.src !== image.src)
131   });
132 }
133 //Return images at add comment
134 renderImages = () => {
135   const { images } = this.state;
136   const style = {
137     width: "100%",
138     height: "100%",
139     backgroundPosition: "50% 50%",
140     backgroundRepeat: "no-repeat",
141     backgroundSize: "cover",
142   }
143   var x = 0;
144   return images.map((image) => {
145     return (
146       <col style={{ width: "30%", height: "30%" }} key={"col" + (x++)}>
147         <Image src={image.src} style={style} rounded />
148         <Button color='green' onClick={() => this.removeImage(image)}
149           style={{
150             backgroundColor: "transparent", borderColor: "transparent",
151             position: 'absolute', color: 'red', top: -5, right: 5
152           }}>
153           X
154         </Button>
155       </col>
156     );
157   });
158 }
159
160 //Return images at get comments
161 renderImagesComments = (comment) => {
162   const style = {
163     width: "70%",
164     height: "70%",
165     backgroundPosition: "50% 50%",
166     backgroundRepeat: "no-repeat",
167     backgroundSize: "cover",
168   }
169   return comment.images.map((image) => {
170     return (
171       <col style={{ width: "30%", height: "30%" }} key={"col" + image}>
172         <Image src={image} style={style} rounded />
173       </col>
174     );
175   });

```

```

//Verify that the file is image
verifyFile = (file, x) => {
  if (file) {
    const current = file;
    const currentType = current.type;
    const currentSize = current.size;
    if (currentSize > imageMaxSize) {
      this.setState({
        formError: this.state.formError +
          currentSize + " bytes is too large(file number " + x + ")
      });
      return false;
    }
    if (!fileTypesArray.includes(currentType)) {
      this.setState({
        formError: this.state.formError +
          "Only image files are allowed(file number " + x + ")"
      });
      return false;
    }
    return true;
  }
}

//Upload image
onDrop = (accepted, rejected) => {
  var x = 1;
  if (rejected) {
    for (const current of rejected) this.verifyFile(current, x++);
  }
  if (accepted) {
    if (accepted.length + this.state.images.length > 3) {
      alert("Only 3 images per place is allowed!");
      return;
    }
    for (const current of accepted) {
      const isVerified = this.verifyFile(current, x++);
      if (isVerified) {
        const reader = new FileReader();
        reader.addEventListener("load", () => {
          const res = reader.result;
          current['src'] = res;
          this.setState({
            images: this.state.images.concat(current)
          });
        }, false);
        reader.readAsDataURL(current);
      }
    }
  }
}

```

Επιπλέον

Για τα rating αστέρια χρησιμοποιείται

<https://www.npmjs.com/package/react-rating-stars-component> .

Για την καρδούλα των favorites χρησιμοποιείται η react-animated-heart .

Για την σελιδοποίηση χρησιμοποιείται <https://www.npmjs.com/package/react-js-pagination>.

Πως να Τρέξει η Ιστοσελίδα

Για να γραφτεί το project χρησιμοποιήθηκε ως editor το Visual Studio 2019. Για να τρέξει ο κώδικας σε VS πρέπει να υπάρχει εγκατεστημένο το npm πακέτο. Επιπλέον, στον βασικό φάκελο αλλά και στον φάκελο client πρέπει να γίνει από μια φορά η εντολή npm install ώστε να εγκατασταθούν όλα τα dependencies και τέλος για να τρέξει τοπικά η ιστοσελίδα, στον βασικό φάκελο πρέπει να γίνει εντολή npm run dev η οποία θα τρέξει και τον client αλλά και τον server

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η δημιουργία μιας ιστοσελίδας από την αρχή δεν είναι το πιο εύκολο εγχείρημα. Όμως, λόγω της τεχνολογίας MERN καταφέραμε να δημιουργήσουμε πολύ πιο εύκολα μια full stack ιστοσελίδα.

Η ιδέα για την δημιουργία ενός ιστότοπου με τοποθεσίες τουριστικού ενδιαφέροντος θα μπορεί να έχει μεγάλες επεκτάσεις λόγω της αυξημένης ζήτησης τέτοιων εφαρμογών στον τομέα του τουρισμού. Επιπλέον, στην ιστοσελίδα TripSpot ο χρήστης μπορεί να προτείνει αυτός συνεχώς τοποθεσίες κάνοντας να νιώθει και αυτός πως αποτελεί ένα μέρος του όλου εγχειρήματος.

Υπάρχουν αρκετές μελλοντικές προσθήκες που θα μπορούσαν να μπουν στην ιστοσελίδα, όμως χρειάζονται ανάλογα και τον απαραίτητο χρόνο και εργασία. Η εισαγωγή με βασικές τοποθεσίες στην βάση δεδομένων είναι μια από αυτές. Επιπλέον, θα μπορούσε να μπει ένας αλγόριθμος με recommendation system ώστε να προτείνει στους χρήστες τοποθεσίες που ενδεχομένως τους ενδιαφέρουν. Επίσης, η προσθήκη like και dislike στα comments. Και τέλος, ίσως η προσθήκη ενός ημερολογίου όπου οι χρήστες θα μπορούν να προγραμματίσουν ένα πλάνο ταξιδιού με συγκεκριμένες ημερομηνίες.

Βιβλιογραφία

1. <https://www.mongodb.com/>
2. <https://reactstrap.github.io/>
3. <https://react-redux.js.org/>
4. <https://www.react.express/>
5. <https://www.npmjs.com/>
6. <https://jwt.io/>
7. <https://www.mapbox.com/>