ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Τεχνολογιες Εφαρμογων Διαδικτυου

Εαρινο εξαμηνο 2020

ΥΠΟΧΡΕΩΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογη Διαδικτυου Ενοικιασης Δωματιων / Κατοικιων

Αριθμός Μητρώου(ΑΜ):

1115201700207

1115201700203

Ονοματεπωνυμο:

Κωνσταντίνος Γεωργιου

Λεωνίδας Εφραιμ

Ακαδημαικη Χρονια 2019-2020

ΠΕΡΙΕΧΟΜΕΝΑ

1

ΕΙΣΑΓΩΓΗ

Στο πλαίσιο του μαθήματος κληθήκαμε να υλοποιήσουμε μια διαδικτυακή εφαρμογή που είχε σκοπό την ενοικίαση δωματίων και κατοικιών. Επειδή μας δόθηκε μεγάλος βαθμός ελευθερίας όσον αφορά την υλοποίση και τη σχεδίαση της εφαρμογής, κάναμε μια μελέτη στο διαδίκτυο με βάση τις ενότητες και τα αντικείμενα που διδαχθήκαμε στο μάθημα ,με σκοπό να βρούμε τις πιο κατάλληλες τεχνολογίες που θα χρησιμοποιούσαμε για την υλοποίηση μας.

2

ΕΓΚΑΤΑΣΤΑΣΗ

2.1 BACKEND

Αρχικά μελετήσαμε διάφορα μοντέλα και τεχνολογίες όσον αφορά τη δομή του server που θα κατασκευάζαμε. Τελικά καταλήξαμε στην παρακάτω δόμη όσον αφορά τις τεχνολογίες που θα χρησιμοποιηθούν στο νωτιαίο άκρο της εφαρμογής (Backend).

NodeJs
Express
PostgresQL

2.1.1 NodeJs

Η απόφασή μας να χρησιμοποιήσουμε NodeJs λήφθηκε με βάση τις δυνατότητες και τη χρηστικότητα που μας παρέχει. Κατά κύριο λόγο την επιλέξαμε γιατί μας δίνει τη δυνατότητα να χρησιμοποιήσουμε μόνο μια γλώσσα προγραμματισμού(Javascript) τόσο στο back-end όσο και στο front-end.Επίσης η Javascript είναι εύκολη στην κατανόηση και στη χρήση της. Η εγκατάσταση της NodeJs γίνεται με την εντολή **sudo apt install nodejs**
Επίσης χρειάζεται και η εγκατάσταση της Node Package Manager(NPM) η οποία είναι μια βιβλιοθήκη ανοιχτού κώδικα, που έχει Node.js πακέτα και γίνεται εγκατάσταση με την εντολή **sudo apt install npm**

2.1.2 Express

write your text here.Express

2.1.3 PostgresQL

Έχουμε επιλέξει την PostgresQL για βάση, επειδή είναι μια SQL like βάση δεδομένων στην οποία μπορούμε να γράψουμε πιο κατανοητά ερωτήματα στον χρήστη . Επίσης η PostgresQL υποστηρίζει JSON αρχεία που μας διευκολύνει με την React και την NodeJs.
Η εγκατάσταση της PostgresQL γίνεται με την παρακάτω εντολή
sudo apt-get install postgresql
Στη συνέχεια συνδεθήκαμε στην postgress
psql -U <user name>
και δημιουργήσαμε τη βάση με την εντολή
CREATE DATABASE Breezebnb
Αφού δημιουργήθηκε η βάση φορτώσαμε τους πίνακες και τις ρυθμίσεις της βάσης
cat database/config.sql | psql -d <database name> -U <user name>
cat models/* | psql -d <database name> -U <user name>
node database/generate-seed.js > database/seed.sql cat database/seed.sql | psql -d <database> -U <user>

Φόρτωση Βάσης στον Server

Για να φορτώσουμε τη βάση στον server δημιουργήσαμε ένα αρχείο με κατάληξη .env το οποίο αντιστοιχούσε στις κατάλληλες μεταβλητές με τα στοιχεία σύνδεσης της βάσης. Τα στοιχεία αυτά φορτώνονται στις συναρτήσεις που είναι υπεύθυνες για την επικοινωνία με τη βάση

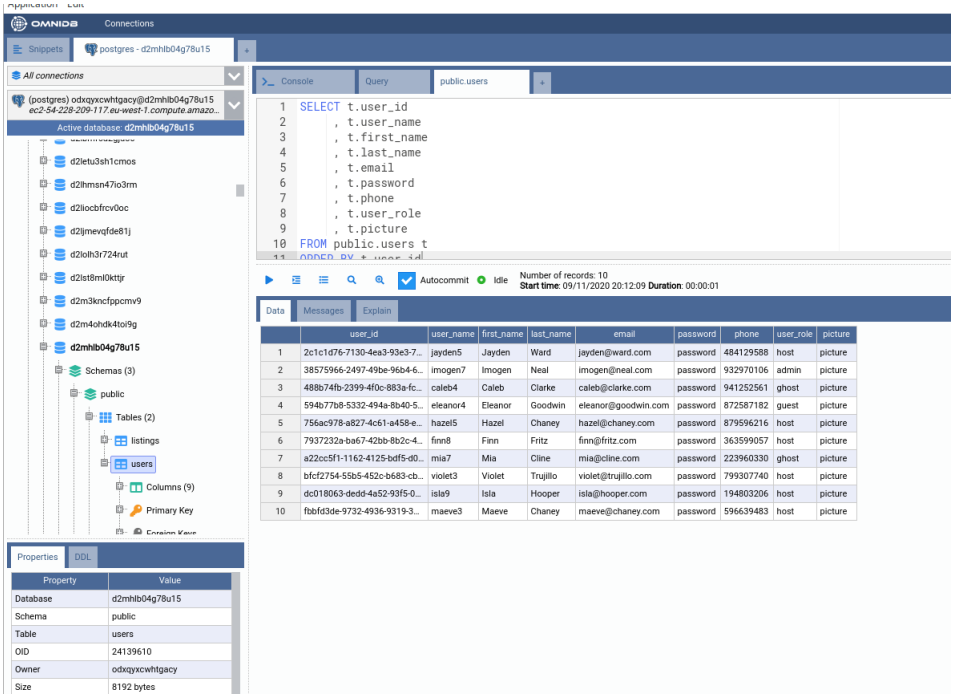
.env

```
# Port that server is listening
PORT = 5000
# Database credentials
DATABASE_URI = 'postgresql://postgres:apoel@localhost:5432/brezzebnb?currentSchema=public'
DATABASE_USER = "postgres"
DATABASE_USER_PW= "password"
DATABASE_HOST = "localhost"
DATABASE_PORT = 5432
DATABASE = "brezzebnb"
```

Διαχείριση Βάσης Δεδομένων - OmniDB

Τη διαχείριση της βάσης δεδομένων την κάναμε με την εφαρμογή OmniDB.

Καθορίζοντας τα στοιχεία σύνδεσης της βάσης μας, μπορούμε να δούμε από τη διεπαφή τα δεδομένα των πινάκων μας, καθώς και να τρέξουμε queries



2.2 FRONTEND

Όσον αφορά το μετωπιαίο άκρο (frontend) αποφασίσαμε να χρησιμοποιήσουμε React, η οποία είναι JavaScript βιβλιοθήκη που μας βοηθά να αναπτύξουμε μια UserInterface εφαρμογή. Επίσης χρησιμοποιήσαμε το μοντέλο MVC όσον αφορά τη διαχείριση της εφαρμογής.

2.2.1 React

Η React έχει αναπτυχθεί από το Facebook και το community η οποία είναι σε Javascript

2.2.2 MVC

Η εφαρμογή έχει στηθεί βάση του μοντέλου MVC(Model-View-Controler) Οπού ο μελίγα λόγια το View είναι υπεύθυνο για την γραφική αναπαράσταση της εφαρμογής. Ενώ το Model είναι υπεύθυνο για την δειαχήρηση των δεδομένων του συστήματος και ο Controler έχει την υπεθυνότητα να στέλνει εντολές στο Model και στο View. Αυτή η αρχιτεκτονική μας έχει βοηθήσει στο να οργανώσουμε καλητέρα τον κωδικά μας, καθός και στην καλήτερη δόμηση της εφαρμογής

2.3 ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΧΕΙΡΗΣΤΗ ΣΕΛΙΔΑΣ(ADMIN)

Για την δημιουρία του διαχειρηστη της σελιδας φτιάξαμε το αρχείο database/admin.js το οποίο ορίζει τα στοιχεία του admin της σελίδας και στην συνέχεια τα καταχωρεί στην βάση με την εντολή
node database/admin.js
Τα στοιχεία σύνδεσης του διαχειρηστη είναι
USERNAME:**admin**
PASSWORD:**admin**

2.4 ΠΑΓΚΟΣΜΙΟΣ ΙΣΤΟΣ (HEROKU APP)

Παράλληλα με την τοπική εφαρμογή, ανεβάσαμε στον παγκόσμιο ιστό την εφαρμογή μας, με τη βοήθεια του Heroku app. Το Heroku είναι μια cloud πλατφόρμα η οποία μας παρέχει δωρεάν hosting για τον server μας. Έτσι λοιπόν ανεβάσαμε τους 2 server μας (Backend - Frontend) καθώς και τη βάση μας στους παρακάτω συνδέσμους

BACK-END

breezebnb.herokuapp.com

WEB SERVER

breezebnb.herokuapp.com NA MPI TO FRONT END

3

BACKEND

3.1 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Στη βάση δεδομένων δημιουργήσαμε τους παρακάτω πίνακες:

listings (Πίνακας με τα καταλύματα)
Rentalsreserved (Πίνακας με τις κρατήσεις)
messages(Πίνακας με τα μηνύματα)
reviewslistings(Πίνακας με τις κριτικές για τους χρηστες)
reviewsusers(Πίνακας με τις κριτικές για τα καταλυματα)
addresses(Πίνακας με τις διυθυνσεις)
photoslistings(Πίνακας με τις φωτογραφίες των καταληματων)
tokens(Πίνακας με τα JWT για το authentication των χρηστών)
coordinates(Πίνακας με τις συντεταγμένες για το openstreetmaps)
listingamenities(Πίνακας με τις παροχές του καταλήματος)
listingspace(Πίνακας με τις πληροφορίες για τα χαρακτησιστικά του καταλύματος)
listingrules(Πίνακας με για τους κανόνες του καταλύματος)
users(Πίνακας με τους χρήστες)

Επίσης χρησιμοποιήσαμε ένα configuration για τις ρυθμίσεις στη βάση Εκεί ρυθμίσαμε το format της ημερομηνίας, έτσι ώστε να είναι το Ευρωπαϊκό,
καθορίσαμε enums για τα user roles και τα property types
και φορτώσαμε το uuid ossp το οποίο μας κάνει generate μοναδικά hash έτσι ώστε

να τα χρησιμοποιήσουμε για primary key σε διάφορους πίνακες

3.2 REST API BACKEND

Για την επικοινωνία με τη βάση δεδομένων χρησιμοποιήσαμε REST API. Στην ουσία είναι συναρτήσεις οι οποίες κάνουν διάφορα queries στη βάση για να πάρουν κάποια δεδομένα και να τα επιστρέψουν στον χρήστη. Για να κληθεί η συνάρτηση πρέπει να κάνουμε ένα request στον αντίστοιχο σύνδεσμο και να του δώσουμε τις σωστές παραμέτρους στη συνάρτηση
Για παράδειγμα αν θέλουμε να δούμε τα στοιχεία κάποιου διαμερίσματος το οποίο είναι διαθέσιμο προς ενοικίαση θα πρέπει να κάνουμε ένα get request στο σύνδεσμο localhost:5000/listings/ae867f1c-1b93-495e-9a32-d26dea7ac749
όπου το ae867f1c-1b93-495e-9a32-d26dea7ac749 είναι το id του καταλύματος. Το request αυτό θα καλέσει την κατάλληλη συνάρτηση η οποία θα εκτελέσει ένα SELECT query στη βάση για να πάρει τα στοιχεία του καταλύματος και να τα επιστρέψει σε μοφρή JSON

3.3 AUTHENTICATION (JWT)

Για το authentication χρησιμοποισαμε Json Web Tokens (JWT). Αρχικά δημιουργήσαμε ενα πίνακα tokens στην βάση η οποία κρατά active tokens που υπάρχουν. Επίσης στην βάση οι κωδικοί των χρηστών είναι αποθηκευμένοι σε μορφή hash.Όταν ένα χρήστης συνδεθέι στην σελίδα τότε γίνετε επιβεβαίωση του hash του κωδικού προσβασης που έδωσε ο χρήστης με το hash που βρίσκετε στην βάση. Αν ο κωδικός είναι σωστός τότε δημιουργήτε ενα token το οποίο αποθηκεύτε στο πινακα token και αποστέλετε στον browser του χρήστη. Στην συνέχεια ο χρηστής για να κανει οποίδήποτε ενέργεια που απετεί ταυτοποίηση χρησιομοποιάτε η συνάρτηση IsAuthenticed η οποία πέρνει σαν παράμετρο το token. Αν το token υπάρχει στον πινακα tokens τότε ο χρήστης μπορεί να κάνει την ενεργεία που επιθυμει

3.4 AUTHORIZATION

Επειδή υπάρχουν διάφοροι ρόλοι χρηστών, ο κάθε χρήστης έχει πρόσβαση σε διαφορετικά νήματα της σελίδας. Πχ ένας ο admin έχει πρόσβαση σελιδα διαχρησης της σελίδα ενώ ένας host δεν έχει πρόσβαση σε αυτή την σελίδα. Για να γίνει αυτή η ταυτοποίση υπάρχουν οι συνάρτησεις isUsernameUnique
isEmailUnique
isRoleValid
isRoleNotAdmin
isApproved
που ελέχουν τον ρόλο του κάθε χρήστη έτσι ώστε να τον ανακατευθύνουν στην κατάλληλη σελίδα

3.5 ΔΙΑΧΕΙΡΗΣΗ ΦΩΤΟΓΡΑΦΙΩΝ

Για την διαχείρηση φωτογραφιών στο backend χρησιμοιποιήσαμε την σελιδα cloudinary η οποία μας παρείχε την δυνατότα να ανεβάσουμε φωτογραφίες στους server τις με API

Για να χρησιμοιποίσουμε της υπηρεσίες της cloudirany προσθέσαμε στο αρχείο env τα παρακάτω στοιχεία έτσι ώστε να μπορέσουμε να συνδεθούμε με τον server τους.
CLOUDNAME=
APIKEY=
APISECRET=
Στην συνέχεια στο backend server φτίαξε τους κατάλληλους controlers έτσι ώστε στον σύνδεσμο

localhost:5000/listings/upload

Να μπορούμε να ανεβάζουμε φωτογραφίες με ένα post request δίνοντας το όνομα

του καταλήματος και το path της φωτογραφίας(το ίδιο ισχεί και για τους χρήστες στον αντισοιχο συνδεσμο)

3.6 OPENSTREETMAPS

Όσο αφορά τους χάρτες χρησιμοιποίσαμε την υπηρέσια mapbox η οποία σε συνδίασαμε το GeoCoder μας όπου μας δίνει τις συντεταγμένες ενός σημείου από τον χάρτή. Η κύρια δουλεία για τους χάρτες είναι στο frontend. Στο backend αυτό που είχαμε να κάνουμε ήταν απλά να αποθηκεύσουμε τις συντεταγμένες τον καταλήματων στον κατάλληλο πίνακα έτσι ώστε να τις εμφανίσουμε στον χάρτη στην συνέχεια. Το πακέτο του GeoCoder ήταν αυτό που μας έδινε τις συντεγμένες όπου με τους κατάλληλους controllers τις αποθηκεύαμε. Η υπολοιπη διαδικάσία της υλοιποίσης εξηγήτε στο front end γιατί εκεί είναι η περισσοτερη υλοιποίση

4

FRONTEND

4.1 BOOTSTRAP

Στο Frontend χρησιμοποίσαμε το Boostrap το οποίο είναι μια συλλογή εργαλείων που μας βοηθά στην ανάπτυξη διαδικτιακών εφαρμογών. Η συλλογή αυτή αρχικά αναπτύχθηκε από το Twitter ως ένα πλαίσιο για την ενθάρρυνση της συνέπειας στα εσωτερικά εργαλεία.

4.2 REST API FRONTEND AXIOS

Αφού έχουμε φτίαξει το API για το backend στην συνέχεια θα έπρεπε να ενώσουμε την frontend εφαρμογή με τα API. Αυτό έγινε με την βιβλιοθήκη AXIOS η οποία απλοιποίησε την διαδικασία αποστολής request. Για να χρησιμοιποίσουμε την βιβλιθήκη αυτή πρέπει πρώτα να την κάνουμε import στο αρχείο μας. Στην συνέχεια απλά καλούμε την συνάρτηση του axios δίνοντας σαν όρισμα την διεύθυση που θέλουμε να γίνει το request, τον τύπο του reqest καθώς body το οποίο περιέχει της κατάληλες παραμέτρους που χρειάζοντε