

«Δικτυοκεντρικά Πληροφοριακά Συστήματα»

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Ψηφιακών Συστημάτων

Ακαδημαϊκό έτος 2022-2023

Ημερομηνία παράδοσης: 27/02/2023

ΑΠΑΛΛΑΚΤΙΚΗ ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

Θέμα: Ανάπτυξη ενός δικτυοκεντρικού πληροφοριακού συστήματος κρατήσεων αυτοκινήτων

Επιμέλεια:

Αλέξανδρος Μπεβεράτος

- Ε19109
- alexandros.beveratos@gmail.com

ΠΕΡΙΕΧΟΜΕΝΑ

Σελίδα 1: Εξώφυλλο και πληροφορίες συγγραφικής ομάδας.

Σελίδα 2: Εισαγωγή και συνοπτική παρουσίαση της εργασίας.

Σελίδα 3: Περιγραφή συστήματος που υλοποιήθηκε

Σελίδες 4-9: Υλοποίηση συστήματος.

Σελίδα 10: Εγχειρίδιο διαχειριστή

Σελίδες 10-16: Εγχειρίδιο χρήστη

ΕΙΣΑΓΩΓΗ

Στο πλαίσιο ανάπτυξης αυτού του δικτυοκεντρικού πληροφοριακού συστήματος κρατήσεων αυτοκινήτων έχουν δημιουργηθεί δύο υποσυστήματα, ένα με την χρήση node express.js framework για την διαχείριση των οντοτήτων και της βάσης δεδομένων του συστήματος και ένα με την χρήση React framework για τις λειτουργίες του πληροφοριακού συστήματος από την πλευρά του χρήστη.

Σκοπός του συστήματος είναι η διαχείριση κρατήσεων ενός καταστήματος ενοικιάσεων αυτοκινήτων και η επίδειξη του στόλου του. Το σύστημα περιέχει τις εξής σελίδες: Αρχική Σελίδα(Home), Dashboard, Στόλος(Fleet), Κρατήσεις(Reservations) και για τον διαχειριστή υπάρχει σελίδα Admin όπου μπορεί να προσθέτει, να τροποποιεί ή να αφαιρεί οχήματα.

Περιγραφή συστήματος που υλοποιήθηκε

Αρχικά, όπως αναφέρθηκε προηγουμένως, το σύστημα αφορά την ενοικίαση αυτοκινήτων και την διαχείριση χρηστών, οπότε χρησιμοποιείται από τον έμπορο αυτοκινήτων που τα νοικιάζει.

Όσον αφορά την λειτουργικότητά του όμως το σύστημα χρησιμοποιεί 2 υποσυστήματα που τρέχουν σε διαφορετικά ports αλλά επικοινωνούν μεταξύ τους με Http requests. Το πρώτο και βασικότερο σύστημα είναι ο server ο οποίος διαχειρίζεται την βάση δεδομένων MongoDB. Το δεύτερο σύστημα είναι ο client ο οποίος διαχειρίζεται την απεικόνιση της ιστοσελίδας στον χρήστη αλλά και κάποια δεδομένα πριν πάνε στον server. Όλο το σύστημα βασίζεται στην γλώσσα προγραμματισμού JavaScript και υποστηρίζεται με frameworks του Node.js(Μέσω NPM).

Ο διαχειριστής εισάγοντας το url της σελίδας διαχειριστή και ύστερα το όνομα και τον κωδικό, μπορεί να μπει και να κάνει αλλαγές στους χρήστες και στις κρατήσεις. Για αλλαγές στον στόλο των αυτοκινήτων ο διαχειριστής μπορεί απλά να μπει με τον λογαριασμό του και να πατήσει το navbar link 'Admin'. Εκεί θα μπορεί να κάνει τις απαραίτητες αλλαγές.

Ο χρήστης θα μπορεί να μπαίνει στην σελίδα και να δει το homepage, τον στόλο και το login/register. Εφόσον κάνει login θα μπορεί να δει το dashboard με τα στοιχεία του και τις κρατήσεις και θα μπορεί να μπει στην σελίδα που κάνεις κράτηση και να αιτηθεί μια γράφοντας την πινακίδα του αυτοκινήτου που θέλει να νοικιάσει. Ο διαχειριστής με την σειρά του θα την ελέγξει να μην συμπίπτει με άλλη κράτηση και θα την κάνει accept ή reject στο αντίστοιχο field στην σελίδα διαχειριστή.

Υλοποίηση συστήματος

Server:

Ο server έγινε initialized αφού φτιάξαμε το server.js, γράφοντας 'npm init -y'. Οπότε δημιουργήθηκαν τα αρχεία package.json και package-lock.json που έχουν να κάνουν με τα dependencies. Ύστερα το server.js μπορεί να τρέξει γράφοντας npm start στο console του server folder. Για την βάση δεδομένων χρησιμοποιείται το MongoDB μέσω του Docker, αλλά με την χρήση του αρχείου docker-compose.yml και ύστερα την εντολή docker-compose up ώστε να γίνει initialized. Για να τρέχει ο server και τα routes χρησιμοποιείται το express.js framework που έχει μπει ως package import. Για την εμφάνιση της σελίδας διαχειριστή χρησιμοποιείται ejs αντί για html καθώς είναι συμβατό με το express. Το σχήμα της βάσης δεδομένων για κάθε είδος οντότητας είναι στον φάκελο models όπου υπάρχει ένα .js αρχείο για το καθένα. Επίσης χρησιμοποιήθηκαν και τα Imports: cors (για cross origin μεταφορές πληροφοριών), multer(για file upload management), bodyparser (για το σώμα που δέχεται ο server), mongoose(για την διαχείριση της ΒΔ) και άλλα.

Όσον αφορά την λειτουργία του, καλούμε το express framework, το οποίο κάναμε import ως app, να χρησιμοποιήσει τα υπόλοιπα imports/εργαλεία με την χρήση του app.use. Για το routing το app χρησιμοποιεί τα get, post, κλπ. Functions. Ξεκινώντας με την σελίδα διαχειριστή, έχουμε 5 βασικά routes όπου τα 4 δέχονται μέθοδο GET και 1 που δέχεται μέθοδο POST. Η αρχική είναι η *app.get('/')* που μας κάνει render την σελίδα η οποία δέχεται σαν μεταβλητή το session του χρήστη και μέσα στο ejs ελέγχει αν έχει συνδεθεί ο διαχειριστής ή όχι και εμφανίζει τα ανάλογα στοιχεία. Εφόσον δεν είναι συνδεδεμένος και πατήσει login, η φόρμα θα στείλει post request στο *app.post('/login')* όπου ελέγχονται τα στοιχεία που έχουν εισαχθεί και αν είναι username: root και password: 123 τότε στο session θα φαίνεται ως authenticated και θα γίνει redirect πίσω στην αρχική όπου θα μπορεί πλέον να επιλέξει τι στοιχεία θέλει να δει ή να αλλάξει. Τα υπόλοιπα renders εισάγονται τα εκάστοτε στοιχεία (χρηστών, κρατήσεων) αφού κάνει query ο σέρβερ. Οι τροποποιήσεις γίνονται πάλι μέσω του σέρβερ χρησιμοποιώντας και άλλα routes τύπου CRUD για το κάθε είδος οντότητας (πχ *app.post('/user/accept')*).

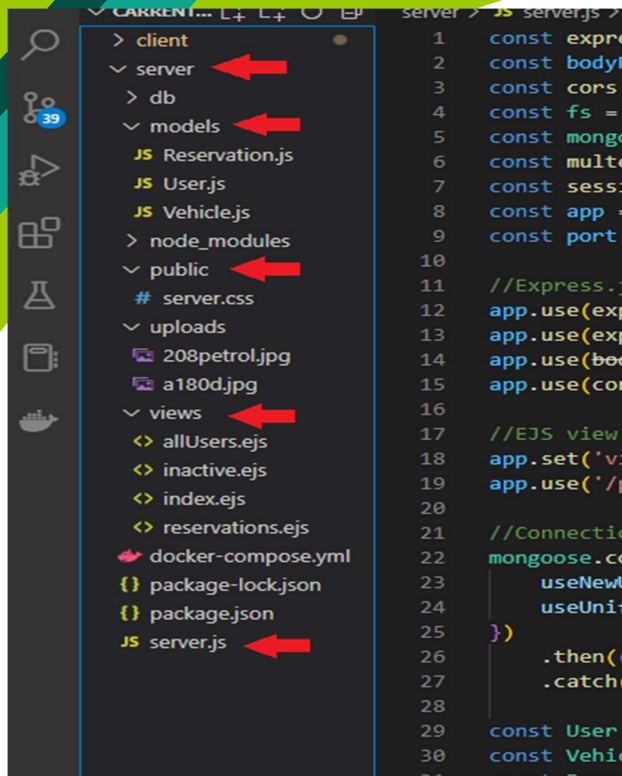
API Routes: Τα routes αυτά δημιουργήθηκαν ώστε ο client να μπορεί να έχει τα απαραίτητα δεδομένα που πρέπει να εμφανιστούν στην οθόνη του χρήστη χωρίς να κάνει post request. Ξεκινώντας με τον στόλο, έχουμε το *app.get('/vehicle/all')* το οποίο αφού αποθηκεύσει σε μεταβλητή όλα τα οχήματα που έκανε query από το MongoDB, θα τα στείλει σαν json στο endpoint που έκανε το request. Το *app.get('/vehicleByReg/:reg')* έχει την ίδια λογική με το προηγούμενο, με την διαφορά όμως ότι χρησιμοποιεί φίλτρο και πρέπει να έχει 1 αποτέλεσμα. Τέλος το *app.get('/reservations/:username')* βρίσκει τις κρατήσεις με φίλτρο τον χρήστη και τις στέλνει με τον ίδιο τρόπο.

CRUD χρήστη: Όταν ένας χρήστης μέσω του client στείλει post request για την δημιουργία λογαριασμού, αυτό θα πάει στο `app.post('/user/new')` όπου αφού τραβήξει όλα τα στοιχεία από το σώμα θα κάνει τους απαραίτητους ελέγχους τύπου να μην υπάρχει ίδιος χρήστης (είτε email είτε username). Σε περίπτωση που ολοκληρωθεί επιτυχώς η διαδικασία τότε θα στείλει το αντίστοιχο status και json με τα στοιχεία του χρήστη, σε αντίθετη περίπτωση θα στείλει το status 422 και το αντίστοιχο μήνυμα λάθους.

Ο διαχειριστής έπειτα αφού μπει να ελέγξει τους λογαριασμούς, αν διαγράψει έναν χρήστη θα σταλεί post request στο `app.post('/user/deleteByUsername')` όπου θα κάνει query με φίλτρο το όνομα χρήστη και μετά θα διαγράψει τον χρήστη. Αν θέλει να αλλάξει ρόλο σε έναν χρήστη, θα σταλεί post request στο `app.post('/user/accept')` όπου θα γίνει query με φίλτρο και να ενημερώσει τον ρόλο του χρήστη.

CRUD οχήματος: Όταν ο διαχειριστής μέσω της σελίδας διαχειριστή στο client στείλει post request για την δημιουργία αυτοκινήτου στο `app.post('/vehicle/new')` το οποίο θα ελέγξει αν ο αριθμός της πινακίδας έχει ξαναχρησιμοποιηθεί, και εφόσον είναι μοναδικός θα δημιουργήσει την νέα οντότητα στην βάση. Όταν στείλει post request για την ενημέρωση των στοιχείων του οχήματος στο `app.post('/vehicle/update/:regnumber')` όπου θα κάνει query σύμφωνα με το φίλτρο και θα ενημερώσει τα στοιχεία από το σώμα του request. Όταν στείλει post request για την διαγραφή ενός οχήματος `app.post('/vehicle/delete/:regnumber')` με regnumber στο url τον αριθμό πινακίδας του αυτοκινήτου, όπου θα κάνει query και διαγραφή της οντότητας του οχήματος.

CRUD κράτησης: Όταν ο χρήστης κάνει κράτηση μέσω του client και στείλει post request στο `app.post('/reservation/new')`, όπου θα δημιουργηθεί μια αίτηση κράτησης εφόσον στο query δεν βρεθεί κράτηση με ίδιο πελάτη την ίδια ημερομηνία και το όχημα που ζητείται υπάρχει. Τέλος, όταν ο διαχειριστής θέλει να απορριψη ή δεχτεί μια κράτηση τότε θα στείλει post request στο `app.post('/reservation/update')`, όπου θα κάνει query για να βρει την κράτηση μέσω του φίλτρου και μετά να τη διαγράψει.



Server main files

Client:

Για την δημιουργία του client σε react στο terminal γράψαμε `npx create-react-app client`. Αυτόματα δημιούργησε διάφορα αρχεία όπως και το initialization του σέρβερ. Το βασικότερο όμως αρχείο είναι το `index.js` όπου βρίσκεται και το client routing. Από εκεί και έπειτα χρησιμοποιούνται τα αρχεία στους φακέλους Components και Screens, τα οποία είναι τύπου `.jsx` και όσα functions κάνουν render στο `/public/index.html`, κάνουν `return <<html>>` κώδικα ο οποίος θα κουμπώσει στο `index.html`. Στο React γενικότερα χρησιμοποιείται πολύ το λεγόμενο State Management, το οποίο έχει να κάνει με τις καταστάσεις των μεταβλητών. Για να κάνουμε state management καλούμε το `import useState` από το React και ύστερα φτιάχνουμε μια `const` μεταβλητή με τον εξής τρόπο: `const [variable, setVariable] = useState('whatever we want: String, bool, int, null, etc.')`. Αυτό μας δίνει την δυνατότητα να αλλάζουμε εύκολα την οθόνη του χρήστη χωρίς να αλλάζουμε σελίδα κάθε φορά που αλλάζει κάτι (χρησιμοποιώντας στο render το conditional ternary operator: `expression1 ? expression2 : expression3`). Επίσης στο συγκεκριμένο project χρησιμοποιείται συχνά το `sessionStorage` για την αποθήκευση χρήσιμων δεδομένων όπως πχ τον χρήστη που είναι συνδεδεμένος. Το `protected-route.jsx` αρχείο έχει δημιουργηθεί για τον έλεγχο του αν ο χρήστης έχει τους κατάλληλους ρόλους για να εισέλθει στο route το οποίο περικλείει το `protected-route`.

NavBar-Homepage

Με το που εισέλθει ο χρήστης στο website, θα βρεθεί στο index route που είναι το homepage ενώ το navbar θα φαίνεται σε όλα τα routes χωρίς να κάνει re-render καθώς είναι parent route. Το navbar χρησιμοποιεί Links για τα υπόλοιπα routes και το Outlet για να κάνουν render εκεί τα υπόλοιπα jsx αρχεία. Επιπρόσθετα το navbar ελέγχει αν ο χρήστης είναι συνδεδεμένος και αν ο ρόλος του είναι διαχειριστή ή όχι και εμφανίζει το αντίστοιχο username και το Link για την σελίδα διαχειριστή εφόσον είναι όντως διαχειριστής.

Fleet

Ο οποιοσδήποτε χρήστης μπορεί να μπει στην σελίδα με τον στόλο. Με το που μπει, το fleet.jsx θα στείλει GET request στον σέρβερ μέσω του `fetch('http://localhost:3001/vehicle/all')` και θα αποθηκεύσει το json με τα οχήματα ώστε να τα προβάλει ένα-ένα μέσω του Vehicle.jsx component. Τα φίλτρα για τον στόλο γίνονται εφικτά μέσω του State Management και του conditional ternary operator που ελέγχει τα φίλτρα real-time.

Register

Ο χρήστης που επιθυμεί να δημιουργήσει έναν λογαριασμό πατάει στο Register Link του NavBar και θα του εμφανίσει μια φόρμα. Στο αρχείο register.jsx χρησιμοποιούμε πολλές μεταβλητές με State Management καθώς πολλές από αυτές χρησιμεύουν στο κομμάτι του real-time check της φόρμας. Επίσης το useEffect χρησιμεύει στην επανάληψη ενός function σε περίπτωση που γίνει κάποια αλλαγή στην μεταβλητή που είναι στο τέλος ως condition πχ `[formValue]`. Έτσι κάθε φορά που ο χρήστης κάνει κάποια αλλαγή στα fields τότε γίνεται και ο ανάλογος έλεγχος σε μερικά από αυτά. Όσον αφορά το Country/City field, κάνουμε get method `fetch('https://countriesnow.space/api/v0.1/countries')` και αποθηκεύουμε το Json με τις πληροφορίες. Όταν ο χρήστης συμπληρώσει τα στοιχεία του και πατήσει register, η φόρμα θα τρέξει το handleSubmit function το οποίο θα κάνει `fetch(http://localhost:3001/user/new)` αλλά με τα εξής στοιχεία: `method: 'POST', headers: { 'Content-Type': 'application/json', 'Accept': 'application/json' }` και body: formValue σε μορφή String.

Login

Ο χρήστης εισάγει τα στοιχεία του και με τον ίδιο τρόπο καλείται η μέθοδος fetch αλλά με διαφορετικό url: `http://localhost:3001/user/login`

Dashboard

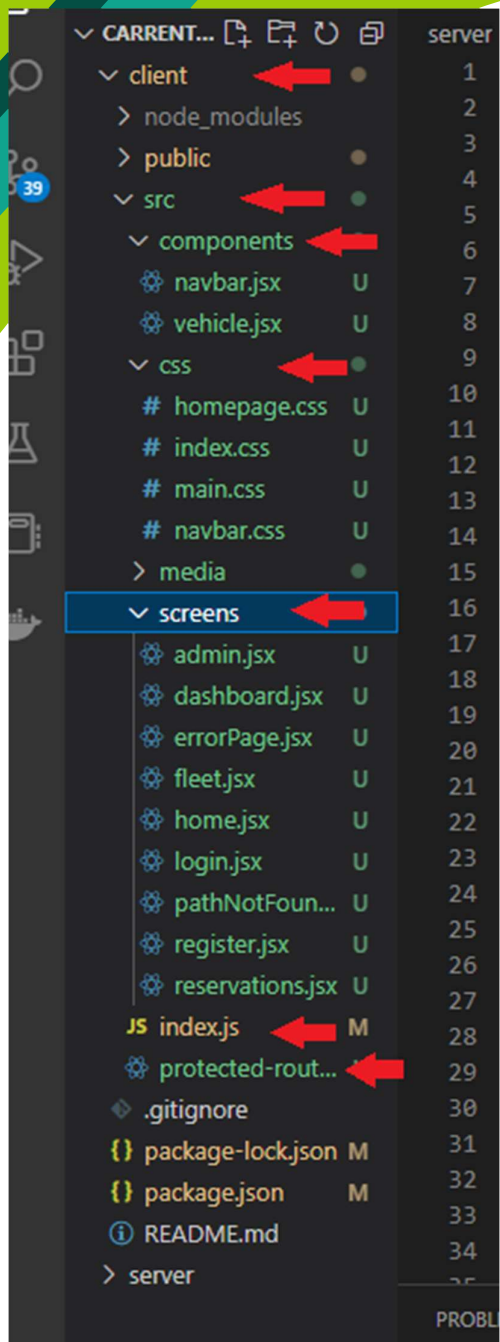
Ο χρήστης που έχει συνδεθεί θα μπορεί να βλέπει τα στοιχεία του και τις κρατήσεις του καθώς το function Dashboard θα κάνει get method `fetch('http://localhost:3001/reservations/${sessionUser.username}')` και ύστερα να αποθηκεύσει τα δεδομένα για να τα προβάλει σε πίνακα.

Reservations

Ο συνδεδεμένος χρήστης που επιθυμεί να κάνει κράτηση θα μπορεί να ελέγξει τον αριθμό πινακίδας του οχήματος που θέλει να νοικιάσει ώστε να είναι έγκυρο. Όταν πατήσει check της πινακίδας, η μέθοδος θα στείλει get method `fetch('http://localhost:3001/vehicleByReg/${formValue.reservedvehicle}')` και θα επιστραφεί το αποτέλεσμα το οποίο θα φανεί με ένα Τικ. Αφού συμπληρώσει και τα υπόλοιπα στοιχεία και πατήσει αίτηση κράτησης, θα καλέσει η φόρμα το `handleSubmit` με `fetch('http://localhost:3001/reservation/new')` με τα ίδια options όπως στο Login και Register.

Admin

Ο συνδεδεμένος χρήστης με τον ρόλο του διαχειριστή, μπορεί να μπει στην σελίδα διαχειριστή και να συμπληρώσει τις φόρμες που χρειάζεται για να προσθέσει, ενημερώσει ή να διαγράψει ένα όχημα. Στην προσθήκη νέου οχήματος αφού συμπληρώσει τα πεδία, θα πατήσει προσθήκη και η φόρμα θα καλέσει την μέθοδο `handleSubmit` για να κάνει `fetch('http://localhost:3001/vehicle/new')` με μέθοδο POST και body το `formData` αντί για το `formValue`, χωρίς headers ώστε να βγούνε αυτόματα από τον Browser και να δέχεται body ως `multipart/form-data`. Η ενημέρωση του οχήματος όταν πατηθεί θα στείλει `fetch('http://localhost:3001/vehicle/update/${updateValue.regnumber}')` με τα ίδια options με το Login, αλλά με την διαφορά ότι θα έχει ως body: `updateValue`. Τέλος, όταν επιλέξει ένα όχημα για να διαγράψει, τότε θα στείλει `fetch('http://localhost:3001/vehicle/delete/${regnumber.licenseplate}', {method: 'POST'})` με το url να συμπεριλαμβάνει τον αριθμό πινακίδας που θέλει να διαγράψει.



Client files

Εγχειρίδιο Διαχειριστή

First-time setup:

Step 1: Make sure you have Node.js installed (check from console with `node -version`)

Step 2: Download and install Docker-Desktop for the containerized DataBase

Step 3: Download and install MongoDB Compass

Step 4: Use Visual Studio Code that allows to have 2 or more terminals open at the same time

Step 5: Open the project folder in VS Code and open 2 terminals (might need to open as workspace if next step doesn't work properly)

Step 6: On first terminal type: `cd server`, then type: `docker-compose up` and lastly type: `npm start`

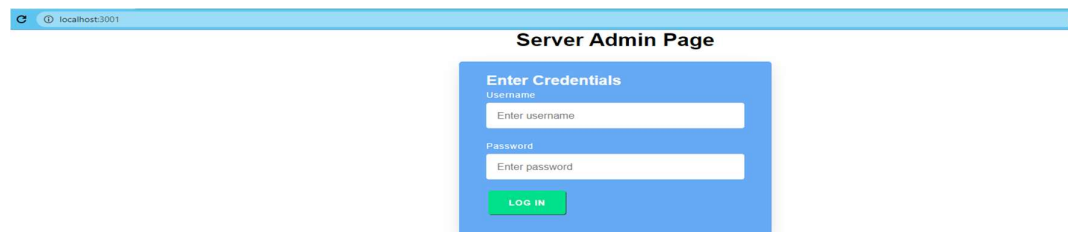
Step 7: On second terminal type: `cd client` and then type: `npm start`

Step 8: Now that Docker is running and container is composed, MongoDB should be working and from MongoDB Compass the collections should be added while using MongoDB url: `mongodb://root:root@localhost:27017/CarRentalWebApp?&authSource=admin`

Εγχειρίδιο χρήστη

Admin Usage:

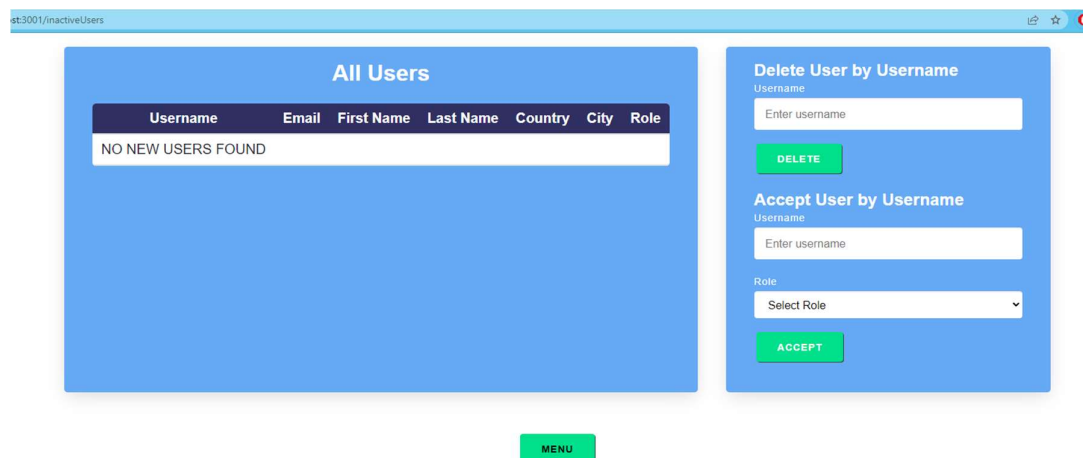
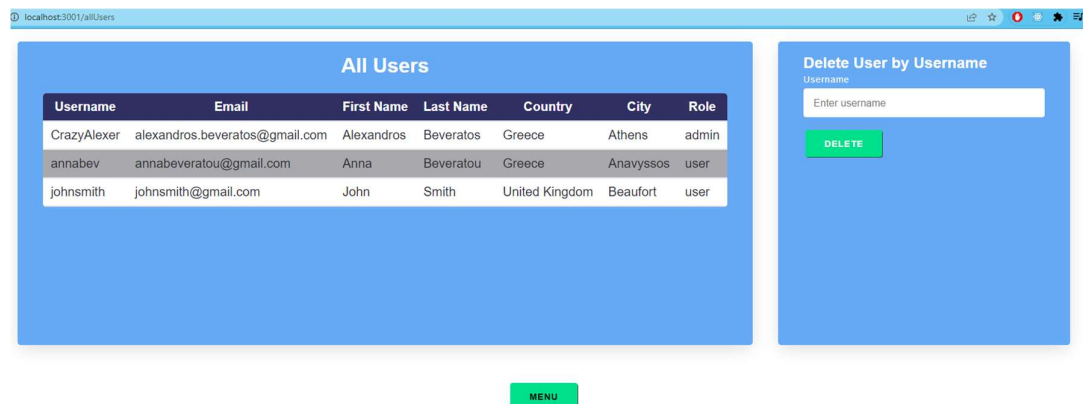
For **Users** and **Reservations** the admin has to log in on this page: `http://localhost:3001/`



After entering the correct credentials, the main menu will show up:



From there the admin can choose to look up and modify Users, new Registrations and Reservations:



The screenshot shows a web browser window with the URL `localhost:3001/reservations`. The main content area is titled "All Reservations" and contains a table with the following data:

| Username | Email | Phone | License Plate | From Date | To Date | Price | Status |
|-------------|--------------------------------|----------|---------------|------------|------------|-------|----------|
| CrazyAlexer | alexandros.beveratos@gmail.com | 69325235 | IKM 4935 | 2023-02-16 | 2023-02-24 | 207 | rejected |
| johnsmith | johnsmith@gmail.com | 69325235 | IKM 4935 | 2023-02-23 | 2023-02-28 | 138 | accepted |

On the right side, there is a sidebar titled "Update Reservation Status" with the following form fields:

- Username:
- License Plate:
- Reservation Status:
- SUBMIT button

Below the browser window, there is a green button labeled "MENU".

For **Fleet** management the admin has to be logged in from an account with admin role on:

<http://localhost:3000/>

After that, select the admin link on the Navigation Bar

The screenshot shows a web browser window with the URL `localhost:3000/`. The navigation bar includes the following links: **RENT WHEELZ**, Home, Dashboard, Fleet, Reservations, **Admin** (highlighted), CrazyAlexer, and Log Out.

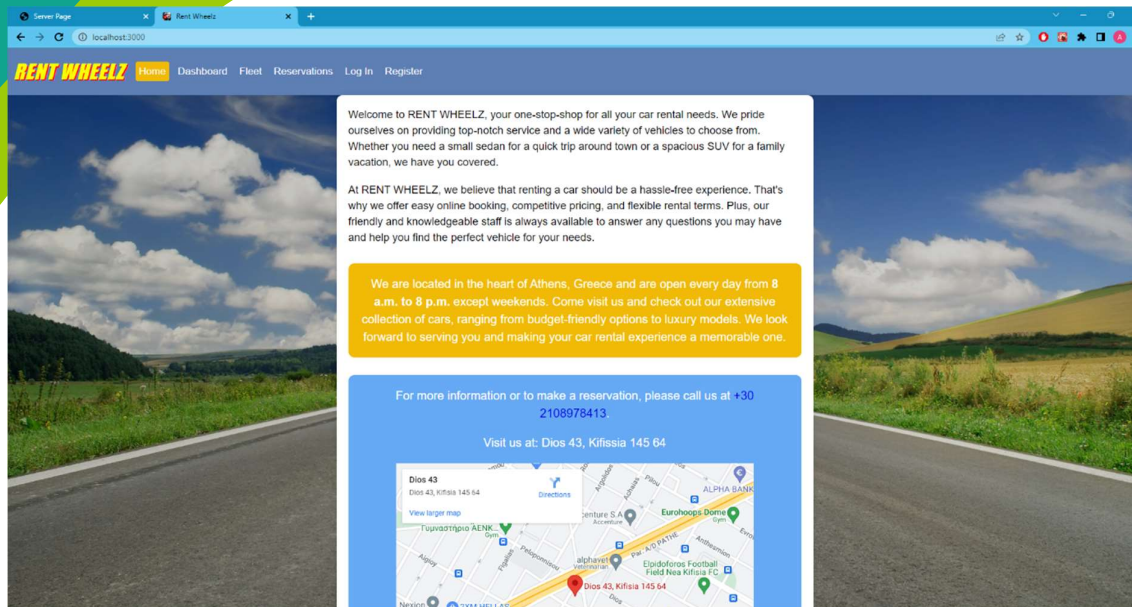
The main content area is titled "Admin Page" and contains the text "Here you can add or remove vehicle listings". There are three main sections:

- Add Vehicle**:
 - Vehicle Manufacturer:
 - Vehicle Model:
 - Engine Type:
 - Engine Size CC:
 - Seats:
 - Availability:
 - Standard Price per Day:
 - Registration Number:
- Update Vehicle**:
 - Engine Type:
 - Engine Size CC:
 - Seats:
 - Availability:
 - Standard Price per Day:
 - Registration Number:
 - UPDATE button
- Delete Vehicle**:
 - Vehicle Registration Number: (with a red X icon)
 - CHECK button
 - DELETE button

And now forms can be filled and executed.

General usage:

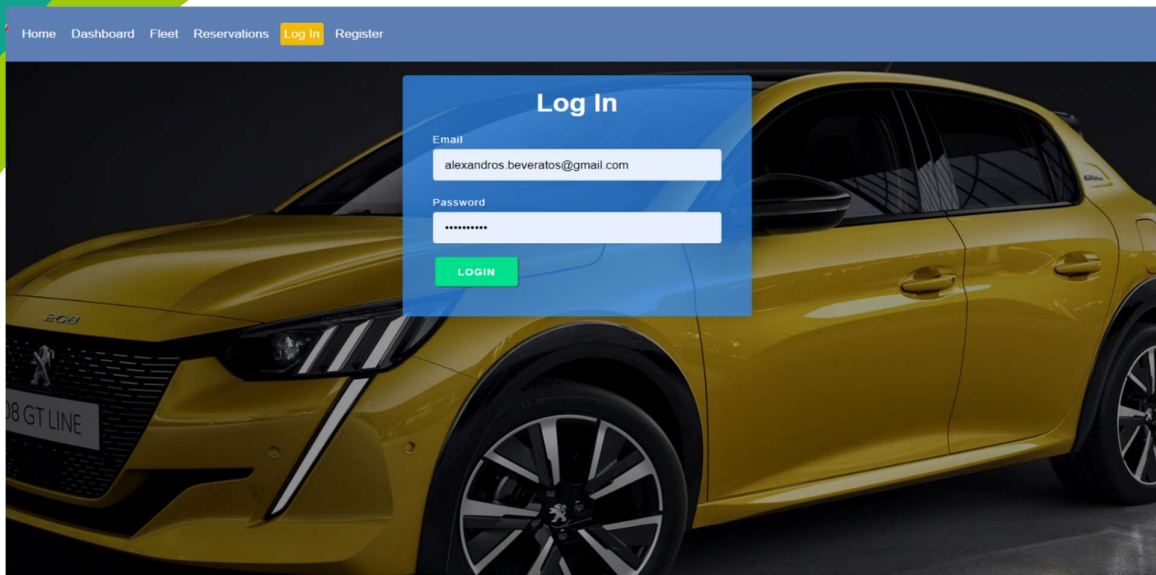
In order to use the website the user must go to the url: <http://localhost:3000/>



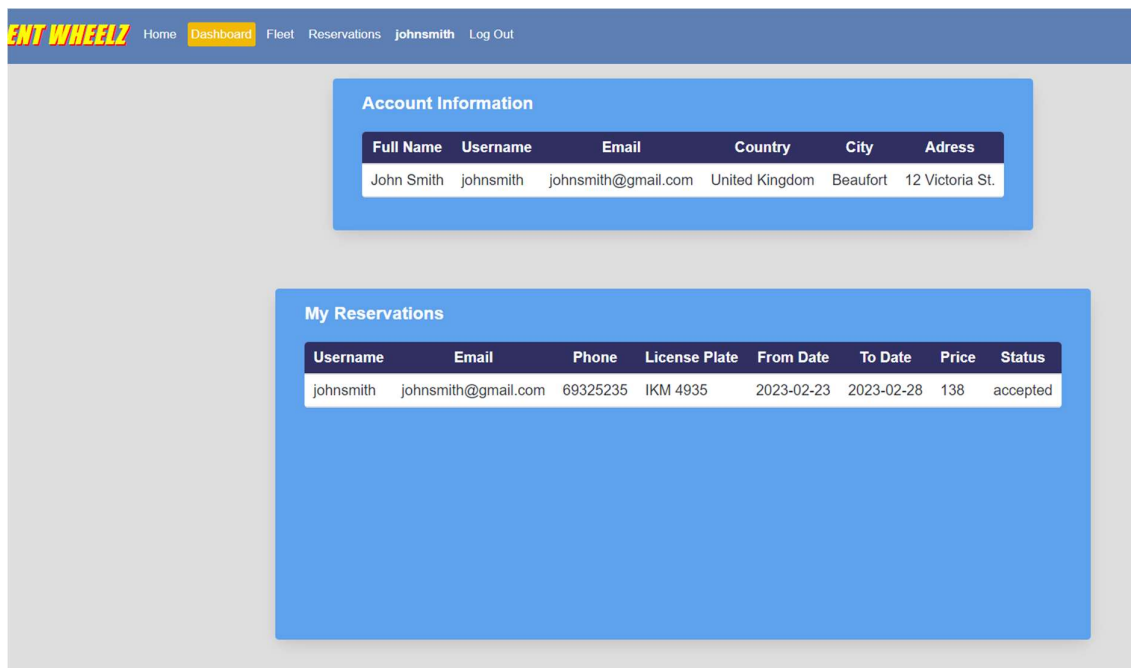
The homepage will show up with a navigation bar that will help locate all the needed services. Users without an account must click on Register in order to make an account creation request.

A screenshot of the 'Create an Account' form on the RENT WHEELZ website. The form is overlaid on a background image of a yellow Peugeot 208 GT LINE. The form fields include: First Name (Alexandros), Last Name (Beveratos), Country (Greece), City (Athens), Address (Rodopoleos 17, 17289), Username (alexbev), Email (alexandros.beveratos@gmail.com), and Password (masked with asterisks). A green 'SIGN UP' button is at the bottom. A note states: 'An admin will approve your request shortly after'. A password requirement note at the bottom says: '8 to 24 characters. Must include uppercase and lowercase letters, a number and a special character. Allowed special characters: ! @ # % %'.

After entering all the necessary fields and clicking sign up without an error, the user will be redirected to login page. If the user's account hasn't been reviewed yet, the message will show up as an error message.



After logging in successfully the user will be redirected to the Dashboard.

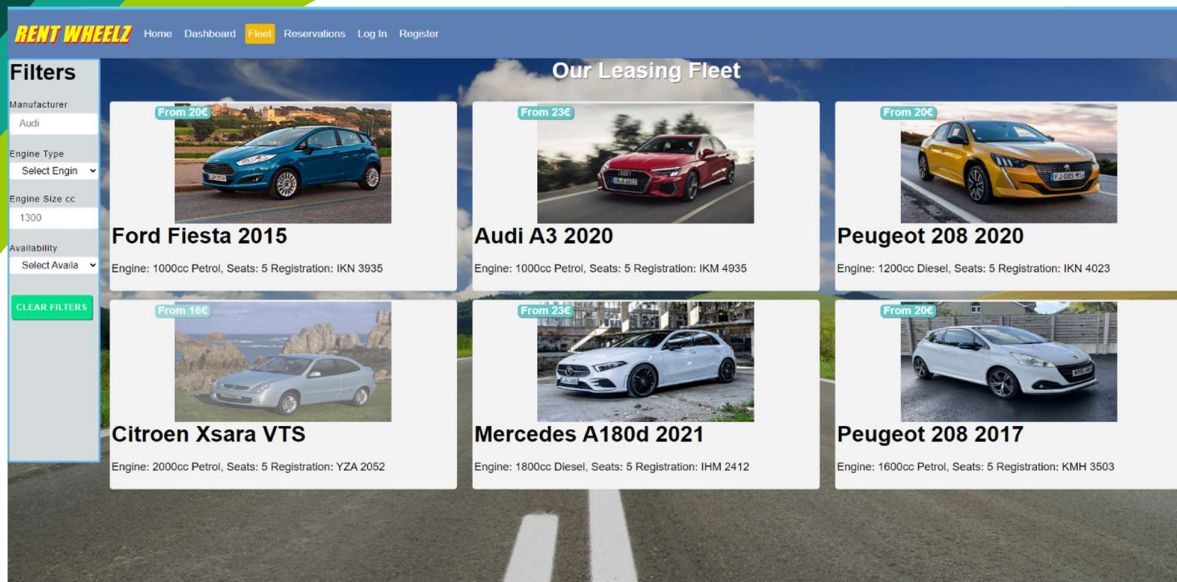


| Full Name | Username | Email | Country | City | Address |
|------------|-----------|---------------------|----------------|----------|-----------------|
| John Smith | johnsmith | johnsmith@gmail.com | United Kingdom | Beaufort | 12 Victoria St. |

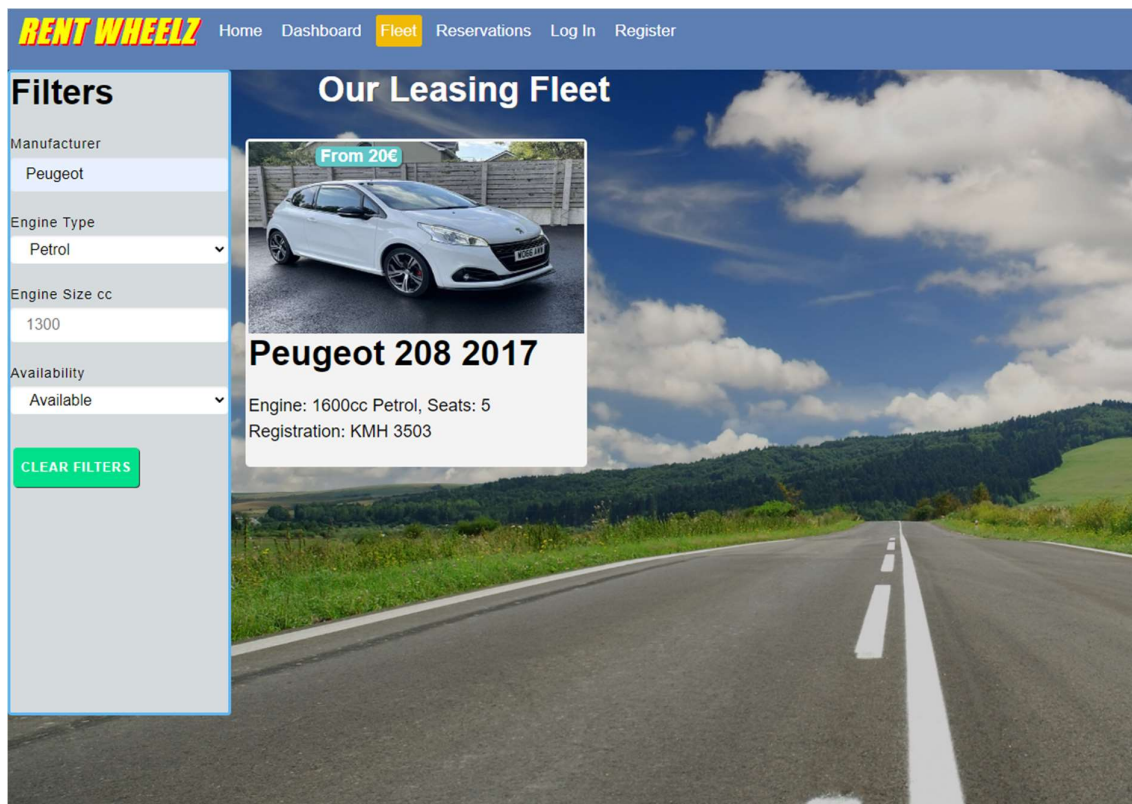
| Username | Email | Phone | License Plate | From Date | To Date | Price | Status |
|-----------|---------------------|----------|---------------|------------|------------|-------|----------|
| johnsmith | johnsmith@gmail.com | 69325235 | IKM 4935 | 2023-02-23 | 2023-02-28 | 138 | accepted |

Here the user can see all the information regarding his account.

Any user even without logging in can go to the Fleet page to look for vehicles.



Also the user can use the Filters sidebar to find a specific vehicle. (In image below only Petrol Peugeot that is available today Shows up instead of both)



After concluding on which vehicle the user wants to make a reservation request, he can go to reservation page after copying the registration plate from the fleet page in order to add it to the reservation form.

ELZ Home Dashboard Fleet **Reservations** johnsmith Log Out

Reservations

Make a reservation

Vehicle Registration Number ✓
IKM 4935 **CHECK**

Choose Date From
23-Feb-2023

Choose Date To
28-Feb-2023

Phone Number
69325235

Minimum price: 138€ **RESERVE**

Make a reservation request and we will contact you about the result of your request