SettleCar API Documentation

Users resource (**/api/users**)

Register User

It will create a new user for the website with the given username, email, phone, password and account type

Postcondition: A new user is created with the sent information

/api/users/ **(post)**

Body:

```
{
    "name": "Albert",
    "phone": "938 421 674",
    "email": "test@mail.com",
    "pass": "123",
    "type": "1"
}
```

name (mandatory): Name of the user.
phone(mandatory): Phone of the user
email (mandatory): Email of the user. Must be different from other existing users
pass (mandatory): Password of the user.
type (mandatory): User type.

Success (200):

```
{
```

```
  "msg": "Registered! You can now log in."
}
```


Errors:
500: Server Error
400: The email exists:


```
[  {  "location": "body",  "param": "email",  "msg": "That email already exists"  }  ]
```


Example of usage:

```
 const response = await fetch(`/api/users/`,
{
    headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    },
  method: "POST",
  body: JSON.stringify({
      name: name,
      phone: phone,
      email: email,
      pass: pass,
      type: check
  })
});
```

Authenticate User

User sends the email and password to authenticate. If the login information is valid a token is stored in the database and in a cookie session, if it is not valid an error message is returned.

Postcondition: A cookie is saved on the client side with the authentication token that identifies the user in the next requests.

/api/users/auth **(post)**

Body:


```
{
  "email": "test@mail.com",
  "pass": "123"
}
```


email (mandatory): email of the user

password (mandatory): Password for the user

Success (200):

```
{
   "msg": "Successful Login!"
}
```

Errors:
500: Server Error
401: Authentication Failed

```
{  "msg": "Wrong email or password!"  }
```

Example of usage:

```
const response = await fetch(`/api/users/auth`,
        {
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
          method: "POST",
          body: JSON.stringify({
              email: email,
              pass: pass
          })
        });
```

Get User Profile

Sends user information of the authenticated user

Precondition: The user must be authenticated
Note: Authenticated users send a cookie with the authentication token so any authenticated user can be identified in every request

/api/users/auth **(get)**

Success (200):

```
{
   "name": "Albert"
   "phone": "938 421 674"
   "email": "test@mail.com"
   "type": "1"
```

}

Errors:
500: Server Error
401: Authentication Failed
{ "msg": "Please log in" }

Example of usage:
```
const response = await fetch(`/api/users/auth`);
```

Logout User

Logs out authenticated user

Precondition: The user must be authenticated
Postcondition: User is no longer authenticated (the cookie with the token is removed)

/api/users/auth **(delete)**

Success (200):
{ "msg": "User logged out!" }

Errors:
500: Server Error
401: Authentication Failed
{ "msg": "Please log in" }

Example of usage:
```
const response = await fetch(`/api/users/auth`,
        {
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
          method: "DELETE",
        });
```

Car resource (**/api/car**)

Get Owner's cars

Sends all the cars owned by a user
Precondition: The user must be authenticated, and must be a owner account

/api/car/auth **(get)**

Success (200):


```
{
   "id": "1t"
   "model": "Volkswage"
   "brand": "Polo"
   "licenseplate": "42-AG-99"
   "car_state": "available"
   "rent": "42"
}
```


Errors:
500: Server Error
400: No cars
{ "msg": "This user has no registered cars" }

Example of usage:

```
const response = await fetch(`/api/car/auth`);
```

Delete a car from the owner

Deletes the given car from the database

Precondition: The user must be authenticated and the car must belong to the owner

/api/car/auth **(delete)**

Success (200):
{}

Errors:
500: Server Error
400: The car doesn't belong
{ "msg": "This car does not belong to the user" }

Example of usage:

```
const response = await fetch(`/api/car/auth`,
        {
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
```

```
        },
      method: "DELETE",
      body: JSON.stringify({
          licenseplate: license
      })
    });
```

Get car

Sends all the information of a car

Postcondition: User must be logged in and be the owner of the car

/api/car/auth/:carid **(get)**

Body:

```
params{
  car_id:2,
}
```

Success (200):

```
{
    id: 2,
    licenseplate: '23-GA-21',
    brand: 'Audi',
    model: 'A4',
    year: '2001',
    bhp: '130hp',
    engine: '1.9TDI',
    fuel: 'Diesel',
    gearbox: '5M',
    drivetrain: 'FWD',
    doors: '5',
    seats: '5',
    bootcapacity: '490L',
    extra_equipment: 'AC;Heated Seats;Bluetooth Radio',
    price_day: '250',
    car_state: 'available',
    user_id: 2,
    images: [
      'exemplo_link‘,
      ‘exemplo_link',
```

]
}

Errors:
500: Server Error
400: The car doesn't belong
{ "msg": "This car does not belong to the user" }

Example of usage:

```
const response = await fetch(`/api/car/auth/${carid}`);
        var result = await response.json();
        return { successful: response.status == 200,
                 cars: result};
```

Search all cars

Sends basic information of all available cars

/api/car **(get)**

Success (200):

```
{
 cars:{
     brand: 'Audi',
     model: 'A4',
     year: '2001',
     rent: '250',
     image: [
       'exemplo_link',
   ]},{..}
 }
```

Errors:
500: Server Error

Example of usage:

```
const response = await fetch('/api/car/');
        var result = await response.json();
        return { successful: response.status == 200,
                 cars: result.result};
```

Search cars by date

Sends basic information of all cars available for rent in a specific adate

/api/ca/search/:stardate/:returndate **(get)**

Body:

params{

start_date:'2023-08-1',

return_date:'2023-08-9'

}

Success (200):

```
{
 cars:{
      brand: 'Audi',
      model: 'A4',
      year: '2001',
      rent: '250',
      image: [
        'exemplo_link',
    ]},{..}
 }
```

Errors:

500: Server Error

Example of usage:

```
const response = await
fetch(`/api/car/search/${start_date}/${return_date}`);
        var result = await response.json();
        return { successful: response.status == 200,
                 cars: result};
```

Auth Middleware

verifyAuth

A middleware function that can be used to check for the authentication token.

Success:

Requires: A valid authentication token in the cookie
Actions:

- Sets **req.user** property with an object with the user information:

```
{ "id": 1 , "name": "Albert" }
```

NOTE: The information includes the ids of the entities. In many cases you will want to hide these ids if you return the objects to the client.

Errors:
500: Server Error
401: No authentication token or invalid token:

```
{    "msg": "Please log in" }
```