

# SettleCar API Documentation

## [Users resource \( /api/users \)](#)

[Register User](#)

[Authenticate User](#)

[Get User Profile](#)

[Logout User](#)

[User Occupied](#)

## [Car resource \( /api/car \)](#)

[Add a car](#)

[Update location](#)

[Update location](#)

[Get Owner's cars](#)

[Car availability](#)

[Delete a car from the owner](#)

[Get a car information for renting](#)

[Get ALL information of a car](#)

[Search all cars](#)

[Search cars by date](#)

## [Rent resource \( /api/rent \)](#)

[Post a rent](#)

[Last rent](#)

[Cancel Rent](#)

[Rent route](#)

[Rent history from a car](#)

[Rent history from a User](#)

[Scheduled rents from a car](#)

[Verify Rent Route](#)

## [Auth Middleware](#)

[verifyAuth](#)

## 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"  
}
```

**Every param is obligatory**

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",});
```

### **User Occupied**

returns true or false depending on if the user is occupied with a rent or not

**Precondition:**User needs to be logged in

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

Success (200):

```
{  
  "occupied": false  
}
```

Errors:

500: Server Error

Example of usage:

```
const response = await fetch(`/api/users/auth/isOccupied`);  
var result = await response.json();  
console.log(result);  
return { successful: response.status == 200,  
          unauthenticated: response.status == 401,  
          user: result};
```

## Car resource ( /api/car )

### **Add a car**

adds a car to the database

**Precondition:** The user must be authenticated, and must be an admin account

/api/car/auth (post)

Body:

```
{  license: '23-GA-21',
    brand: 'Audi',
    model: 'A4',
    year: '2001',
    bhp: '130hp',
    engine: '1.9TDI',
    fuel: 'Diesel',
    gearbox: '5M',
    drivetrain: 'FWD',
    door_n: '5',
    seat_n: '5',
    bootcapacity: '490L',
    priceday: '250',
    "extras": [
      "AC",
      "Heated Seats"
    ],
    "services": {
      "inspection": "2023-05-28",
      "insurance": "2023-05-30"
    },
    images: [
      'exemplo_link',
      'exemplo_link',
    ]
}
```

**all parameters are obligatory**

Success (200):

```
{
  "status":200
}
```

Errors:

500: Server Error

400: there is an error for each parameter not being correct

Example of usage:

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

### **Update location**

Updates the car's location on the database

/api/car/updateloc/:carid **(patch)**

Params:

car id

Body:

```
{
  "loc":{
    "lat":"38.629650533269675"
    "lon":"-8.91365651088844"
  }
}
```

**all parameters are obligatory**

Success (200):

```
{
  "status": 200,
  "msg": "updated successfully"
}
```

Errors:

500: Server Error

### Update location

Updates the car's location on the database

/api/car/updateservice/:carid **(patch)**

Params:  
car id

Body:  
{ date: '2024-01-25T00:00:00.000Z', name: 'inspection' }

**all parameters are obligatory**

Success (200):

```
{
  "status": 200,
  "msg": "updated successfully"
}
```

Errors:  
500: Server Error

### 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`);
```



### Car availability

Sends the information about the car to populate the calendar

**Precondition:** The user must be authenticated

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

Body:

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

Success (200):

```
{  
  "rents":  
  [  
    {  
      "beginning":"2023-08-02T23:00:00.000Z",  
      "end":"2023-08-11T23:00:00.000Z"  
    }  
  ],  
  "max":"2023-12-04T00:00:00.000Z"  
}
```

Errors:

500: Server Error

Example of usage:

```
const response = await fetch(`/api/car/auth/availability/${carid}`);
```

### **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 a car information for renting

Returns only the information required for rents

/api/car/:id **(delete)**

Body:  
params{  
 car\_id:2,}

Success (200):

```
{
  "id": 1,
  "licenseplate": "",
  "brand": "Volkswagen",
  "model": "Passat",
  "year": "2001",
  "bhp": "130hp",
  "engine": "1.9TDI",
  "fuel": "Diesel",
  "gearbox": "5M",
  "drivetrain": "FWD",
  "doors": "5",
  "seats": "5",
  "bootcapacity": "495L",
  "extra_equipment": "AC;Heated Seats;Bluetooth Radio",
  "price": "120",
  "state": "available",
  "user_id": "",
  "images": [
    "https://media.discordapp.net/attachments/1094210400469397614/1094210455838408764/Passa_image1.jpg?width=1246&height=701",
    "https://media.discordapp.net/attachments/1094210400469397614/1094210456354295868/Passat_image2.jpg?width=1246&height=701"
  ],
  "car_state": ""
}
```

Errors:

500: Server Error

400: The car doesn't belong

```
{ "msg": "This car does not exist" }
```

Example of usage:

```
const response = await fetch(`/api/car/${id}`);
```

### Get ALL information of a 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}`);
```

### **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/');
```

### **Search cars by date**

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

/api/car/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:

```
fetch(`/api/car/search/${start_date}/${return_date}`);
```

## Rent resource ( `/api/rent` )

### **Post a rent**

creates a rent

**Precondition:** The user must be authenticated

`/api/rent/auth`(**post**)

Body:

```
body{  
  car:'2',  
  beginning:'2023-08-1',  
  end:'2023-08-9'  
}
```

Success (200):

```
{  
  status: 200,  
  "msg": "Rent registered successfully"  
}
```

Errors:

500: Server Error

Example of usage:

```
const response = await fetch(`/api/rent/auth`,  
  {  
    headers: {  
      'Accept': 'application/json',  
      'Content-Type': 'application/json'  
    },  
    method: "POST",  
    body: JSON.stringify({  
      beginning: beginning,  
      end: end,  
      car: car_id  
    })  
  });
```

### **Last rent**

returns the last rent scheduled

**Precondition:** The user must be authenticated

/api/rent/getScheduled(**get**)

Success (200):

```
{
  "vehicle": "Volkswagen Passat (2001)",
  "id": 5,
  "start_date": "2023-02-01T00:00:00.000Z",
  "end_date": "2023-02-04T00:00:00.000Z",
  "price": "(360€ +250  ",
  "status": "Finished"
}
```

Errors:

500: Server Error

Example of usage:

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



## **Cancel Rent**

Cancels a scheduled rent

**Precondition:** The user needs to be authenticated and have a rent scheduled

/api/rent/auth/delete **(delete)**

Body:

```
{  
  "rentid": 2,  
}
```

Success (200):

```
{  
  "msg": "Rent deleted sucessfully."  
}
```

Errors:

500: Server Error

403: Rent already started

```
[ { "location": "body", "param": "rents", "msg": "You cant cancel a rent that has already started" } ]
```

404: The rent doesn't exist:

```
[ { "location": "body", "param": "rents", "msg": "this rent doesnt exist" } ]
```

Example of usage:

```
const response = await fetch(`/api/rent/auth/delete/`,  
  {  
    headers: {  
      'Accept': 'application/json',  
      'Content-Type': 'application/json'  
    },  
    method: "DELETE",  
    body: JSON.stringify({  
      rentid: rentid  
    })  
  })  
});
```

### **Rent route**

Returns the route made by the car on a rent

**Precondition:** The user needs to be authenticated and must be an admin

/api/rent/auth/getCourseOwner/:rentId/:date **(get)**

Body:

```
params{
  rentId:"2"
  date:"2022-02-06"
}
```

Success (200):

```
{
  "type": "FeatureCollection",
  "features": [
    ... ],}
```

Errors:

500: Server Error

Example of usage:

```
const response = await fetch(`/api/rent/auth/getCourseOwner/${rentid}/${day}`);
```

### Rent history from a car

**Precondition:** The user needs to be authenticated

/api/rent/auth/auth/history/fromCar/:carid (get)

Body:  
params{  
 carid:"2"  
}

Success (200):  
[{  
 "id":5,  
 "beginning":"2023-02-01T00:00:00.000Z",  
 "end":"2023-02-04T00:00:00.000Z",  
 "usr":"driver",  
 "price":"(360€)+250"  
}]

Errors:  
500: Server Error

Example of usage:

```
const response = await fetch(`/api/rent/auth/history/fromCar/${carid}`);
```

### Rent history from a User

**Precondition:** The user needs to be authenticated

/api/rent/auth/auth/scheduled/fromUser (get)

Success (200):  
[{"id":5,  
 "beginning":"2023-02-01T00:00:00.000Z",  
 "end":"2023-02-04T00:00:00.000Z",  
 "price":"(360€)+250 ",  
 "vehicle":"Volkswagen Passat (2001)"  
}]

Errors:  
500: Server Error

Example of usage:

```
const response = await fetch(`/api/rent/auth/history/fromUser/`);
```

## Scheduled rents from a car

**Precondition:** The user needs to be authenticated

/api/rent/auth/auth/scheduled/fromCar/:carid (**get**)

Body:  
params{  
 carid:"2"  
}

Success (200):  
[  
 {  
 "id":1,  
 "beginning":"2023-08-02T23:00:00.000Z",  
 "end":"2023-08-11T23:00:00.000Z",  
 "price":"(1080€) +null"  
 }  
]

Errors:  
500: Server Error

Example of usage:

```
const response = await fetch(`/api/rent/auth/scheduled/fromCar/${carid}`);
```

### Verify Rent Route

verifies the route a car did in a rent

/api/rent/auth/auth/verify (get)

```
Body{  
  rentid:"2"  
}
```

Success (200):  
{status:200,result:{'msg': 'User has to pay an adicional: \$penalty }  
Success (200):  
result:{'msg':'No penaltys found'}

Errors:  
500: Server Error  
400: no routes in the rent

Example of usage:

```
const response = await fetch(`/api/rent/auth/verify`,{  
  method: 'PATCH',  
  body: JSON.stringify({  
    rentid: rentid,  
  }),  
  headers: {  
    'Content-type': 'application/json; charset=UTF-8',  
  },})
```

## 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" }
```