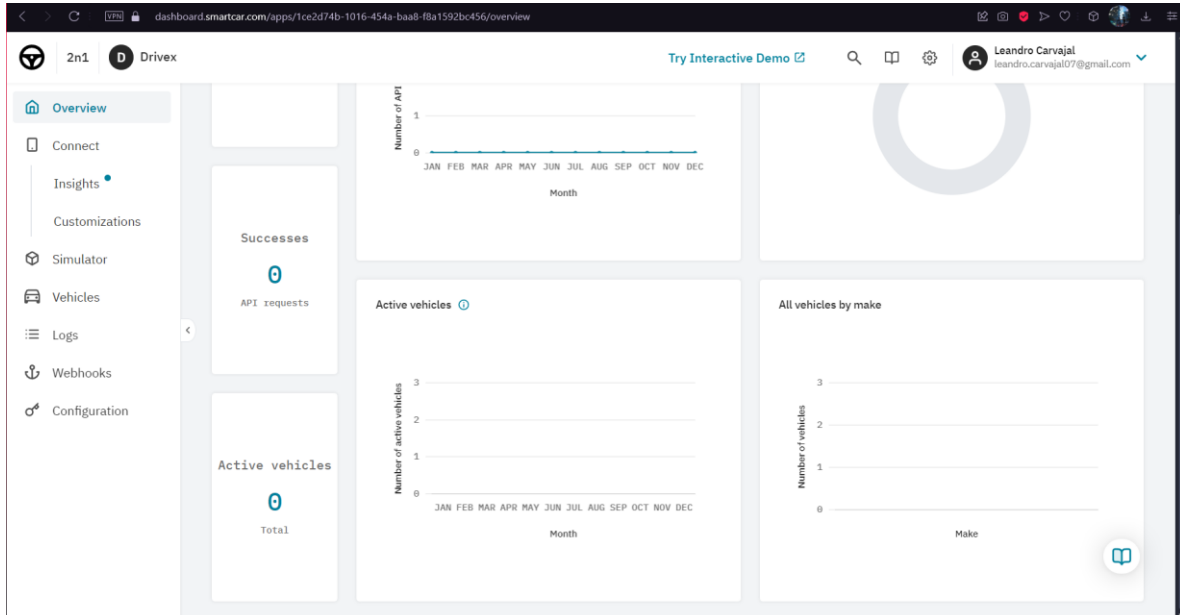


Reporte de pruebas de Smart Car API

I. Ingreso al dashboard de Smart Car. E información OAuth.



The screenshot shows the 'Configuration' page of the Smart Car API dashboard. The left sidebar is the same as the overview page. The main content area is titled 'OAuth' and contains the following fields:

- CLIENT ID:** A text field containing '1ce2d74b-1016-454a-baa8-f8a1592bc456' with a 'Copy' button.
- CLIENT SECRET:** A text field containing a masked secret with a 'Regenerate' button.
- REDIRECT URIS:** A list of redirect URIs with 'Copy' and 'Delete' buttons for each.
- Save:** A button to save the configuration.

II. Simulación de un vehículo compatible con la API, el cual presenta credenciales simuladas para probar la autenticación.

The screenshot shows the 'Vehicle Simulator' dashboard. On the left is a sidebar with navigation links: Overview, Connect, Insights, Customizations, Simulator (active), Vehicles, Logs, Webhooks, and Configuration. The main content area has a header with '2n1', 'D Drivex', a 'Try Interactive Demo' link, search, and user profile 'Leandro Carvajal'. Below the header is a table with columns 'VEHICLE', 'TRIP TYPE', and 'TRIP STATE'. It lists two vehicles: '2021 Tesla Model 3' (US) with 'Long Trip' and '2022 Tesla Model 3' (US) with 'Day Commute', both in 'In progress' state. Below the table are two boxes: 'Simulator Documentation' with a 'Read the docs' link, and 'API Reference' with a 'Check it out' link. A disclaimer at the bottom states: 'All logos and brand names are property of their respective owners, and the use does not imply endorsement of or affiliation with Smartcar.'

VEHICLE	TRIP TYPE	TRIP STATE
2021 Tesla Model 3 US	Long Trip	In progress
2022 Tesla Model 3 US	Day Commute	In progress

Simulator Documentation
Learn about when and how to use the Vehicle Simulator feature while testing your application.
[Read the docs](#)

API Reference
Everything you need to get started using Smartcar's APIs, from endpoints and SDKs to error handling guides.
[Check it out](#)

All logos and brand names are property of their respective owners, and the use does not imply endorsement of or affiliation with Smartcar.

This screenshot shows the detailed view of a vehicle simulation for a '2021 Tesla Model 3'. The VIN is '1SCP6V7N04TH13TE' and the Vehicle ID is '46854af3-a63c-4b83-accb-94d5b7f984d'. A 'Connect Credentials' button is visible. The 'Trip Progress' section features a timeline with five stages: 1. Parked (06:00), 2. Driving (07:00), 3. Plugged in (10:30), 4. Driving (16:30), and 5. Parked (20:00). Below the timeline are four summary cards: 'Long Trip (8 hours)' (TRIP TYPE), 'Driving' (VEHICLE STATE), '1 hr 59 min' (TRIP TIME ELAPSED), and '88.5 kilometers' (DISTANCE DRIVEN). The 'Current Vehicle State' section displays three metrics: 'EV battery level' (72% remaining, 231.5 km range), 'Odometer' (55088.5 km distance), and 'Tire pressure' (219.3 kPa for both back left and back right). Controls for 'Unit System' (Metric) and 'Response Latency' (Realistic) are at the bottom right, along with a 'View average API request latencies' link.

2021 Tesla Model 3
VIN: 1SCP6V7N04TH13TE | Vehicle ID: 46854af3-a63c-4b83-accb-94d5b7f984d | [Connect Credentials](#)

Trip Progress

Parked (1) 06:00 → Driving (2) 07:00 → Plugged in (3) 10:30 → Driving (4) 16:30 → Parked (5) 20:00

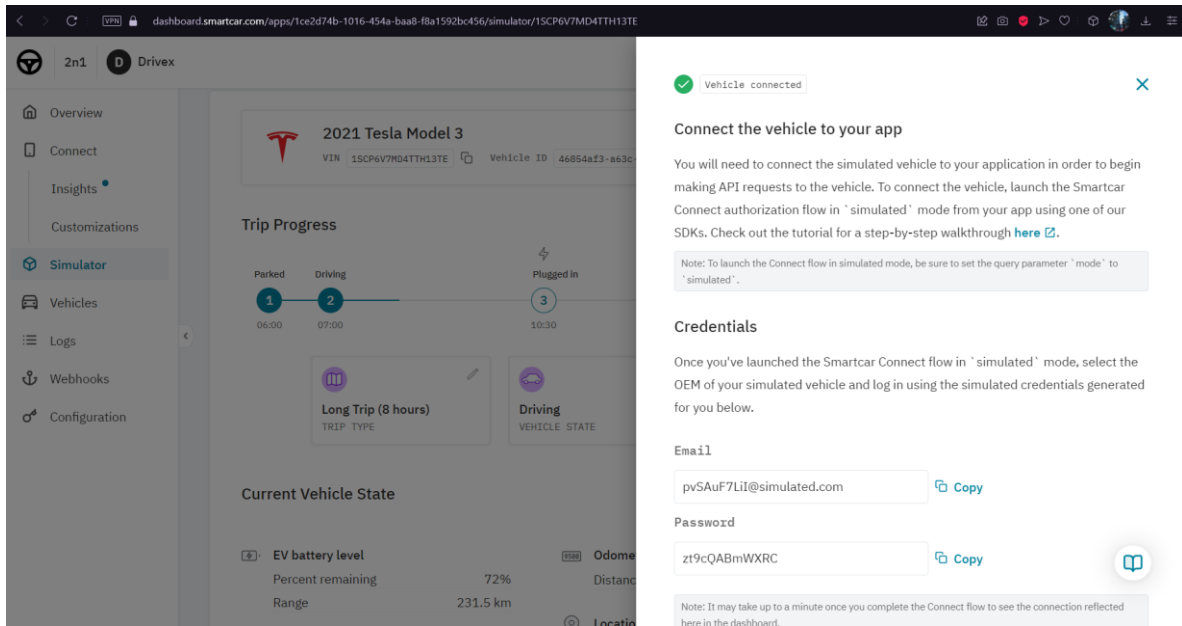
Summary Cards:

- Long Trip (8 hours) - TRIP TYPE
- Driving - VEHICLE STATE
- 1 hr 59 min - TRIP TIME ELAPSED
- 88.5 kilometers - DISTANCE DRIVEN

Current Vehicle State

- EV battery level:** 72% remaining, 231.5 km range
- Odometer:** 55088.5 km distance
- Tire pressure:** 219.3 kPa (Back left, Back right)

Unit System: Metric | Response Latency: Realistic | [View average API request latencies](#)

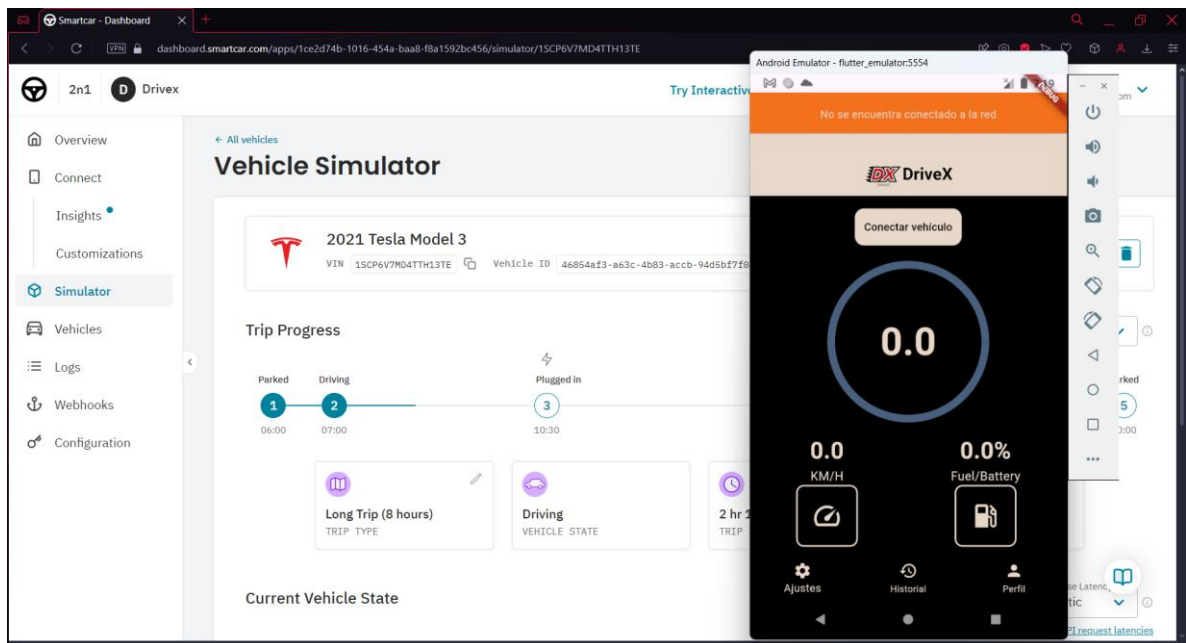


III. Con aquello ya revisado se puede comenzar las pruebas en la aplicación desarrollada en Flutter.

1. Prueba de ingreso a la autenticación Smart Car.

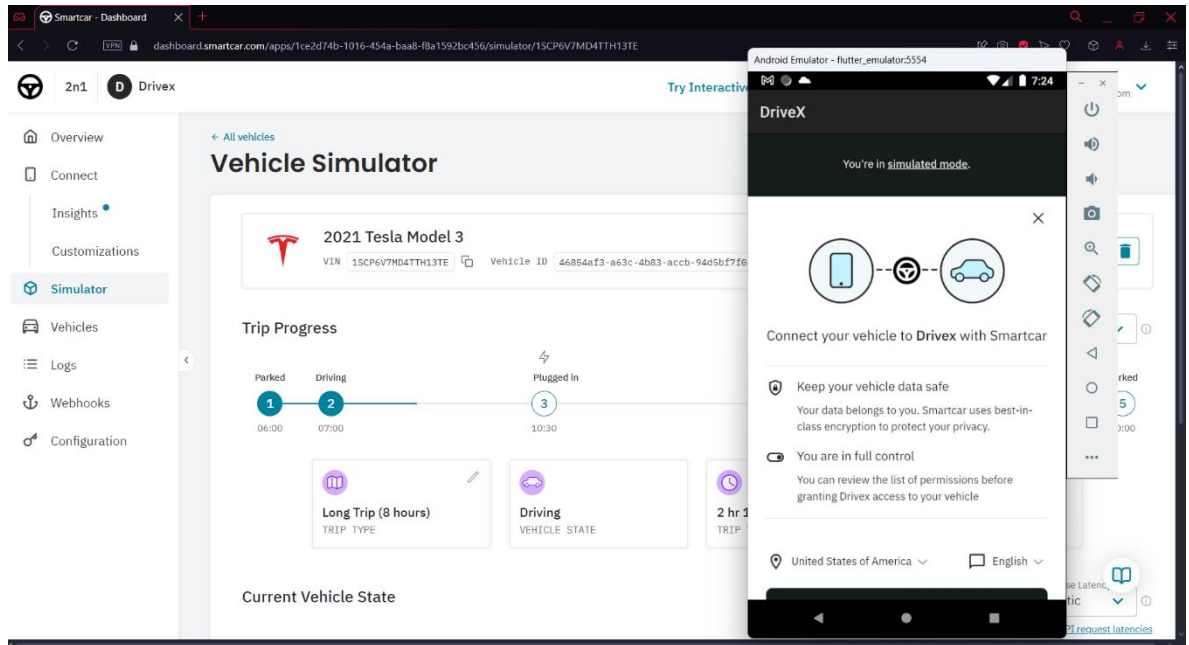
1.1 Sin conexión a internet.

- Resultado esperado: No se inicia el proceso de autenticación.
- Resultado obtenido:



1.2 Con conexión a internet.

- Resultado esperado: Ingreso a la interfaz de autenticación.
- Resultado obtenido:



- Código implementado para el requerimiento:

```
Future<void> _connectVehicle() async
{
  if(APIRequestHelper.instance.isConnected())
  {
    return;
  }

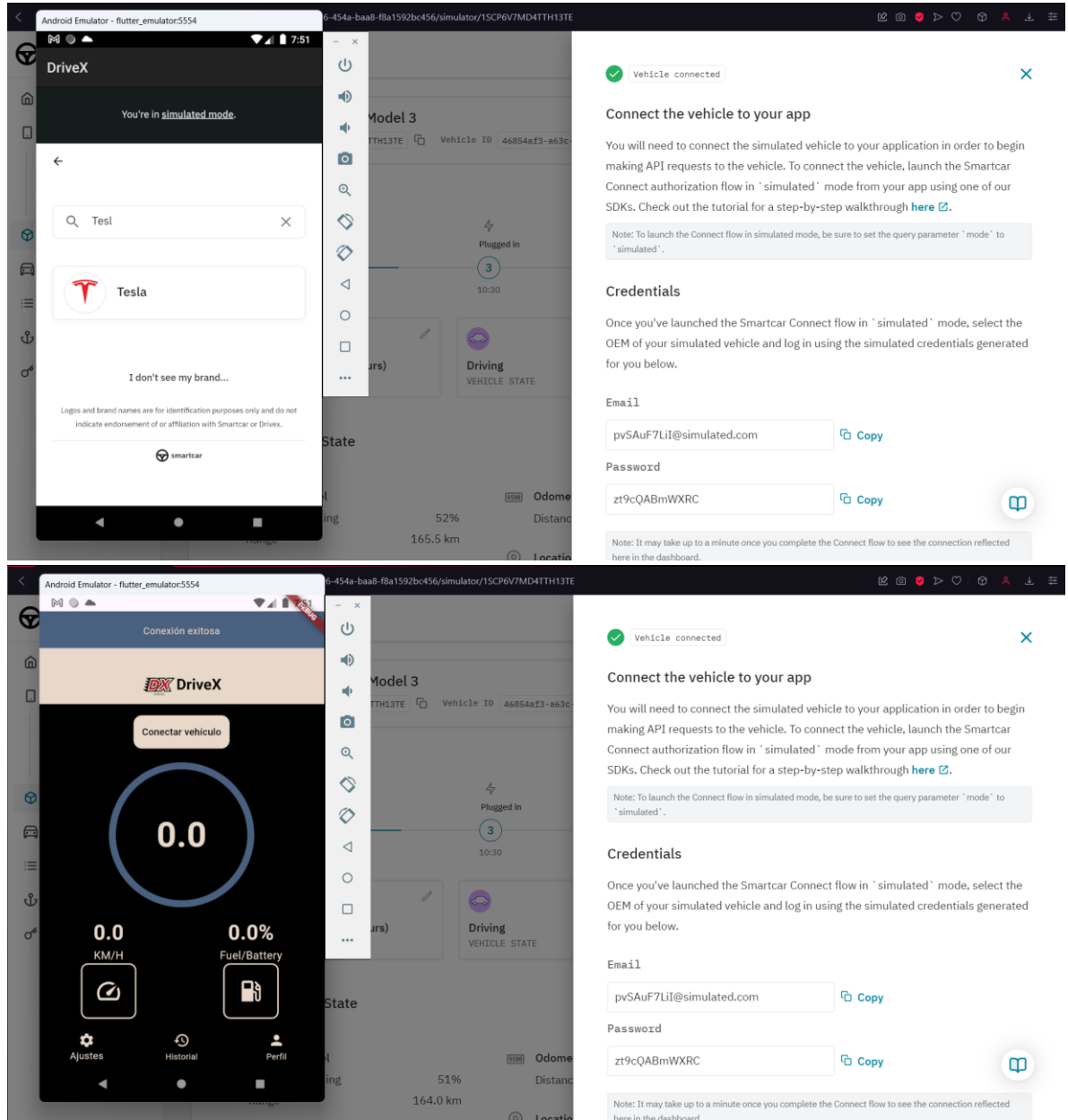
  final List<ConnectivityResult> result = await
Connectivity().checkConnectivity();

  if(result.contains(ConnectivityResult.wifi) ||
result.contains(ConnectivityResult.mobile))
  {
    await Smartcar.launchAuthFlow();
  }
  else
  {
    ScaffoldMessenger.of(context).showMaterialBanner(
      MaterialBanner(
        backgroundColor: GlobalSettings().thirdColor,
        content: Center(
          child: Text(
            'No se encuentra conectado a la red',
            style: TextStyle(
              color: GlobalSettings().foreColor,
            ),
          ),
        ),
        actions: const [SizedBox.shrink()],
      ),
    );
    Future.delayed(
      const Duration(
        seconds: 3,
      ),
    ).then((_) =>
ScaffoldMessenger.of(context).hideCurrentMaterialBanner());
  }
}
```

2. Prueba de autenticación de Smart Car

2.1 Ingreso de marca de vehículo y credenciales solicitadas:

- Resultado esperado: Conexión exitosa con el vehículo simulado.
- Resultado obtenido:



- Código implementado para el requerimiento:

```
static Future<void> getAccessToken() async
{
    Client client = Client();

    var authEncode =
base64.encode(utf8.encode('${APIRequestHelper.instance.clientId}:${APIRequestHelper.instance.clientSecret}'));
    final Response response = await
client.post(Uri.parse('https://auth.smartcar.com/oauth/token'), headers:
    {
        'Authorization': 'Basic $authEncode',
        'Content-Type': 'application/x-www-form-urlencoded',
    },
    body:
'grant_type=authorization_code&code=${APIRequestHelper.instance.authCode}&redirect_uri=${APIRequestHelper.instance.uri}'
    );

    if(response.statusCode == 200)
    {
        final data = json.decode(response.body);
        APIRequestHelper.instance.accessToken = AccessToken(
            value: data['access_token'],
            type: data['token_type'],
            expiresIn: data['expires_in'],
            refreshValue: data['refresh_token']
        );
    }

    print("Response: " + response.statusCode.toString());
}
```

```
static Future<void> loadUser() async
{
    if(!instance.isConnected())
    {
        return;
    }

    Client client = Client();

    final Response response = await
client.get(Uri.parse("https://api.smartcar.com/v2.0/user"),
```

```

        headers: {
            'Authorization': 'Bearer
${APIRequestHelper.instance.accessToken!.value}'
        }
    );

    if(response.statusCode == 200)
    {
        var data = json.decode(response.body);
        User user = User(id: data['id']);
        List<User> users = await APIDatabaseHelper.getUsers();
        bool newUser = true;
        for(int i = 0; i < users.length; i++)
        {
            if(users[i].id == user.id)
            {
                newUser = false;
                break;
            }
        }
        if(newUser)
        {
            APIDatabaseHelper.insertUser(user);
        }
        APIRequestHelper.instance.user = user;
    }

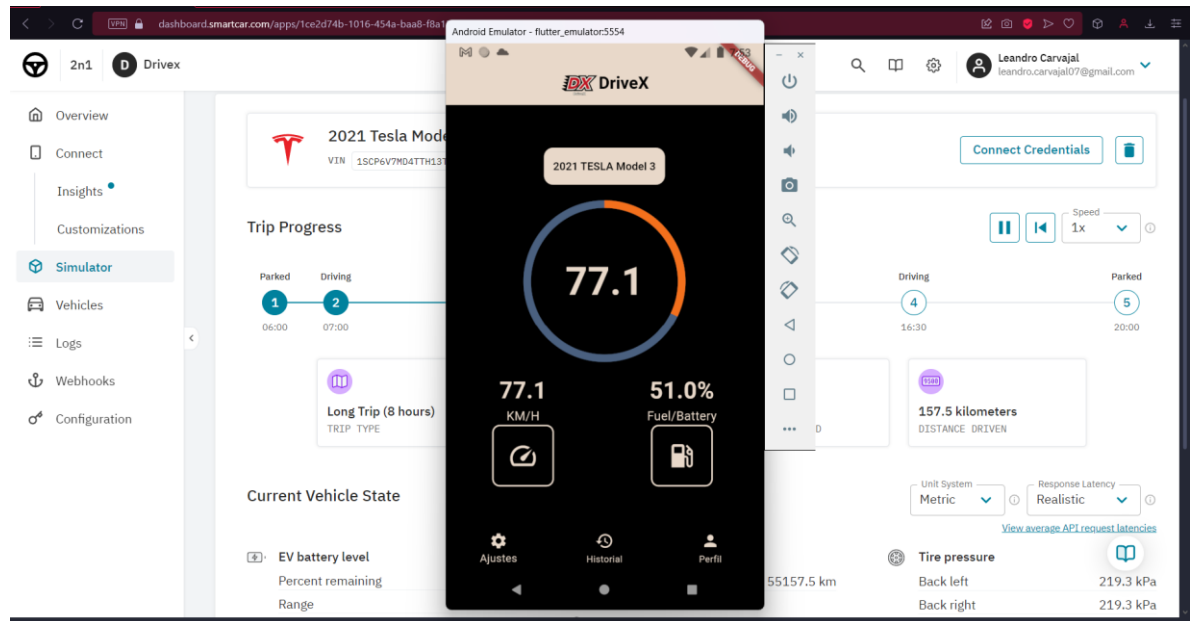
    print("Respuesta user: " + response.statusCode.toString());
    return null;
}

```


3. Prueba de obtención de información del vehículo

3.1 Tras realizar la conexión en menos de un minuto la aplicación obtiene la información relevante para la pantalla principal.

- Resultado esperado: Obtención de la información correspondiente al vehículo (Marca, modelo, año, odómetro y nivel del tanque).
- Resultado obtenido:



- Código implementado para el requerimiento:

```
static Future<Vehicle?> getVehicle(String id) async
{
  if(!instance.isConnected())
  {
    return null;
  }

  Client client = Client();

  final Response response = await
  client.get(Uri.parse("https://api.smartcar.com/v2.0/vehicles/$id"),
    headers: {
      'Authorization': 'Bearer
${APIRequestHelper.instance.accessToken!.value}'
    }
  );
};
```

```

        if(response.statusCode == 200)
        {
            var data = json.decode(response.body);
            return Vehicle(userID: APIRequestHelper.instance.user!.id, id:
data['id'], make: data['make'], model: data['model'], year: data['year']);
        }

        print("Respuesta vehiculo $id: " + response.statusCode.toString());
        return null;
    }
}

```

```

static Future<void> loadVehicles() async
{
    if(!instance.isConnected())
    {
        return;
    }

    Client client = Client();

    final Response response = await
client.get(Uri.parse("https://api.smartcar.com/v2.0/vehicles"),
    headers: {
        'Authorization': 'Bearer
${APIRequestHelper.instance.accessToken!.value}'
    }
);

    if(response.statusCode == 200)
    {
        var data = json.decode(response.body);
        print(data['vehicles'].length);
        for(int i = 0; i < data['vehicles'].length; i++)
        {
            Vehicle? vehicle;
            do
            {
                vehicle = await getVehicle(data['vehicles'][0]);
                if(vehicle != null)
                {
                    APIRequestHelper.instance.vehicles.add(vehicle);
                    print(vehicle.id);
                }
            }
        }
    }
}

```

```

        }while(vehicle == null);
    }

    List<Vehicle> vehicles = await APIDatabaseHelper.getVehicles();

    for(int i = 0; i < vehicles.length; i++)
    {
        await APIDatabaseHelper.insertVehicle(vehicles[i]);
    }
    print(data.toString());
}
}

```

```

static Future<double?> getFuelRemaining(Vehicle vehicle) async
{
    if(!instance.isConnected())
    {
        return null;
    }

    Client client = Client();

    final Response response = await
client.get(Uri.parse("https://api.smartcar.com/v2.0/vehicles/${vehicle.id}/fuel" ),
    headers: {
        'Authorization': 'Bearer
${APIRequestHelper.instance.accessToken!.value}'
    }
    );

    if(response.statusCode == 200)
    {
        var data = json.decode(response.body);
        return data['percentRemaining'] * 100;
    };

    print("Respuesta combustible" + response.statusCode.toString());
    return null;
}

```

```

static Future<double?> getEVRemaing(Vehicle vehicle) async
{
    if(!instance.isConnected())

```

```

    {
        return null;
    }

    Client client = Client();

    final Response response = await
client.get(Uri.parse("https://api.smartcar.com/v2.0/vehicles/${vehicle.id}/b
attery" ),
        headers: {
            'Authorization': 'Bearer
${APIRequestHelper.instance.accessToken!.value}'
        }
    );

    if(response.statusCode == 200)
    {
        var data = json.decode(response.body);
        switch(data['percentRemaining'])
        {
            case int i: return i.toDouble() * 100;
            case double d: return d * 100;
        }
    }
};

print("Respuesta bateria" + response.statusCode.toString());
return null;
}

```

```

static Future<double?> getOdometer(Vehicle vehicle) async
{
    if(!instance.isConnected())
    {
        return 0;
    }

    Client client = Client();

    final Response response = await
client.get(Uri.parse("https://api.smartcar.com/v2.0/vehicles/${vehicle.id}/o
dometer"),
        headers: {
            'Authorization': 'Bearer
${APIRequestHelper.instance.accessToken!.value}'

```

```
    }  
    );  
  
    print("Respuesta odometro: ${response.statusCode}");  
    if(response.statusCode == 200)  
    {  
        var data = json.decode(response.body);  
        var value = data['distance'];  
  
        switch(value)  
        {  
            case int i: return i.toDouble();  
            case double d: return d;  
            case String s: return double.parse(s);  
        }  
    }  
}
```