

Vehículos conectados a la nube con IoT

Luis Coco Enríquez



luxoft
A DXC Technology Company



El futuro de los coches definidos por software está aquí.



[Luxoft's remote repair of software-defined vehicles - YouTube](#)

¿Qué es un vehículo conectado?

Transformación digital de los vehículos mediante **ordenadores a bordo** con conexión a internet.

Electronic Control Units **ECUs** vs **IVC** In-Vehicle-Computer y el **IVI** In-Vehicle-Infotainment, ambos con capacidad **IoT**, mediante tarjeta **SIM**.

Protocolos de comunicación: LTE, 4G, 5G, etc.

Comunicación del vehículo con la infraestructura, los peatones, con otros vehículos, con satélites, etc.

V2I Vehicle to Infrastructure

V2N Vehicle to Network

V2P Vehicle to Pedestrian

V2V Vehicle to Vehicle

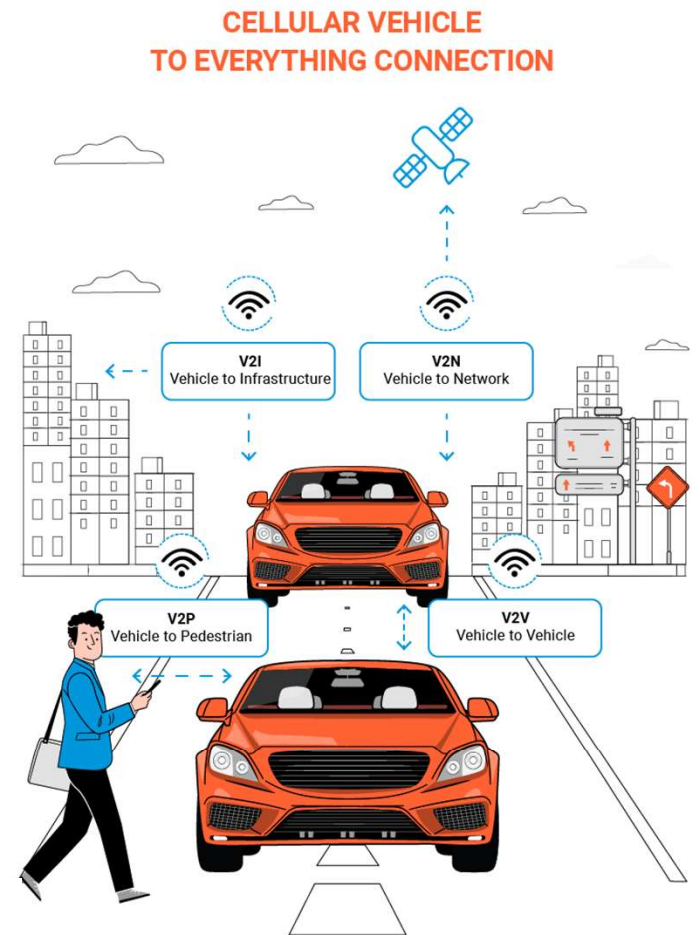
V2G Vehicle to Grid

V2X Vehicle connected to everything

[Ericsson Connected Vehicle](#)

[Eviden - Connected Vehicle](#)

[Connected vehicles video](#)



Ordenadores a bordo de un vehículo conectado

Los coches de última generación disponen de: control de **navegación**, **conducción autónoma** y sistemas avanzados de seguridad, controles digitales del vehículo (aire acondicionado, consumo eléctrico, etc).

El ordenador de a bordo de los coches **Tesla** combina tres ordenadores con funcionalidades diferenciadas pero interconectados entre ellos: **Media unit**, **Telematic board**, **Full Self-Driving computer**.



Mercedes EQE 350+ cockpit



2023 Tesla Model S Plaid video



Renault Austral 2023 Multimedia & Cockpit



2023 BMW X1 Multimedia & Digital Cockpit



Nissan X-TRAIL 2023 cockpit and interior



AUDI Q6 e-tron 2024 digital cockpit and infotainment

Hardware de un vehículo conectado Tesla

[Tesla's new car on-board computer](#)

[Tesla's NEW HW4 Car Computer](#)

[Tesla Hardware 3 \(Full Self-Driving Computer\) Detailed](#)

[Tesla Model S Plaid | ADAS Breakdown](#)

[Tesla's new self-driving \(HW4\) computer leaks: Here's a teardown](#)



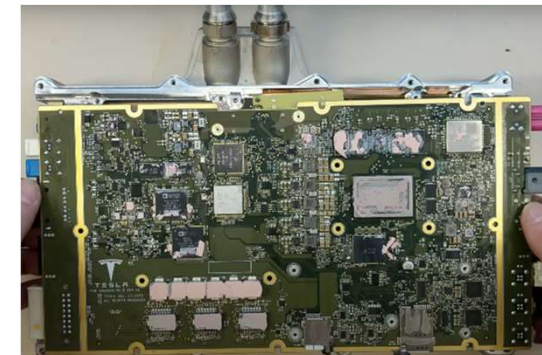
Graphic card



Computer back-side



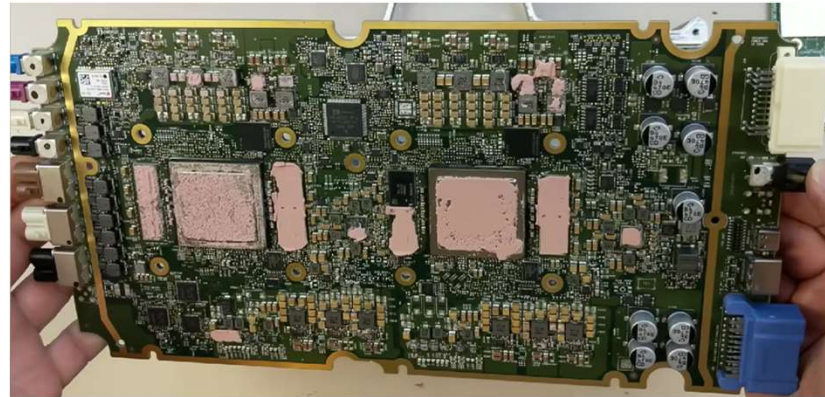
[Tesla Model 3 Y S X EU LTE](#)
[3G Modem, Adapter,](#)
[Connectivity, QUECTEL](#)
[AG525R PCBA](#)



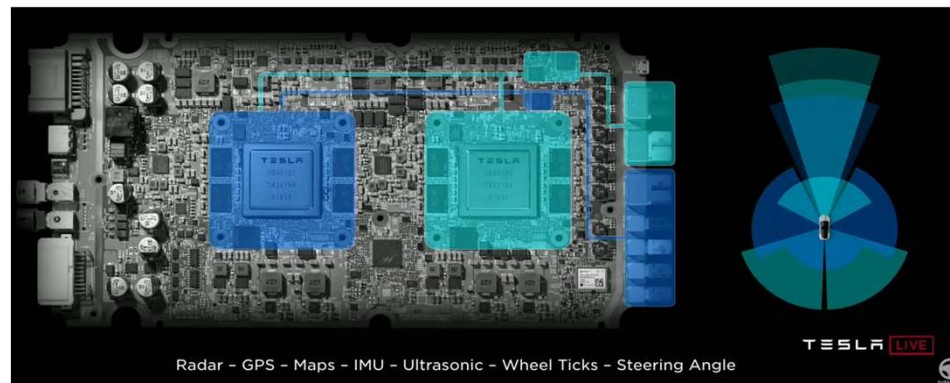
Hardware de un vehículo conectado



AutoPilot Computer back-side

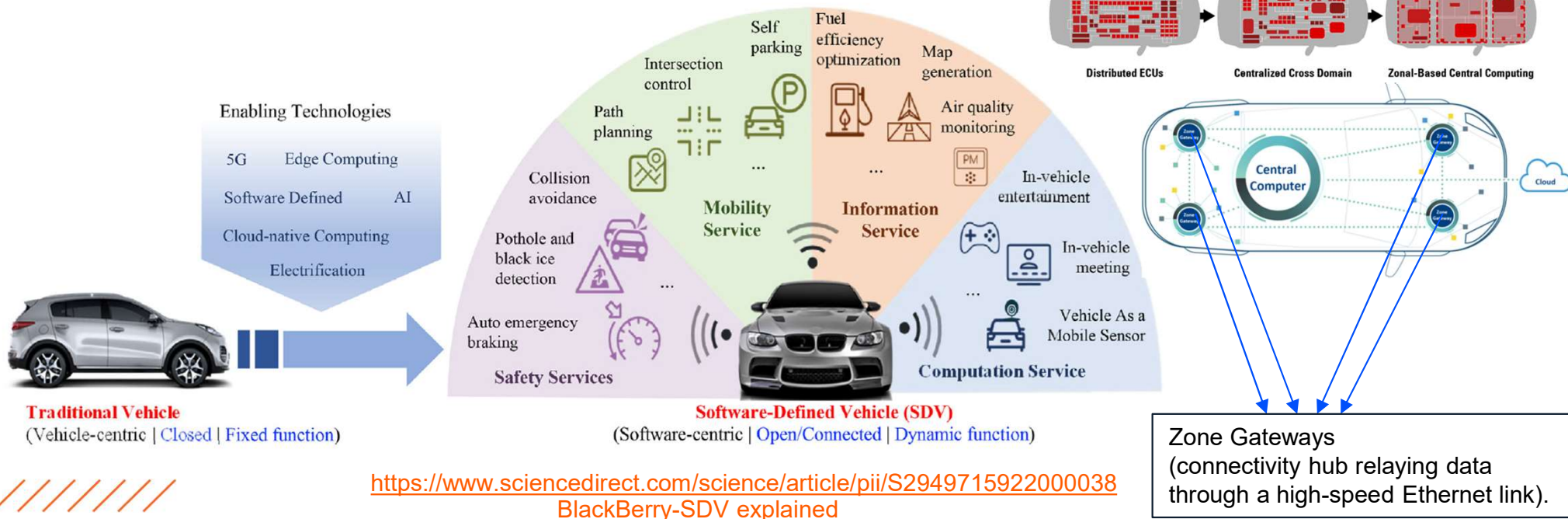


AutoPilot Computer front-side



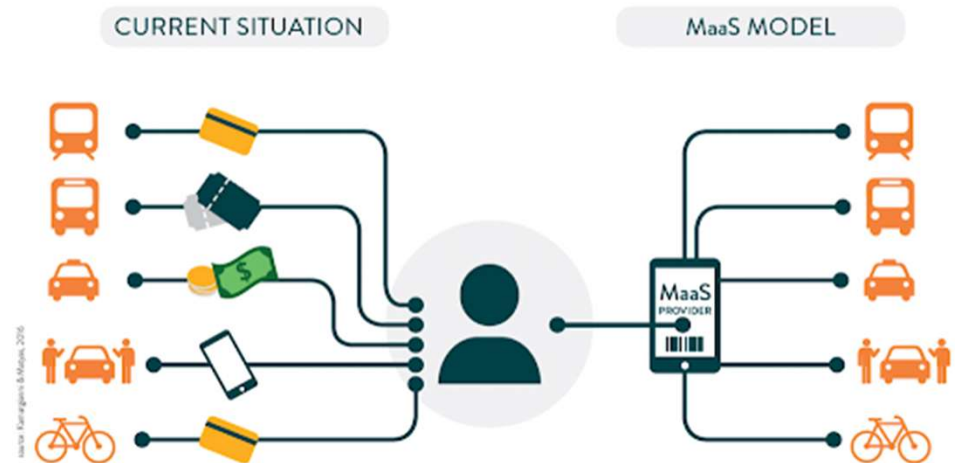
Vehículos definidos por software (SDV)

- La **arquitectura central o zonal** forma la columna vertebral de los SDV (multiple ECUs vs central computer)
- [Unveiling the Transformation of Software-Defined Vehicles \(eetimes.eu\)](https://www.eetimes.com/unveiling-the-transformation-of-software-defined-vehicles/) [The Eclipse Foundation](https://www.eclipsefoundation.org/)
- Sinergias/conexión con los **servicios en la nube**: AWS, Azure, Google Cloud, etc.
- Actualizaciones inalámbricas (**OTA**).



¿Qué es Mobility as a Service (MaaS)?

- **Plataforma inteligente** de **vehículos conectados**.
- **Planificar rutas**. Optimizar flujos de transporte.
- Sustituir compra de vehículos por **Pay as you drive**, y promover el **carsharing**.
- **Reducir** las superficies para **aparcamientos**.
- “**Ventanilla única**” para diferentes medios de transporte. Una aplicación y un sistema de pago.
- **Vehículos autónomos**, self-driving bus.
- **Compartir datos en tiempo real**: tráfico, meteorológicas, disponibilidad de vehículos.
- **Experiencia** de viaje **personalizada**: preferencias y hábitos de los usuarios.
- **Optimizar costes** y reducir la huella de carbono.



[Mobility as a Service | URBAN MOBILITY SIMPLY EXPLAINED](#)

[Mobility as a Service Components](#)

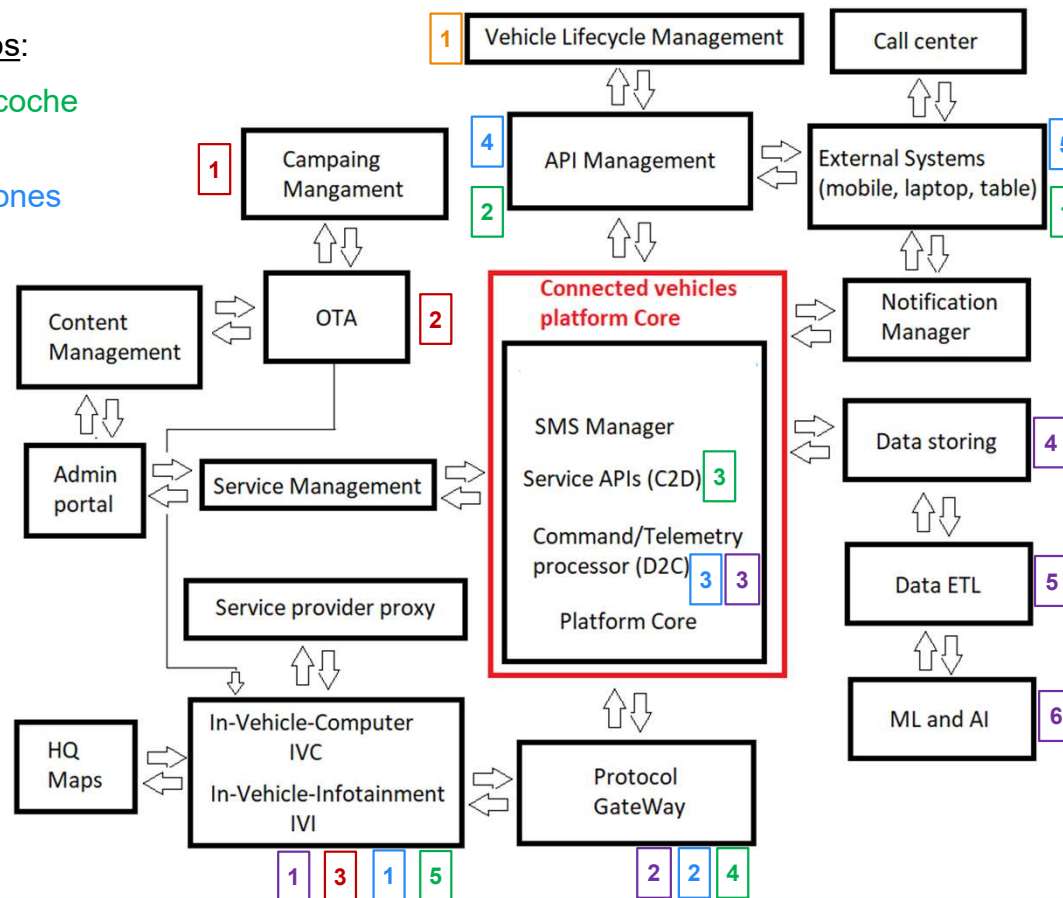
Arquitectura de una Plataforma de vehículos conectados

Flujos principales de datos:

- Envío de Comandos al coche (Cloud to Device C2D)
- Telemetrías y Notificaciones (Device to Cloud D2C)

Otros flujo de datos:

- Gestión de vida del vehículo
- Gestión de campaña + OTA
- Servicios de ML e AI
- Acceso a servicios en internet



Requisitos de las plataformas de vehículos conectados:

- Escalabilidad
- Fault tolerant (redundancia)
- Optimización de coste
- Baja latencia
- Ciberseguridad

[Building and Modernizing Connected Vehicle platforms with AWS IoT](#)

[Automotive messaging, data & analytics reference architecture - Azure Event Grid](#) | [Microsoft Learn](#)



Demostración de caso práctico con .NET



@luiscoco



Paso 1. Instalación del entorno de desarrollo

1.1. Instalación de **.NET 8 SDK**: <https://dotnet.microsoft.com/es-es/download/dotnet/thank-you/sdk-7.0.403-windows-x64-installer>

Comprobar la instalación ejecutando el comando: **dotnet --version**

1.2. Instalación de **Visual Studio 2022 Community Edition**:
<https://visualstudio.microsoft.com/es/vs/community/>

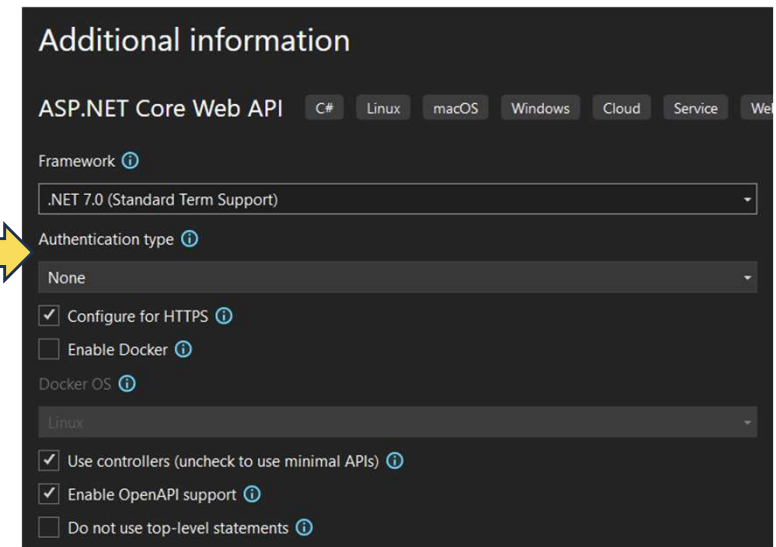
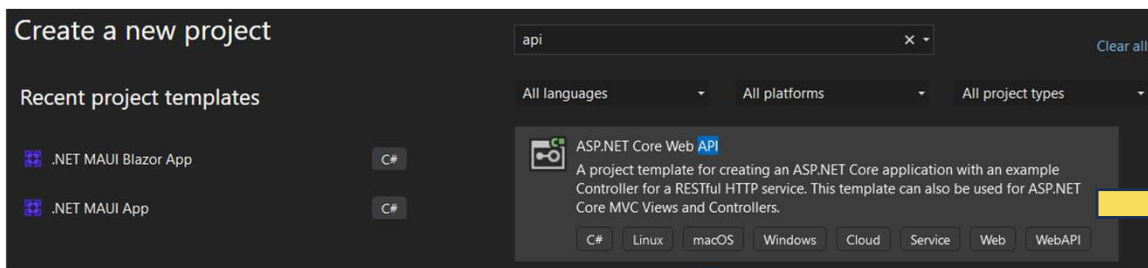
1.3. Instalación de **VSCode**: <https://code.visualstudio.com/download>

1.4. Instalación de **Docker Desktop**:
<https://docs.docker.com/desktop/install/windows-install/>

1.5. Instalación de **Studio 3T Free for MongoDB**: <https://studio3t.com/download/>

Paso 2. Creamos una aplicación .NET API

Opción 1. Utilizamos Visual Studio 2022



Opción 2. Utilizamos comandos dotnet CLI

```

C:\>dotnet --version
7.0.402

C:\>md codemotionTeslaAPI

C:\>cd codemotionTeslaAPI

C:\codemotionTeslaAPI>dotnet new webapi --framework net7.0
The template "ASP.NET Core Web API" was created successfully.

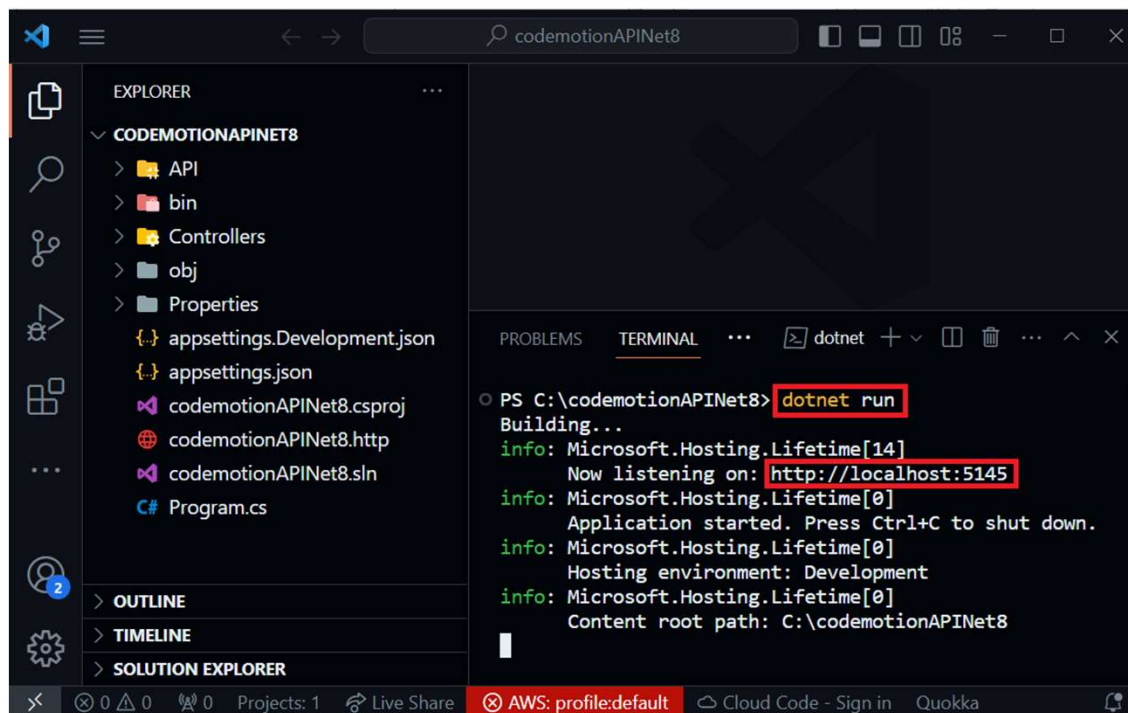
Processing post-creation actions...
Restoring C:\codemotionTeslaAPI\codemotionTeslaAPI.csproj:
  Determining projects to restore...
  Restored C:\codemotionTeslaAPI\codemotionTeslaAPI.csproj (in 2.48 sec).
Restore succeeded.
    
```

```

dotnet --version
md codemotiondotnet8api
cd codemotiondotnet8api
dotnet new webapi --framework net8.0
code .
    
```


Paso 2. Desarrollar nuestra aplicación con VSCode

Después de crear la aplicación con **dotnet CLI** ejecutamos el comando: **code .**
Para ejecutar la Web API ejecutamos el commando: **dotnet run**



Command Prompt

```
C:\>cd codemotionAPINet8
C:\codemotionAPINet8>code .
```

<http://localhost:5136/swagger/index.html>

Paso 3. Instalación de dependencias (VSCode)

Instalamos las siguientes librerías (paquetes Nuget) ejecutando los siguientes comandos:

```
dotnet add package Microsoft.AspNetCore.Cors (Enable Cross-Origin Requests CORS)
dotnet add package Microsoft.Extensions.Configuration (Read and write in
appSetting.json file)
dotnet add package MongoDB.Driver (MongoDb communication)
dotnet add package Newtonsoft.Json (JSON Serialization and Deserialization)
```

Para añadir los paquetes al proyecto ejecutamos el comando: **dotnet restore**

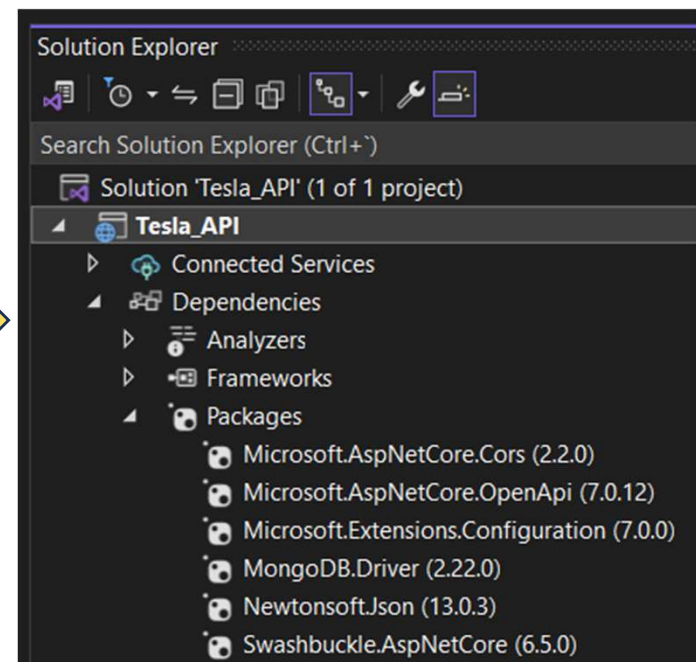
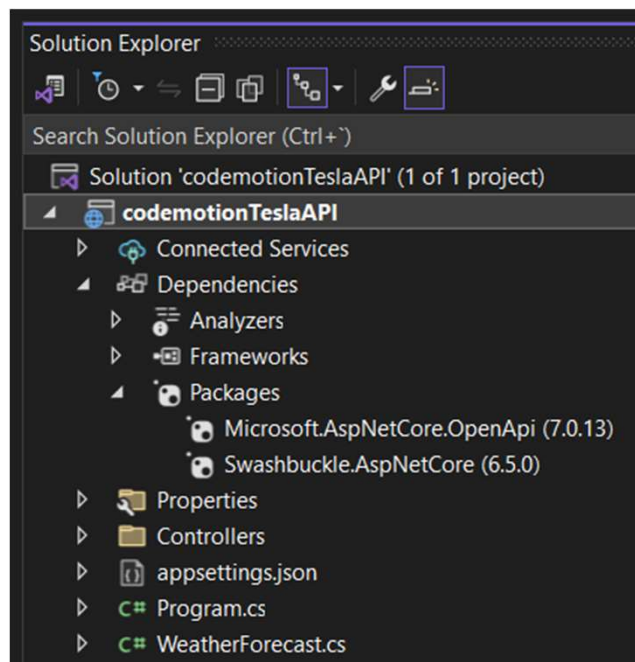
Verificamos que la instalación de la librerías ha sido correcta, consultando el archive **csproj**.

```
dotnet add package Microsoft.AspNetCore.Cors
dotnet add package Microsoft.Extensions.Configuration
dotnet add package MongoDB.Driver
dotnet add package Newtonsoft.Json
```

Paso 3. Instalación de dependencias

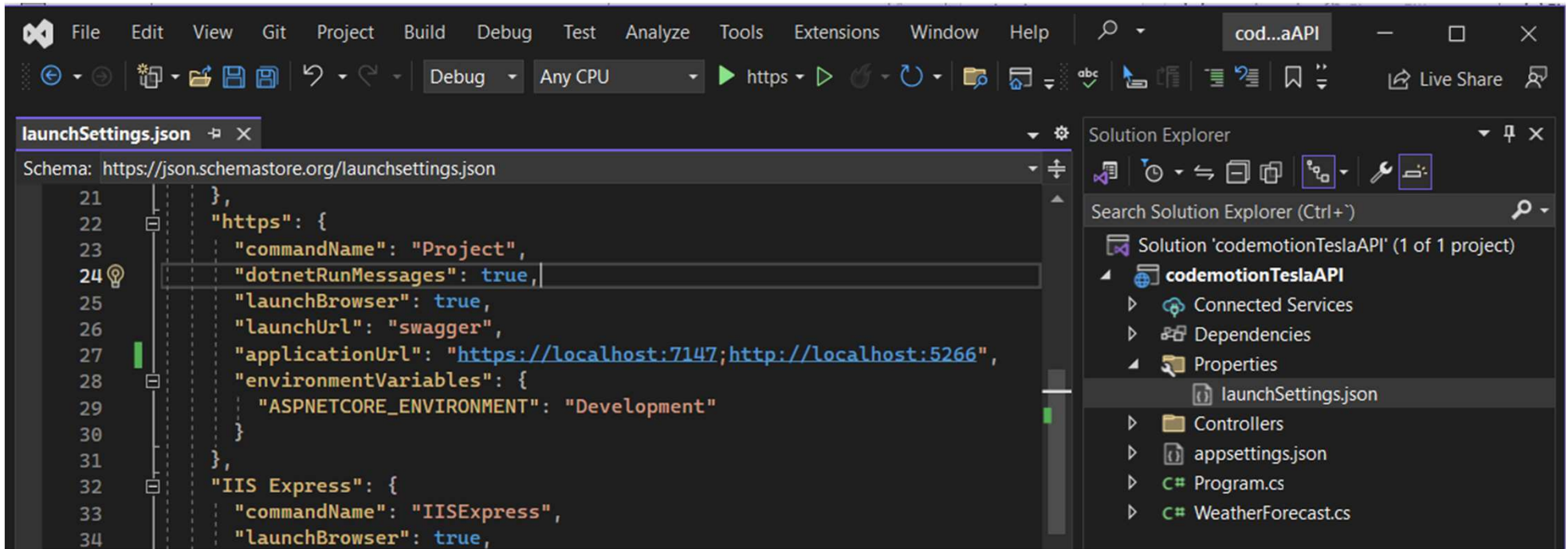
Instalamos las siguientes librerías con ayuda de **NuGet**:

Microsoft.AspNetCore.Cors Microsoft.Extensions.Configuration MongoDB.Driver
Newtonsoft.Json



Paso 4. Modificamos el archivo “launchSettings.json”

Fijamos el número de puerto (7147) para ejecutar la aplicación API



```

21      },
22      "https": {
23        "commandName": "Project",
24        "dotnetRunMessages": true,
25        "launchBrowser": true,
26        "launchUrl": "swagger",
27        "applicationUrl": "https://localhost:7147;http://localhost:5266",
28        "environmentVariables": {
29          "ASPNETCORE_ENVIRONMENT": "Development"
30        }
31      },
32      "IIS Express": {
33        "commandName": "IISExpress",
34        "launchBrowser": true,

```

Solution Explorer

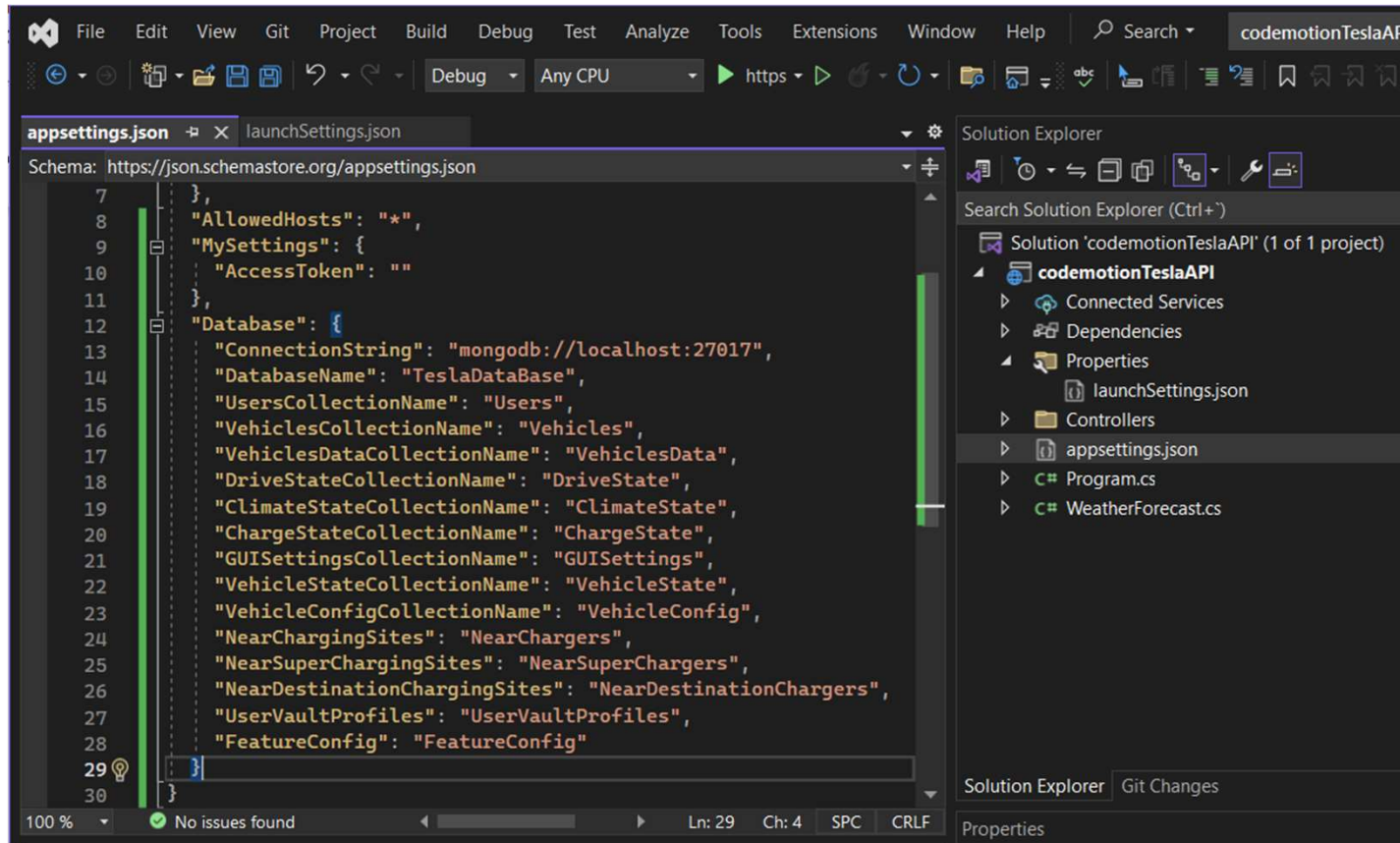
Search Solution Explorer (Ctrl+)

Solution 'codemotionTeslaAPI' (1 of 1 project)

- codemotionTeslaAPI
 - Connected Services
 - Dependencies
 - Properties
 - launchSettings.json
 - Controllers
 - appsettings.json
 - Program.cs
 - WeatherForecast.cs

Paso 5. Modificamos el archivo “appSettings.json”

Introducimos los datos de la base de datos MongoDB y el AccessToken



```

Schema: https://json.schemastore.org/appsettings.json
7  },
8  "AllowedHosts": "*",
9  "MySettings": {
10   "AccessToken": ""
11 },
12 "Database": {
13   "ConnectionString": "mongodb://localhost:27017",
14   "DatabaseName": "TeslaDataBase",
15   "UsersCollectionName": "Users",
16   "VehiclesCollectionName": "Vehicles",
17   "VehiclesDataCollectionName": "VehiclesData",
18   "DriveStateCollectionName": "DriveState",
19   "ClimateStateCollectionName": "ClimateState",
20   "ChargeStateCollectionName": "ChargeState",
21   "GUISettingsCollectionName": "GUISettings",
22   "VehicleStateCollectionName": "VehicleState",
23   "VehicleConfigCollectionName": "VehicleConfig",
24   "NearChargingSites": "NearChargers",
25   "NearSuperChargingSites": "NearSuperChargers",
26   "NearDestinationChargingSites": "NearDestinationChargers",
27   "UserVaultProfiles": "UserVaultProfiles",
28   "FeatureConfig": "FeatureConfig"
29 }
30 }
  
```

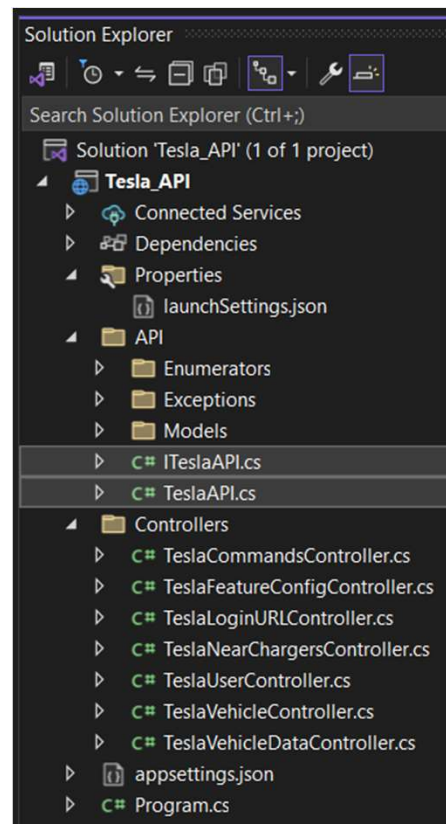
100 % No issues found Ln: 29 Ch: 4 SPC CRLF

Solution Explorer: codemotionTeslaAPI (1 of 1 project)

- codemotionTeslaAPI
 - Connected Services
 - Dependencies
 - Properties
 - launchSettings.json
 - Controllers
 - appsettings.json
 - Program.cs
 - WeatherForecast.cs

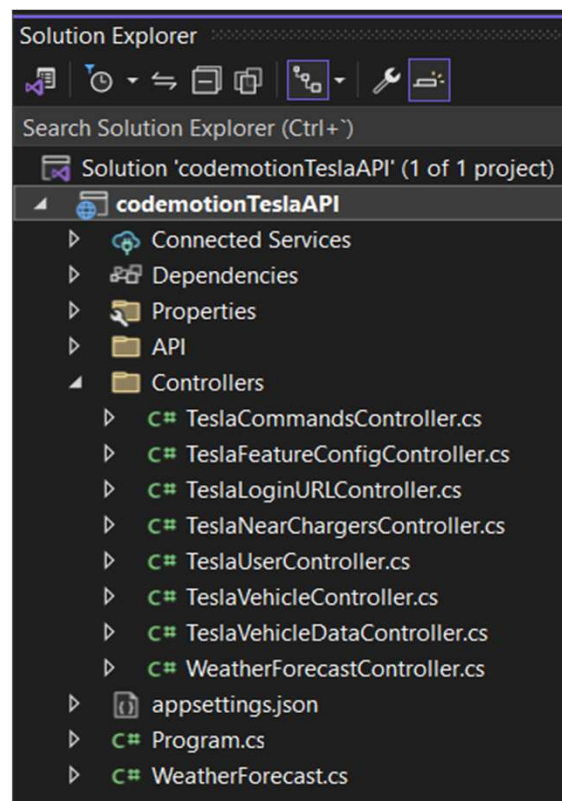
Paso 6. Creamos los modelos de datos y servicio “TeslaAPI”

Creamos una nueva carpeta “API” donde ubicaremos los modelos de datos de nuestro aplicación. Creamos las clases que incluyen los modelos de datos de la aplicación.



Paso 7. Creamos los controladores de la aplicación

Creamos una nueva carpeta “API” donde ubicamos los modelos de datos de nuestro aplicación. Creamos las clases que incluyen los modelos de datos de la aplicación.



Paso 7 (opción 2). Creamos los controladores con inyección de dependecia del servicio "TeslaAPI"

```
using TeslaAPI;
...
public class TeslaVehicleController : ControllerBase
{
    ...
    private readonly ITeslaAPI _teslaAPIService;

    public TeslaVehicleController(IConfiguration configuration, IMongoCollection<Vehicle> vehiclesCollection,
        ILogger<TeslaVehicleController> logger, ITeslaAPI teslaAPIService)
    {
        ...
        _teslaAPIService = teslaAPIService;
        ...
    }

    [HttpPost("getVehicle", Name = "GetVehicle")]
    public async Task<Vehicle> GetVehicle()
    {
        Vehicle vehicle = new Vehicle();
        vehicle = await _teslaAPIService.WakeupAsync("vehicleNumber");
        vehicle = await _teslaAPIService.GetVehicleAsync("vehicleNumber ");

        return vehicle;
    }
}
```


Paso 7 (opción 2). Creamos los controladores con inyección de dependencia del servicio "TeslaAPI"

Recomiendo la opción con inyección de dependencia, es más **extensible** y **mantenible** a largo plazo.

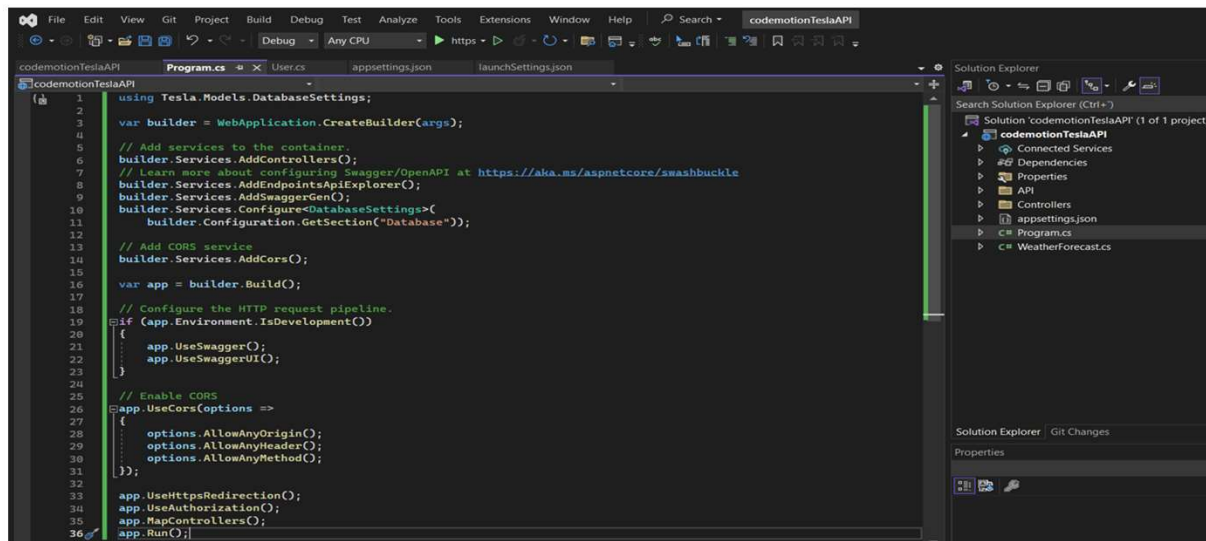
Capacidad de prueba: mejor capacidad de prueba al inyectar la interfaz `ITeslaAPIService`, puede sustituirla fácilmente con una implementación simulada durante las pruebas unitarias.

Desacoplamiento: el controlador sólo depende de la interfaz `ITeslaAPIService`, lo que lo hace más flexible y fácil de reemplazar o ampliar en el futuro.

Mantenibilidad: si necesita cambiar la implementación de la API de Tesla o agregar nuevas funciones, puede hacerlo sin modificar el código del controlador.

Paso 8. Modificar el archive program.cs

Incluimos los servicios de la base de datos **MongDB** y **CORS**



```

1 using Tesla.Models.DatabaseSettings;
2
3 var builder = WebApplication.CreateBuilder(args);
4
5 // Add services to the container.
6 builder.Services.AddControllers();
7 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
8 builder.Services.AddEndpointsApiExplorer();
9 builder.Services.AddSwaggerGen();
10 builder.Services.Configure<DatabaseSettings>(>
11     builder.Configuration.GetSection("Database"));
12
13 // Add CORS service
14 builder.Services.AddCors();
15
16 var app = builder.Build();
17
18 // Configure the HTTP request pipeline.
19 if (app.Environment.IsDevelopment())
20 {
21     app.UseSwagger();
22     app.UseSwaggerUI();
23 }
24
25 // Enable CORS
26 app.UseCors(options =>
27 {
28     options.AllowAnyOrigin();
29     options.AllowAnyHeader();
30     options.AllowAnyMethod();
31 });
32
33 app.UseHttpsRedirection();
34 app.UseAuthorization();
35 app.MapControllers();
36 app.Run();
    
```

Si preferimos realizar la **inyección de dependencias** del servicio **TeslaAPI** en los constructores de los controladores, entonces incluir el siguientes código en el archive **program.cs**:

(opción 1)

services.AddScoped<ITeslaAPI, TeslaAPI>();

(opción 2)

services.AddTeslaApi();

Paso 9. Ejecutar contenedor de MongoDB

Ejecutar Docker Desktop

Opcion 1: Descargar y ejecutar el contenedor docker de MongoDB :

```
docker run -d --rm -p 27017:27017 --name mongo --network mongoCluster mongo:latest mongod --bind_ip_all
```

```
docker ps -a
```

or

Opcion 2: Descargar y ejecutar los contenedores docker de la MongoDB ReplicaSet:

```
docker run -d --rm -p 27017:27017 --name mongo1 --network mongoCluster mongo:latest mongod --replSet myReplicaSet --bind_ip localhost,mongo1
```

```
docker run -d --rm -p 27018:27017 --name mongo2 --network mongoCluster mongo:latest mongod --replSet myReplicaSet --bind_ip localhost,mongo2
```

```
docker run -d --rm -p 27019:27017 --name mongo3 --network mongoCluster mongo:latest mongod --replSet myReplicaSet --bind_ip localhost,mongo3
```

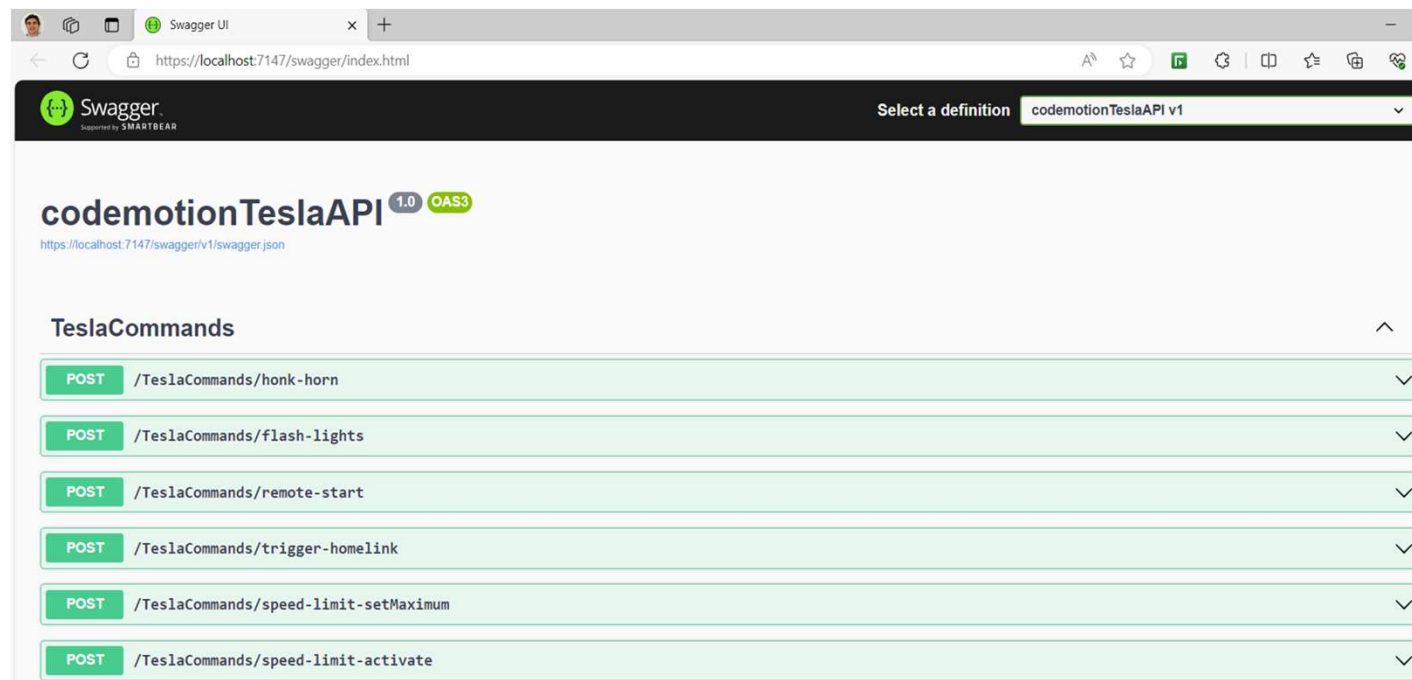
```
docker exec -it mongo1 mongosh --eval "rs.initiate({ _id: 'myReplicaSet', members: [{ _id: 0, host: 'mongo1', priority: 3}, { _id: 1, host: 'mongo2', priority: 2}, { _id: 2, host: 'mongo3', priority: 1}])"
```

```
docker ps -a
```

Paso 10. Compilar y Ejecutar la aplicación

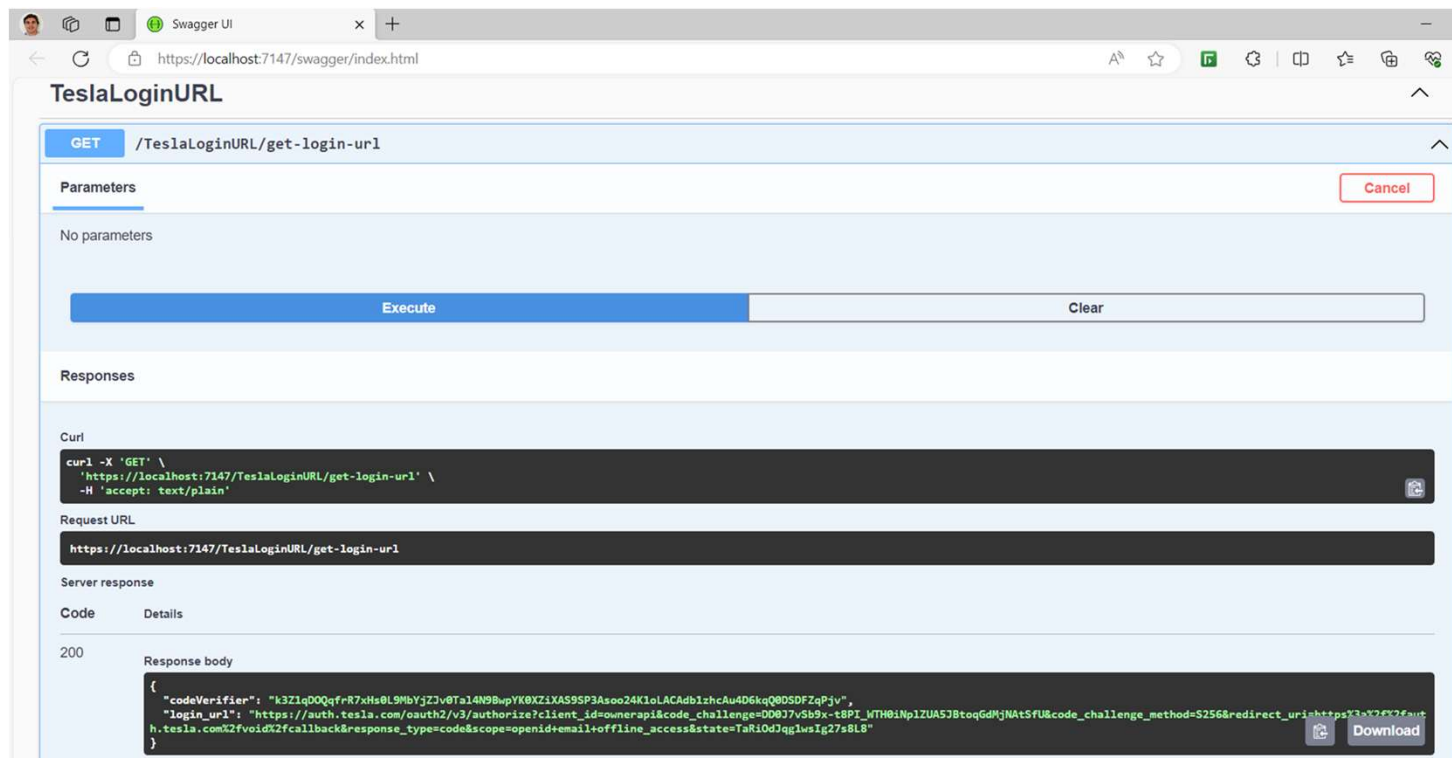
Antes de compilar y ejecutar la aplicación eliminamos los dos archivos “**WeatherForecast**” del ejemplo original de las aplicaciones .NET Web API

Compilamos y ejecutamos la aplicación .NET Web API

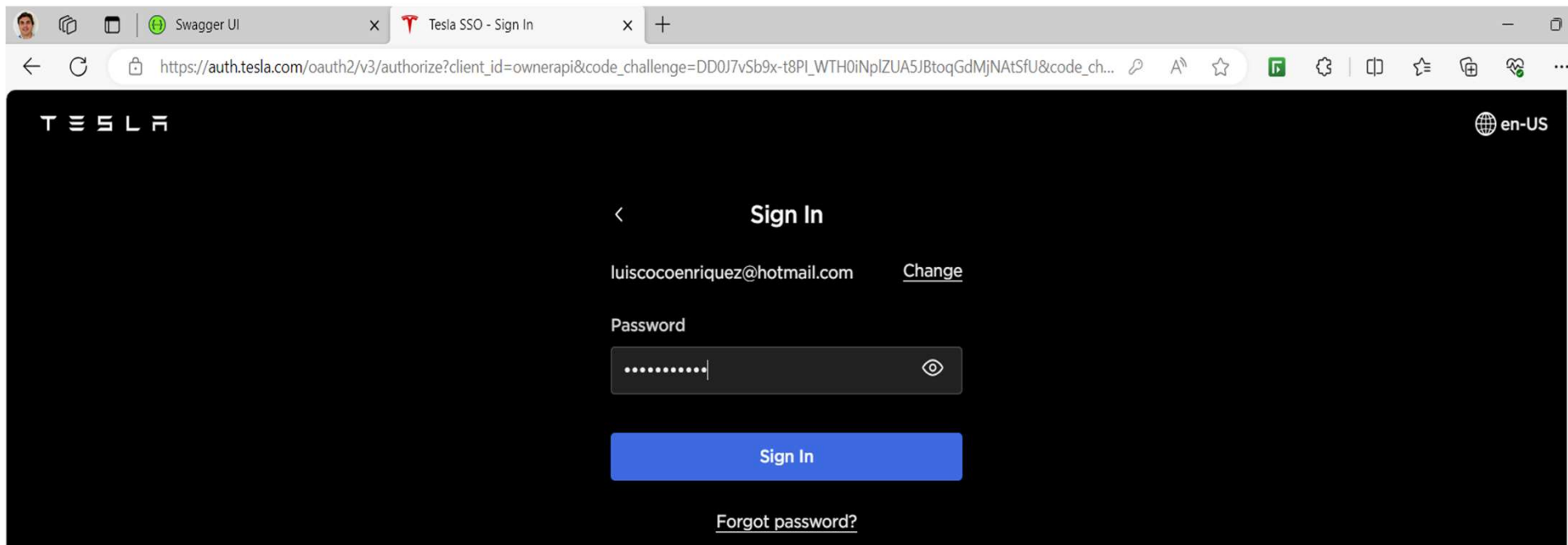


Paso 11. Login

Como paso previo para poder enviar solicitudes a las APIs de Tesla tenemos que autenticarnos con nuestro nombre de usuario y nuestro password en la página de Login de Tesla. Llamada a la acción **GetLoginURL()**, definida en el controlador **TeslaLoginURLController**



Paso 12. Continuamos con el Login




The screenshot shows a web browser window with two tabs: 'Swagger UI' and 'Tesla SSO - Sign In'. The address bar displays the URL: `https://auth.tesla.com/oauth2/v3/authorize?client_id=ownerapi&code_challenge=DD0J7vSb9x-t8PI_WTH0iNplZUA5JBtoqGdMjNAtSfU&code_ch...`. The Tesla SSO Sign In page has a dark background. At the top left is the Tesla logo, and at the top right is a globe icon with 'en-US'. The main content area is centered and contains a back arrow, the title 'Sign In', the email 'luiscocoenriquez@hotmail.com' with a 'Change' link, a 'Password' label, a password input field with a masked password '.....' and a toggle eye icon, a blue 'Sign In' button, and a 'Forgot password?' link at the bottom.

TESLA en-US

< Sign In

luiscocoenriquez@hotmail.com [Change](#)

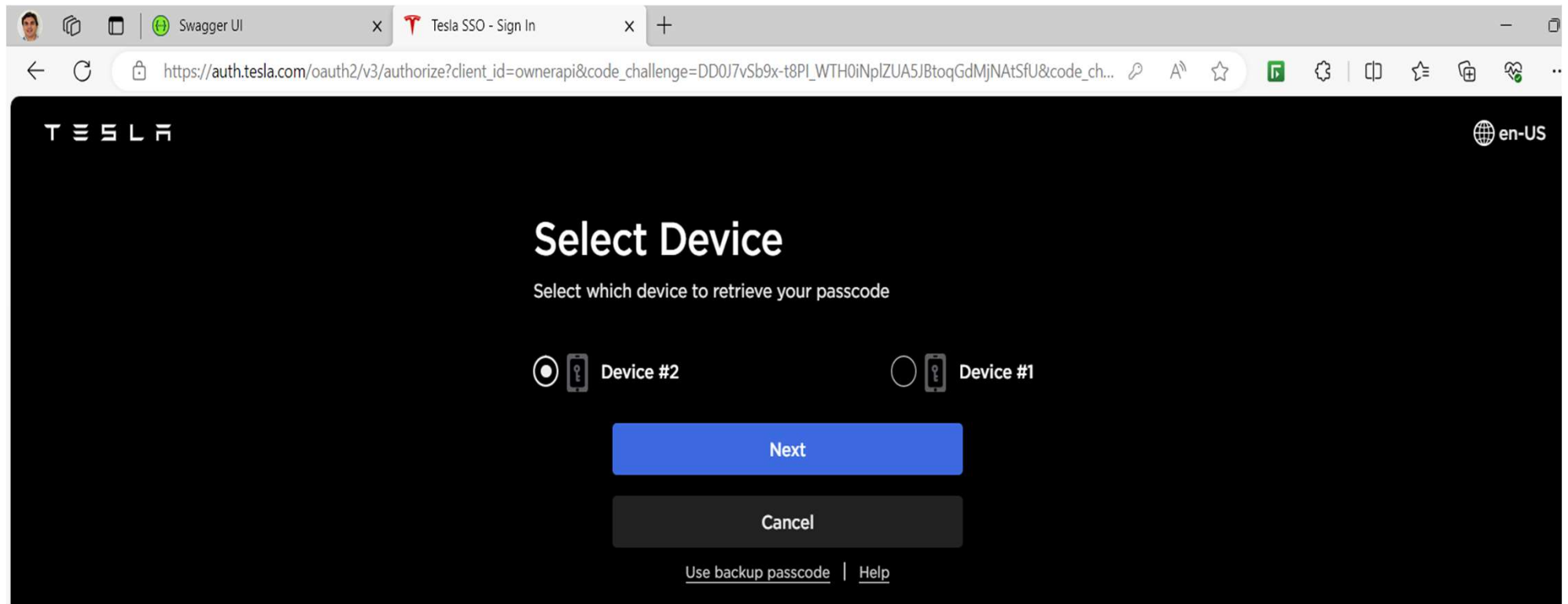
Password

..... 

Sign In

[Forgot password?](#)

Paso 13. MFA(Multi-Factor Authentication)



The screenshot shows a web browser window with two tabs: 'Swagger UI' and 'Tesla SSO - Sign In'. The address bar displays the URL: `https://auth.tesla.com/oauth2/v3/authorize?client_id=ownerapi&code_challenge=DD0J7vSb9x-t8PI_WTH0iNpIZUA5JBtoqGdMjNatSfU&code_ch...`. The Tesla SSO page has a dark background with the Tesla logo in the top left and 'en-US' in the top right. The main heading is 'Select Device' with the instruction 'Select which device to retrieve your passcode'. There are two radio button options: 'Device #2' (selected) and 'Device #1'. Below these are two buttons: 'Next' (blue) and 'Cancel' (grey). At the bottom, there are links for 'Use backup passcode' and 'Help'.

TESLA en-US

Select Device


Select which device to retrieve your passcode

☒ Device #2 ☐ Device #1

Next

Cancel

[Use backup passcode](#) | [Help](#)



The screenshot shows the Swagger UI for a REST API. The endpoint is `POST /TeslaLoginURL/get-token-after-login`. The parameters section lists two parameters: `codeverifier` (string, query) with value `k3Z1qDOOqfrR7xHs0L9MbYjZv0TaIH9Bw` and `redirectUrl` (string, query) with value `https://auth.tesla.com/void/callback?code=EL`. The responses section shows a curl command: `curl -X 'POST' https://localhost:7147/TeslaLoginURL/get-token-after-login?codeverifier=k3Z1qDOOqfrR7xHs0L9MbYjZv0TaIH9BwYK8RZ1XA595P3Asoo24K1oLACAdhshu4D6qQ0MS0FZgPjv&redirectUrl=https%3A%2F%2Fauth.tesla.com%2Fvoid%2Fcallback?code=EL -H 'accept: text/plain'`.

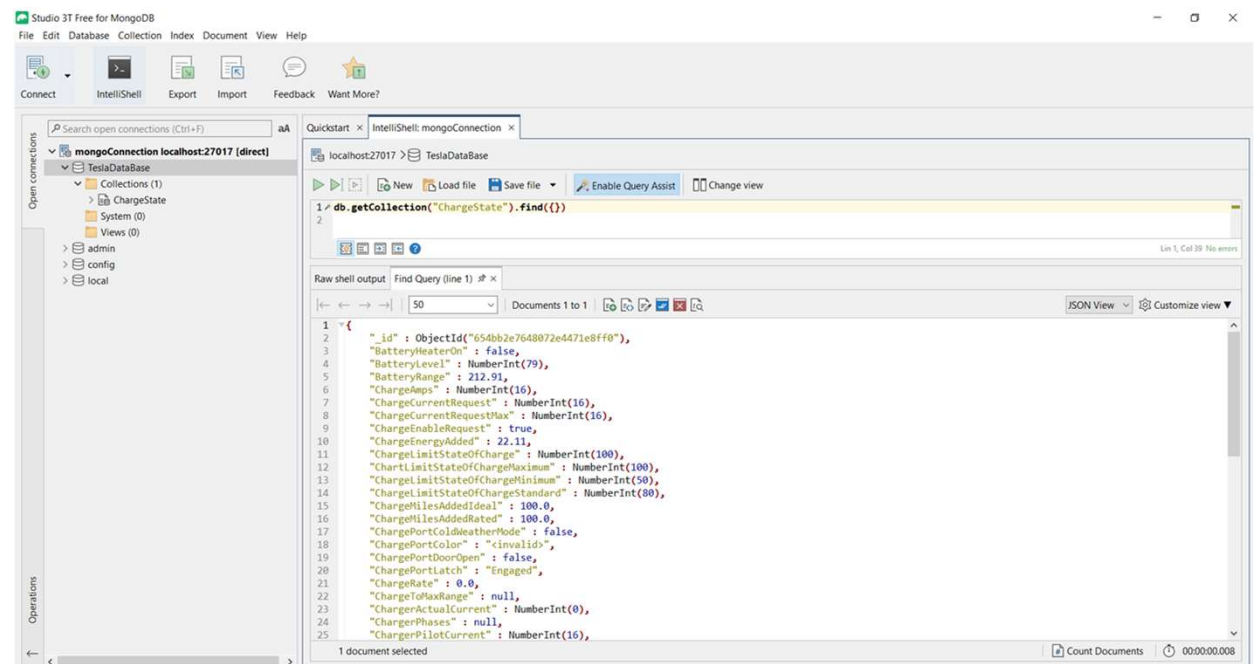
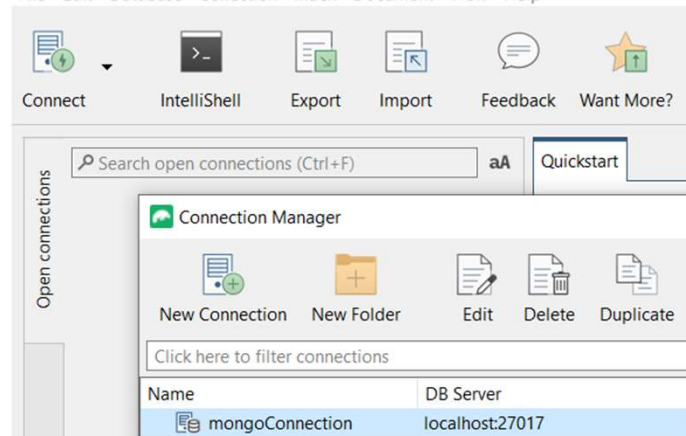
```
string token = configuration.GetSection("MySettings").GetSection("AccessToken").Value;
Client.DefaultRequestHeaders.Add("User-Agent", "Tests");
Client.DefaultRequestHeaders.Add("Authorization", $"Bearer {token}");
```

[illegible]

Paso 15. Comprobamos los datos escritos en MongoDB con Studio 3T

Studio 3T Free for MongoDB

File Edit Database Collection Index Document View Help



Paso 16. Comprobamos los datos escritos en MongoDB con comandos

```

C:\Users\LEnriquez>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
10878a94b059   mongo:latest   "docker-entrypoint.s..." About an hour ago Up About an hour   0.0.0.0:27017->27017/tcp   mongo

C:\Users\LEnriquez>docker exec -it mongo bash
root@10878a94b059:/# mongosh
Current Mongosh Log ID: 654bb3a1915464d60d7d7578
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.1
Using MongoDB:      7.0.2
Using Mongosh:      2.0.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2023-11-08T14:55:01.175+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://docs.mongodb.org/core/prodnotes-filesystem
2023-11-08T14:55:02.538+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2023-11-08T14:55:02.538+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
2023-11-08T14:55:02.538+00:00: vm.max_map_count is too low
-----

test> show dbs
TeslaDataBase 40.00 KiB
TeslaDataBase> use TeslaDataBase
already on db TeslaDataBase
TeslaDataBase> db.ChargeState.findOne()

```

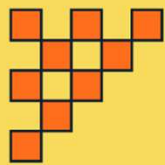
docker exec -it
mongo bash

mongosh
show dbs
use TeslaDataBase
show collections
db.VehiclesData.findOne()
db.VehiclesData.find()
db.Users.find()



PREGUNTAS?





REPOSITARIOS DE CÓDIGO Y OTROS...



Repositorio de códigos

- <https://www.teslaapi.io/>
- <https://www.tesla.com/developer-docs>
- <https://developer.tesla.com/docs/fleet-api>
- [GitHub - Azure/azure-iot-protocol-gateway:](#)
[Azure IoT protocol gateway enables protocol translation for Azure IoT Hub](#)
- <https://github.com/aws/aws-iot-fleetwise-edge>

Otros casos de estudio Luxoft

- **Coches que se curan solos:** <https://www.luxoft.com/videos/luxofts-remote-repair-of-software-defined-vehicles>
- <https://www.luxoft.com/files/pdfs/case-studies/Public-Transportation-Smart-Card.pdf>
- **Conectividad en tiempo real para seguros:** <https://www.luxoft.com/case-studies/enabling-effective-monetization-of-connected-car-data>
- **CX en movilidad urbana:** <https://www.luxoft.com/case-studies/an-intuitive-customer-experience-that-is-reshaping-the-future-of-urban-mobility-from-research-to-realization>
- **Gamification en coches:** <https://www.luxoft.com/blog/gamification-could-revolutionize-automotive-industry>



¡Contacta conmigo!

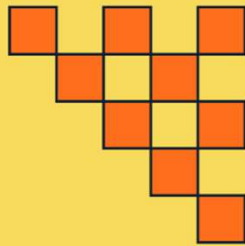


[@GitHub](#)



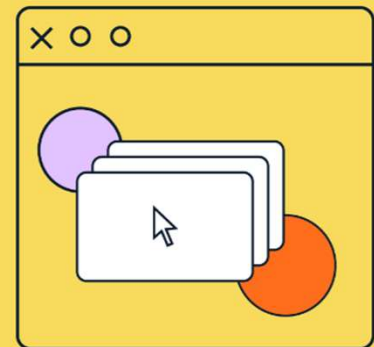
[@Linkedin](#)





{E} WORKSHOP FEST

¡Gracias!





{E} WORKSHOP FEST

**¡Recuerda puntuar
el Workshop!**

