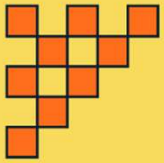


WORKSHOP FEST

Vehículos conectados a la nube con IoT

Luis Coco Enríquez

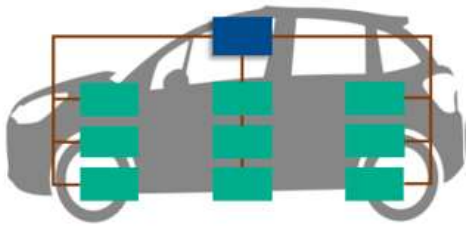
Luxoft
A DXC Technology Company

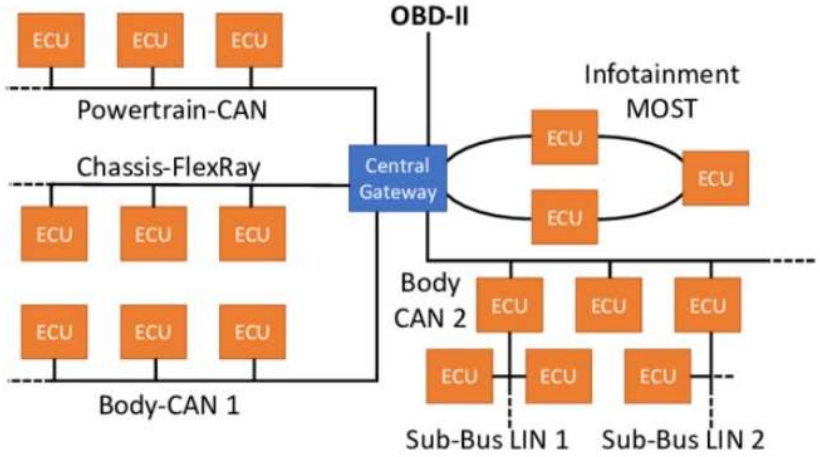
Luxoft: coches definidos por software ya están aquí!

Luxoft's remote repair of software-defined vehicles

Cuadro de mando y arquitectura tradicional de un coche

OBD2
On-Board Diagnostic II





ECU Electronic Control **U**nits conectadas mediante el protocolo **CAN** Bus

Cuadros de mando digitales

Los **coches de última generación** disponen de: control de **navegación**, "**infotainment**" (Sistema de entretenimiento), **conducción autónoma** y **sistemas seguridad avanzados**, **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

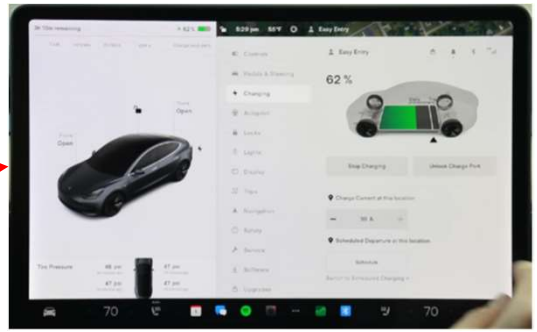
AUDI Q6 e-tron 2024 digital cockpit and infotainment

WORKSHOP FEST

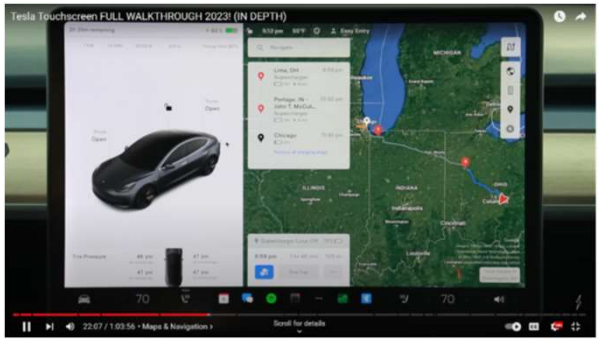
Pantalla del ordenador de abordo de un Tesla Model 3

Touchscreen | Model 3 and Model Y Tesla Touchscreen FULL WALKTHROUGH 2023! (IN DEPTH)

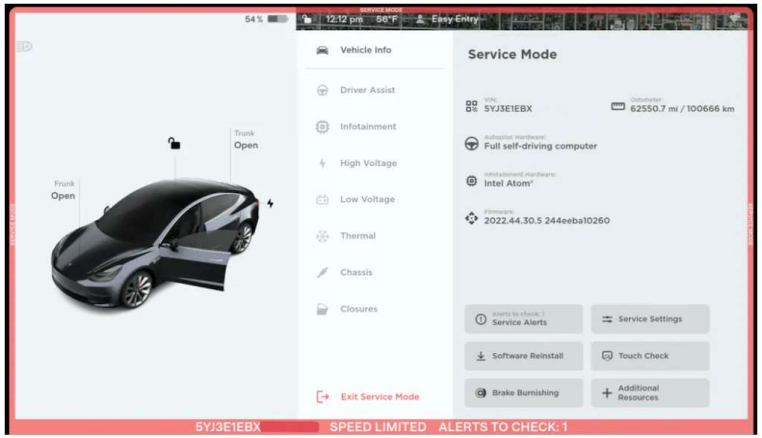
Controles de los **sistemas** del vehículo



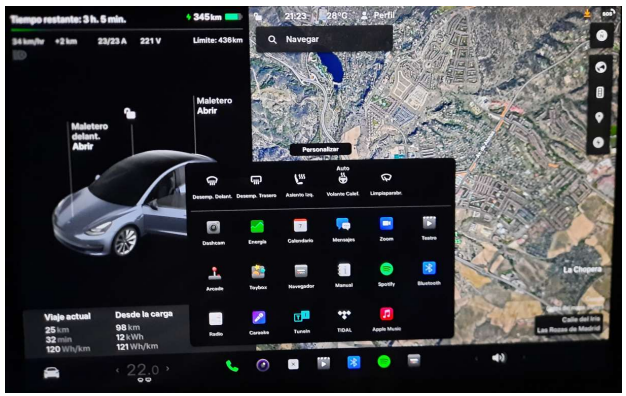
Sistema de **navegación** con Google Maps

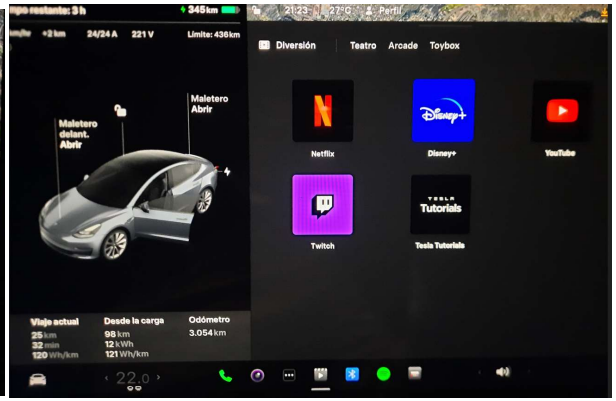


Menu para **diagnóstico y reparación**



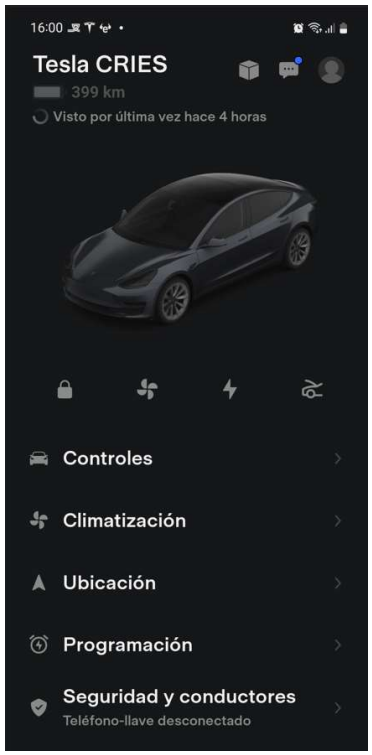
Aplicaciones de **infoentretenimiento**

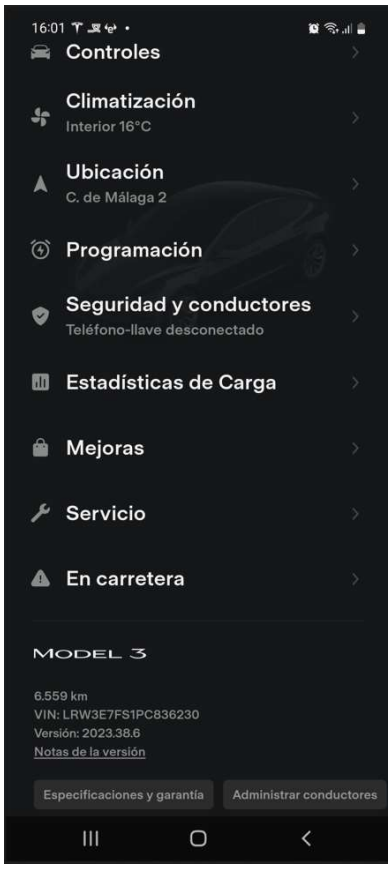


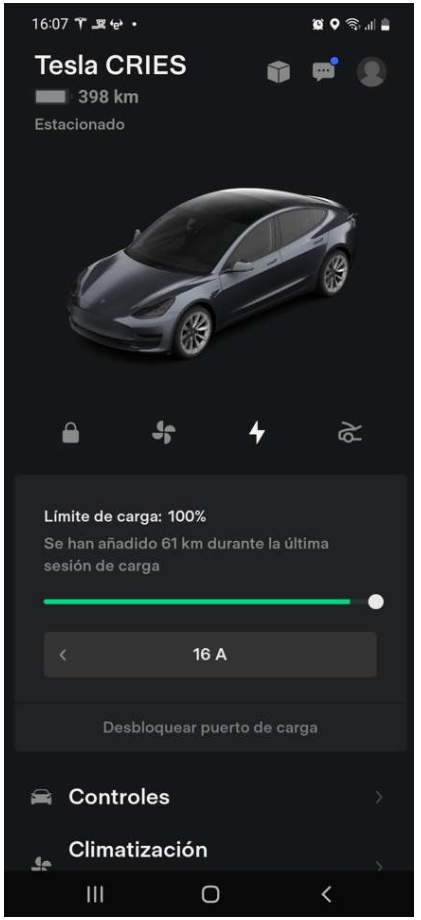


WORKSHOP FEST

Aplicaciones móvil para menejo remoto del vehículo







¿Qué es un vehículo conectado?

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

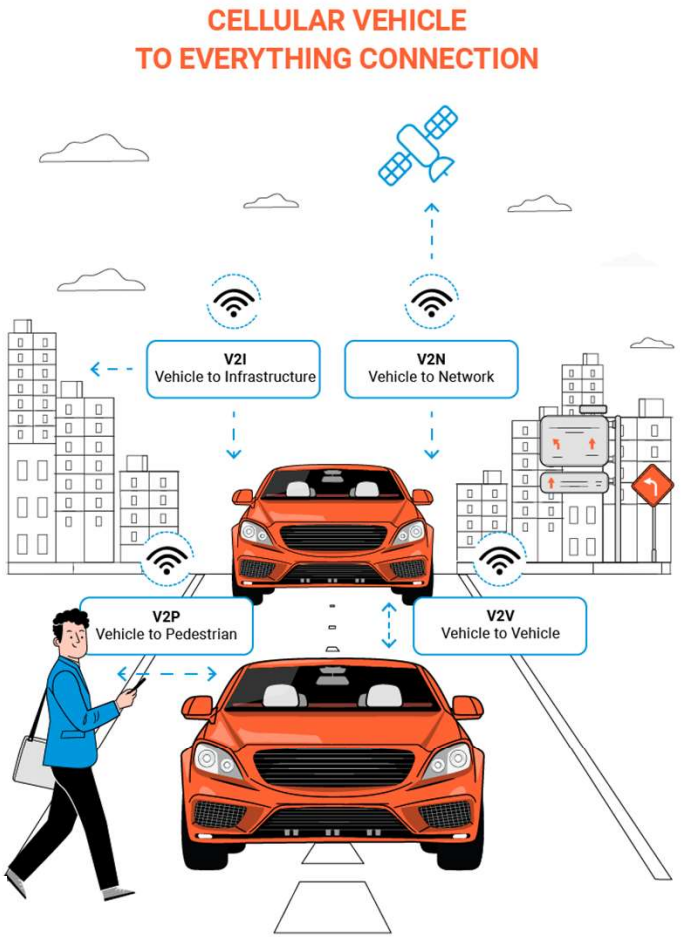
Electronic Control Units **ECUs** vs **IVC** In-Vehicle-Computer y el **IVI** In-Vehicle-Infotainment, ambos con capacidad **IoT**, mediante tarjeta **SIM**.

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

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

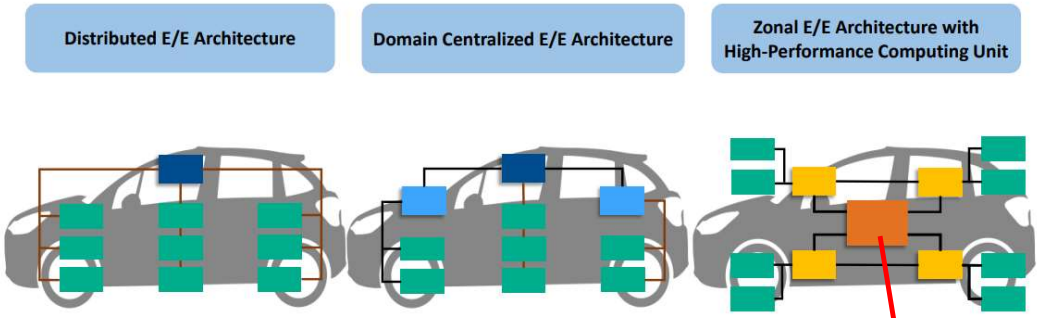
V2I Vehicle to Infrastructure
V2N Vehicle to Network
V2P Vehicle to Pedestrian
V2V Vehicle to Vehicle
V2G Vehicle to Grid
V2X Vehicle connected to everithing

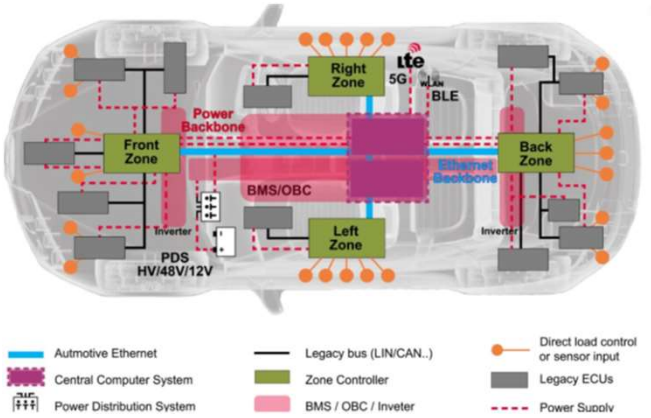
Ericsson Connected Vehicle Eviden - Connected Vehicle Connected vehicles video



WORKSHOP FEST

Evolución arquitectura de control de un coche

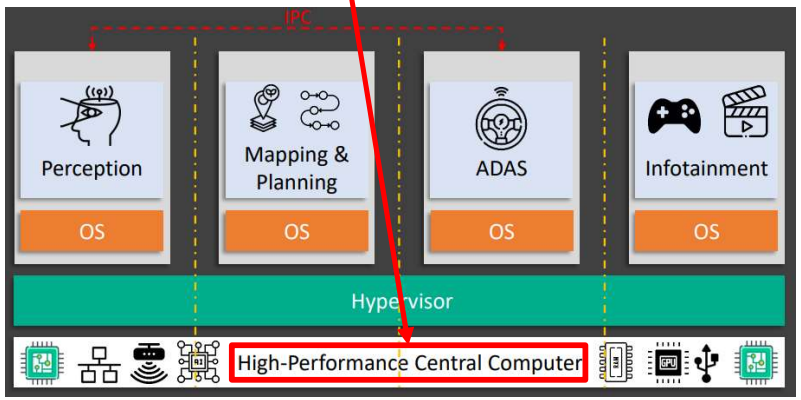


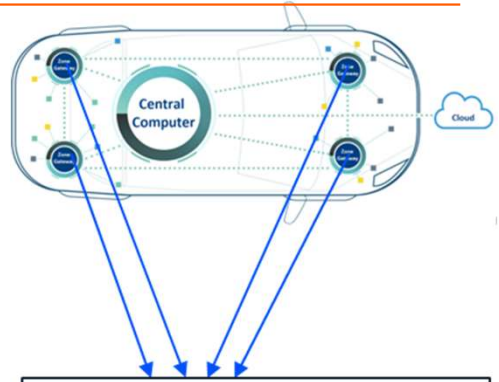


Zonal E/E will transform automotive architectures

PDS Proximity Detection System
BMS Battery Management System
OBC On Board Charging system

OBD2
On-Board Diagnostic II





Zone Gateways
(connectivity hub relaying data through a high-speed Ethernet link).

Hardware de un vehículo conectado Tesla

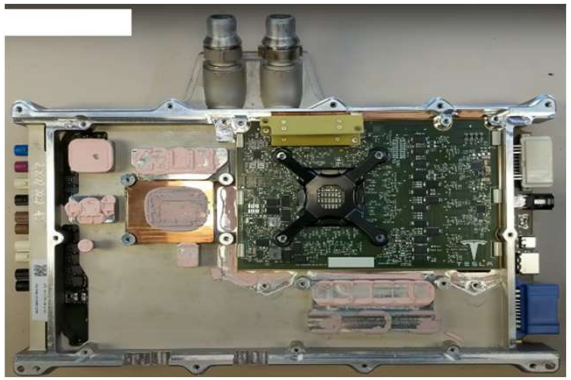
[Tesla’s new car on-board computer]

[Tesla's NEW HW4 Car Computer]

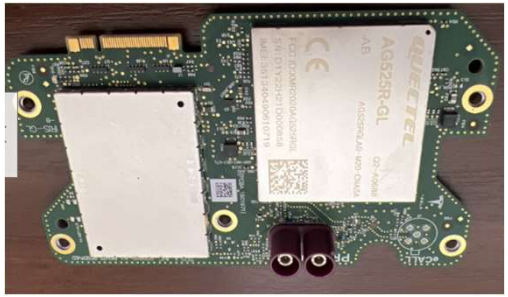
[Tesla Model S Plaid | ADAS Breakdown]

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

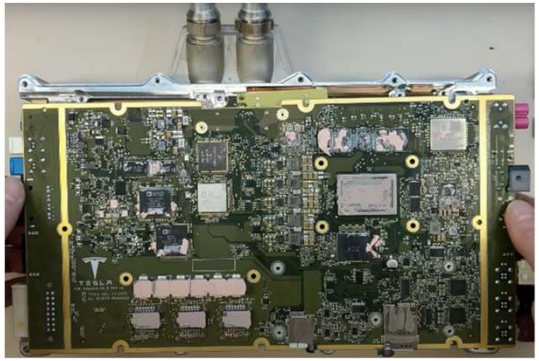
Computer back-side



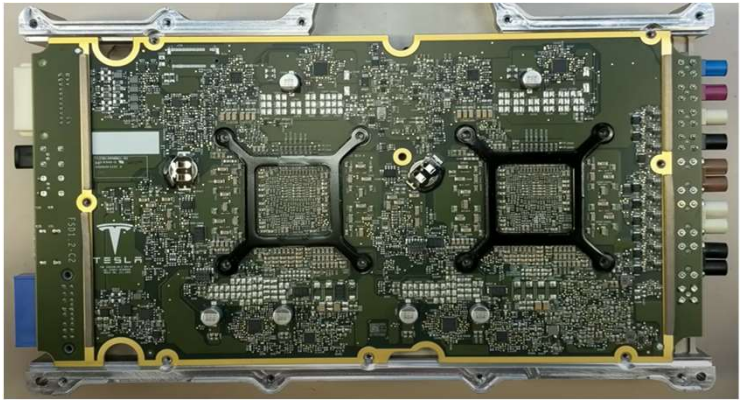
Graphic card



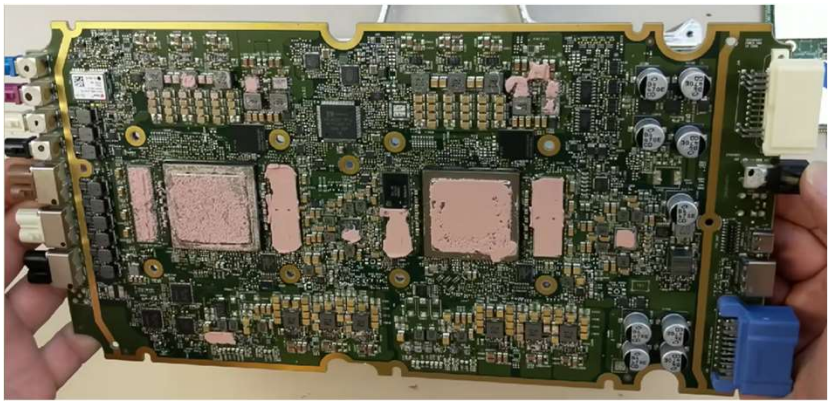
[Tesla Model 3 Y S X EU LTE 3G Modem, Adapter, Connectivity, QUECTEL AG525R PCBA]



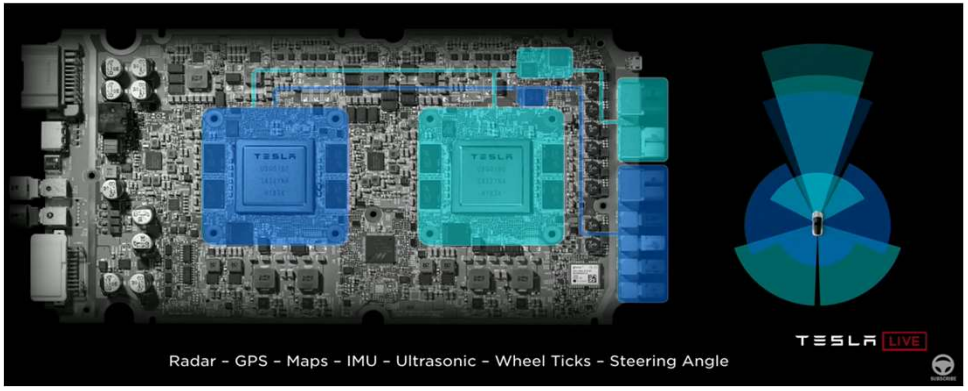
Hardware de un vehículo conectado



AutoPilot Computer back-side

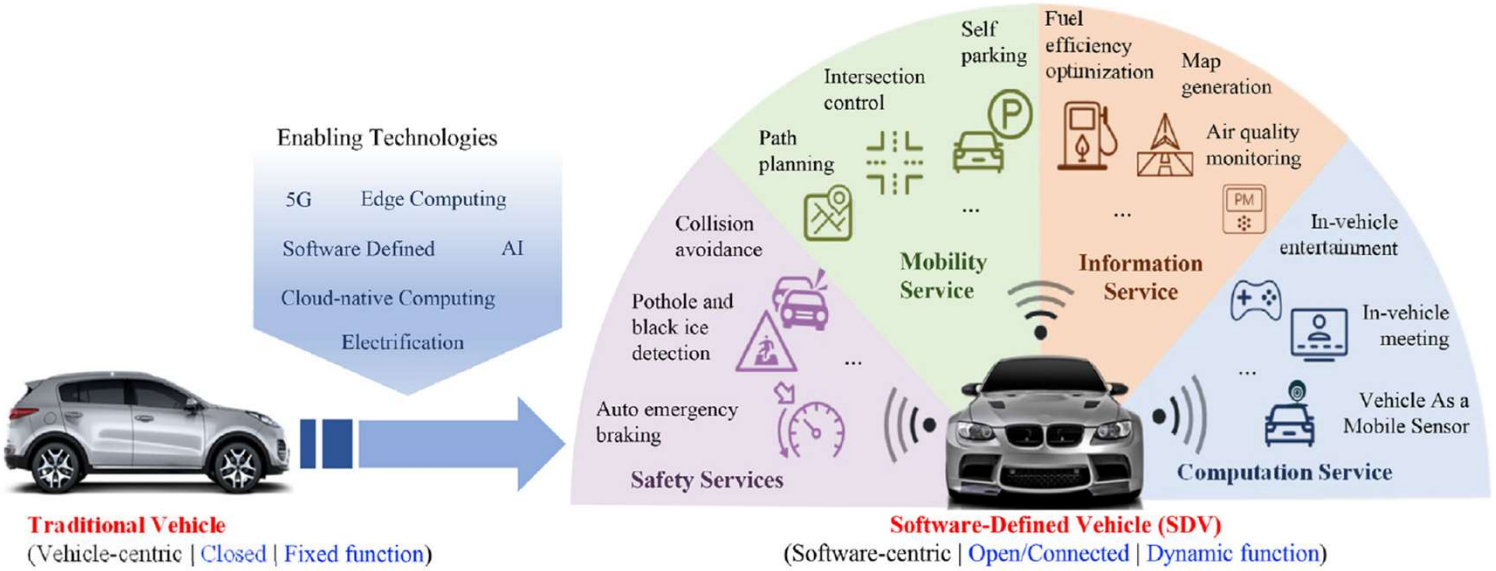


AutoPilot Computer front-side



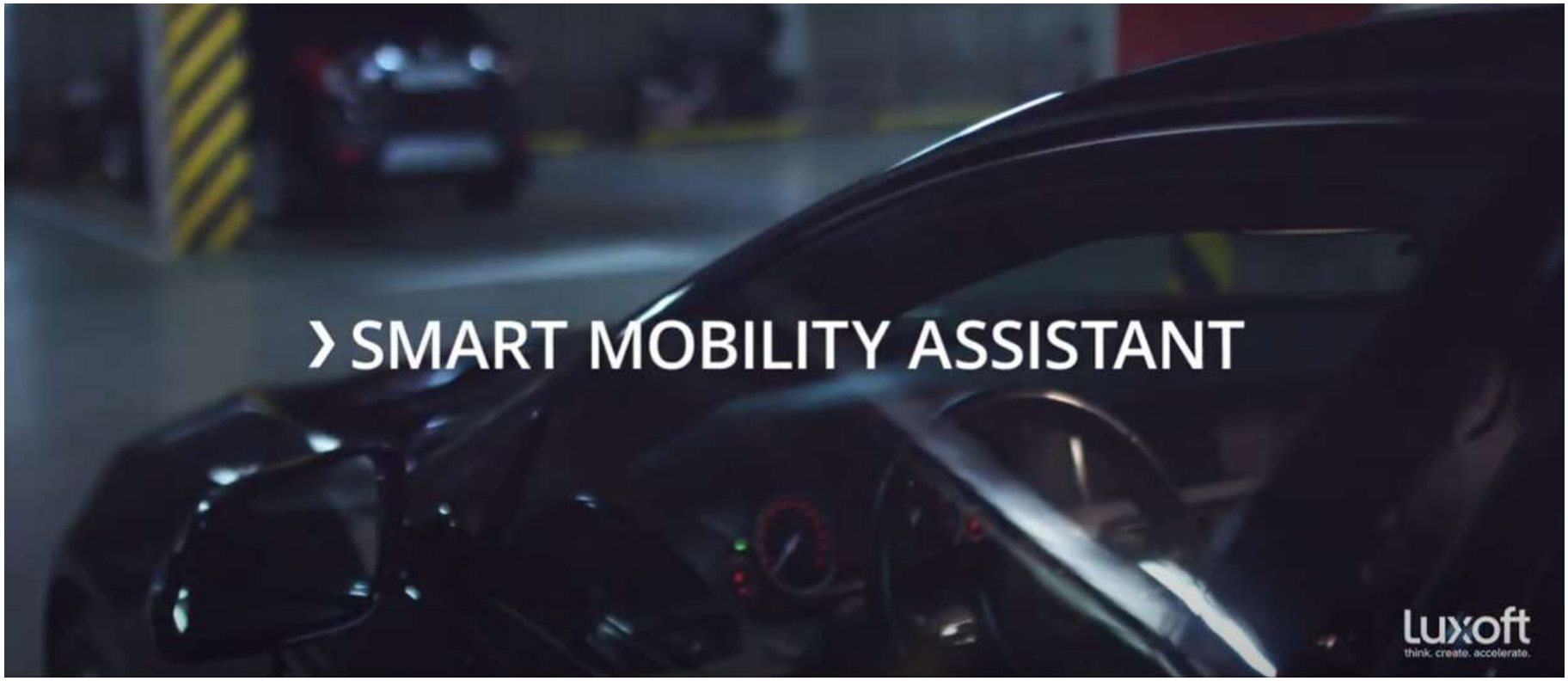
Vehículos definidos por software (SDV)

- 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) The Eclipse Foundation
- Sinergias/conexión con los **servicios en la nube**: AWS, Azure, Google Cloud, etc.
- Actualizaciones inalámbricas (**OTA**).



https://www.sciencedirect.com/science/article/pii/S2949715922000038
BlackBerry-SDV explained

Luxoft: asistente de movilidad



Smart Mobility Assistant | Luxoft Videos

WORKSHOP FEST

¿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**.

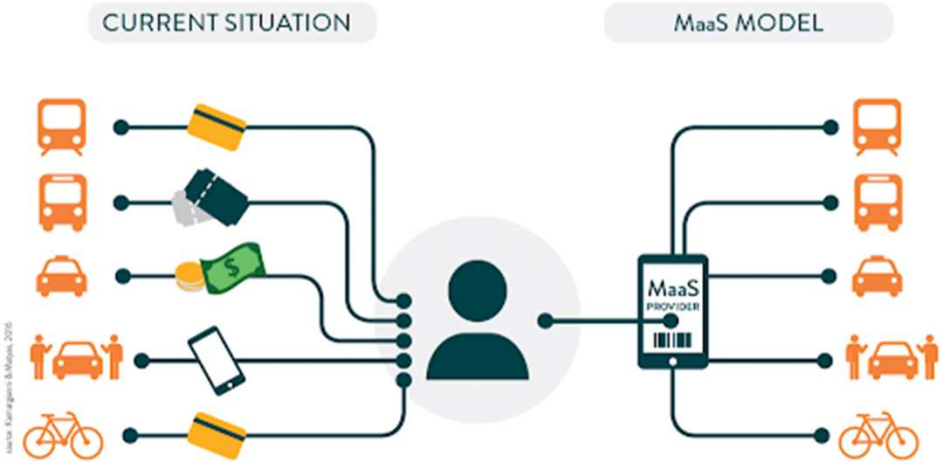
- "**Ventanilla única**" para diferentes medios de transporte. Una aplicación y un sistema de pago.

- **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

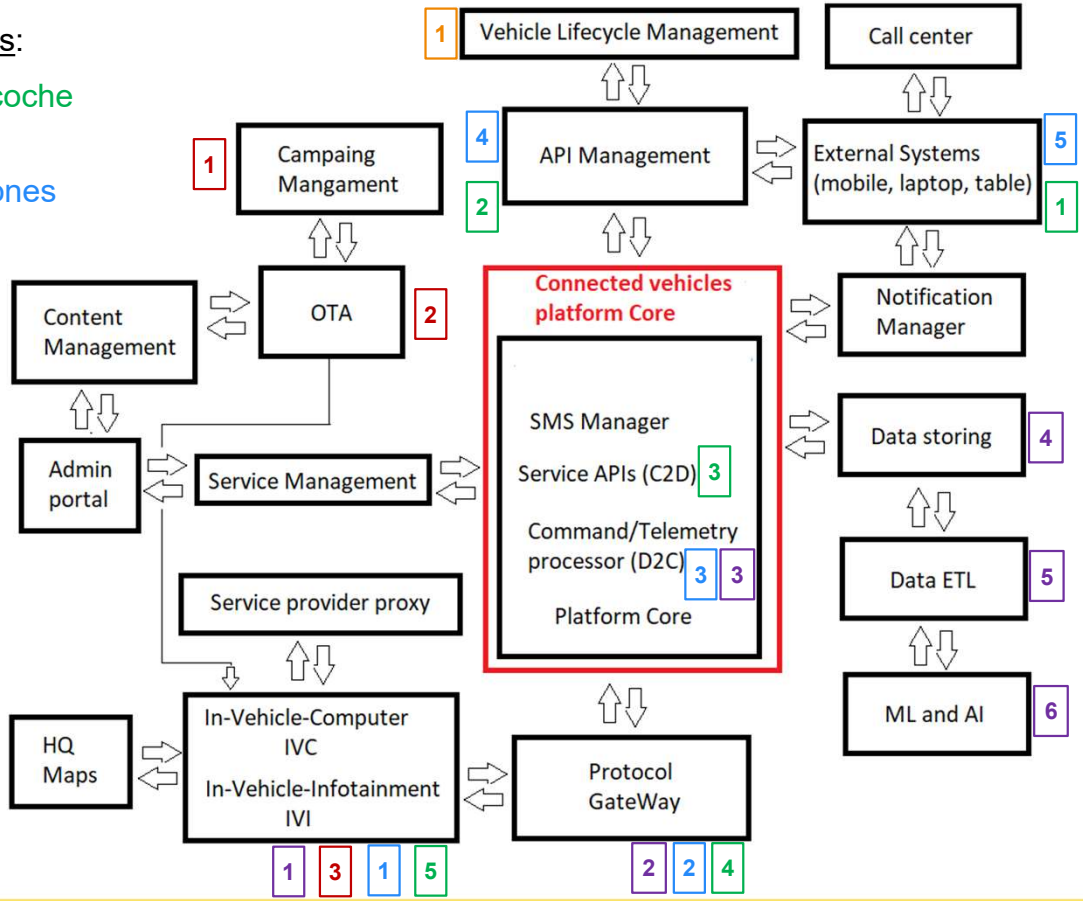
Arquitectura de una Plataforma de vehículos conectados

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

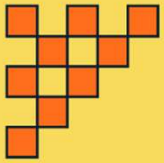


Requisitos de las plataformas de vehículos conectados:

- 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



WORKSHOP FEST

Cómo crear una Web API con .NET 8 para conectarnos a nuestro coche eléctrico Tesla Model 3

https://github.com/luiscoco/codemotion-Barcelona-2023-public-repo

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-8.0.100-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/

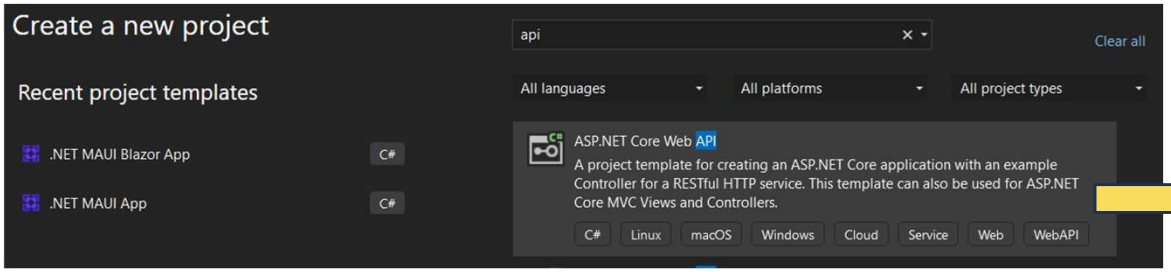
1.3. Instalación de **VSCode**: https://code.visualstudio.com/download

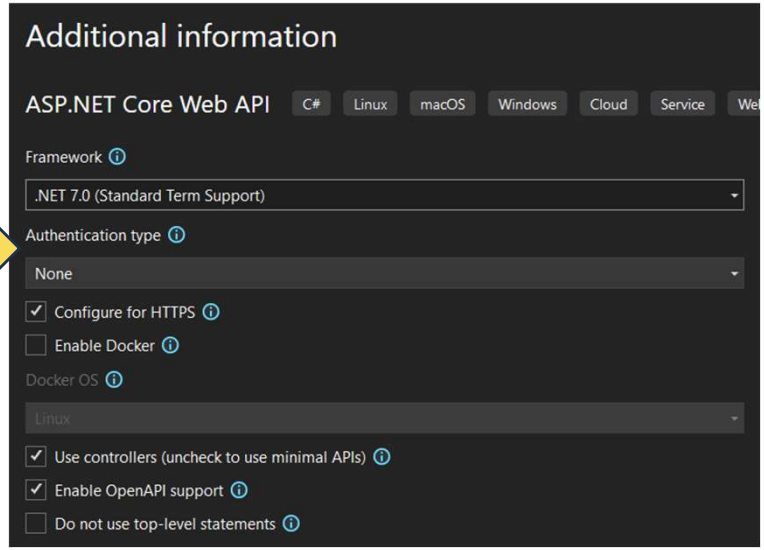
1.4. Instalación de **Docker Desktop**: https://docs.docker.com/desktop/install/windows-install/

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

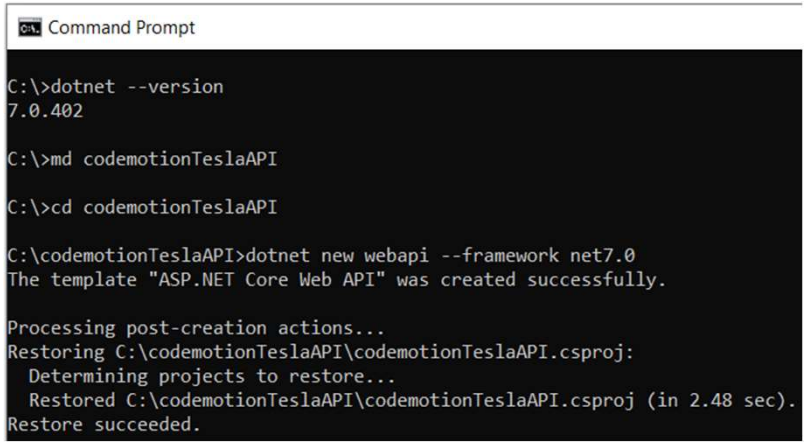
Paso 2. Creamos una aplicación .NET API

Opción 1. Utilizamos **Visual Studio 2022**





Opción 2. Utilizamos commandos **dotnet CLI**

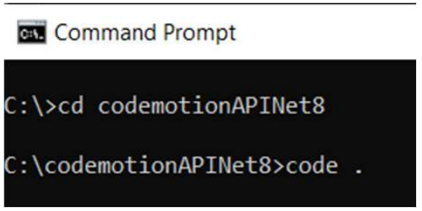


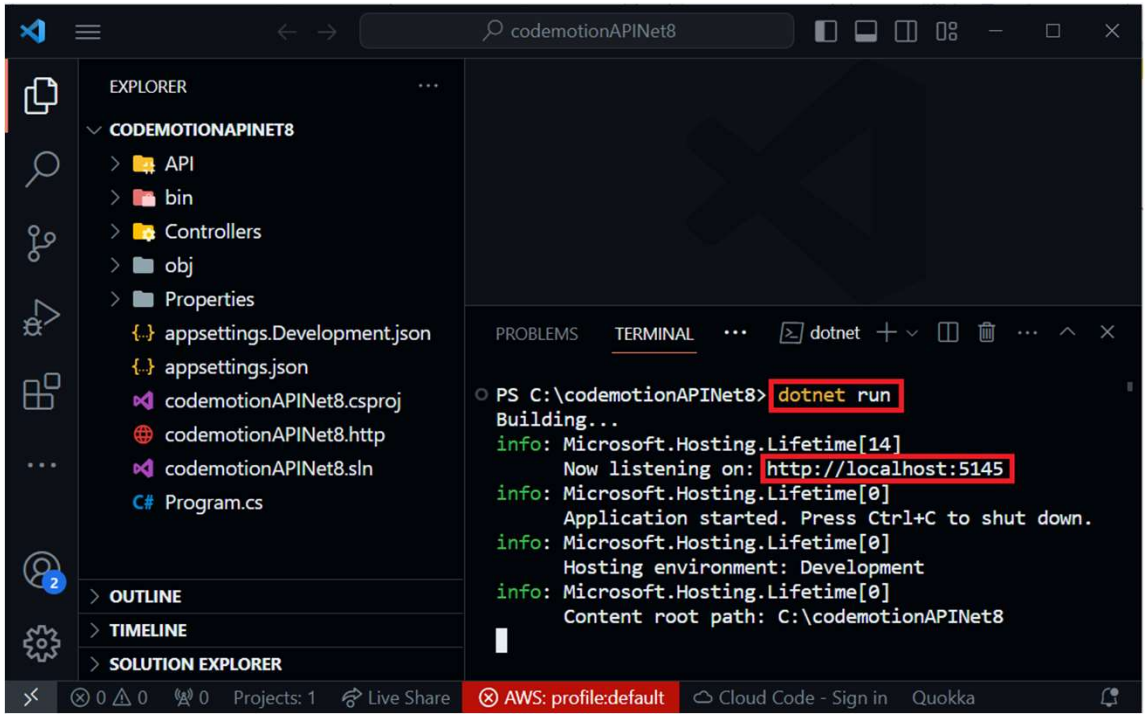

```
dotnet --version
dotnet new list
dotnet new --help webapi
md codemotiondotnet8api
cd codemotiondotnet8api
dotnet new webapi --framework net8.0
code .
```

Paso 3. 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**





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

Paso 4. 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**

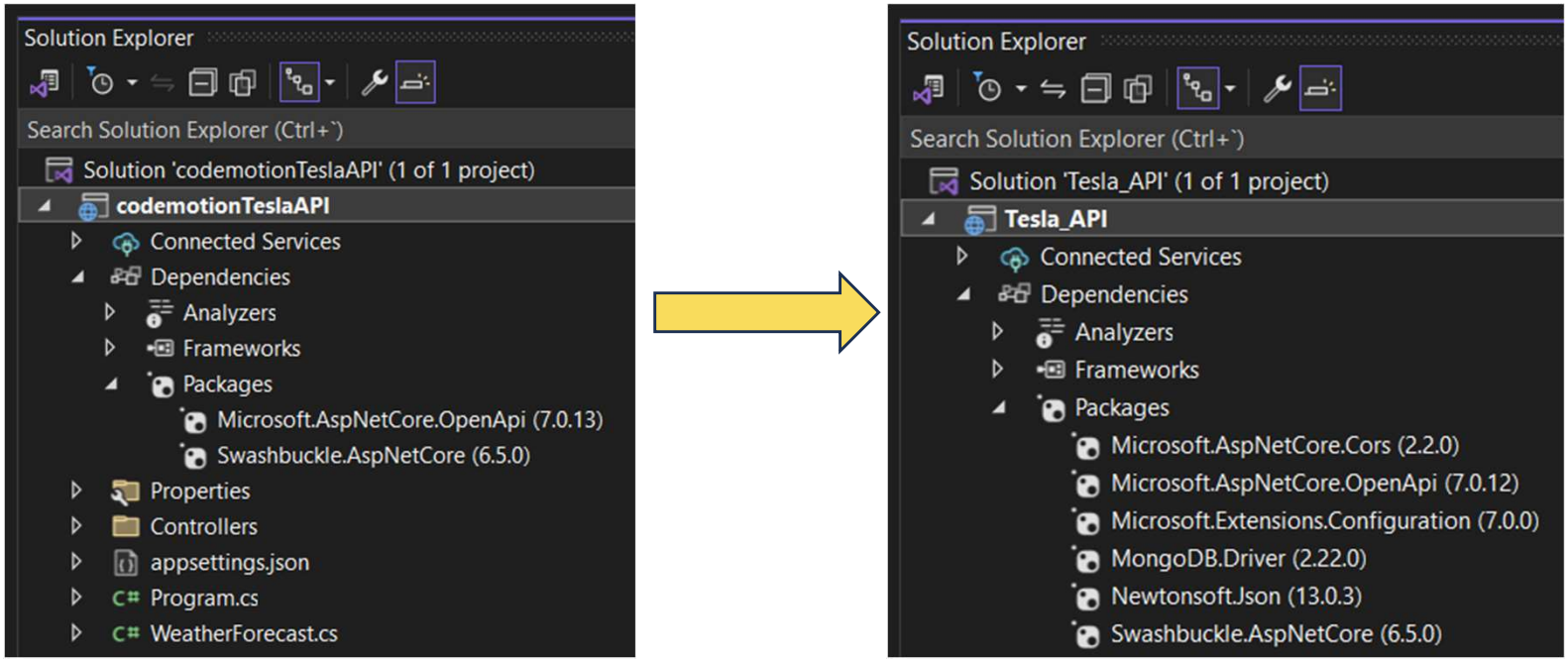
Verificamos que la instalación de la librerias ha sido correcta, consultando el archive **csproj**.

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

Paso 5. Instalación de dependencias

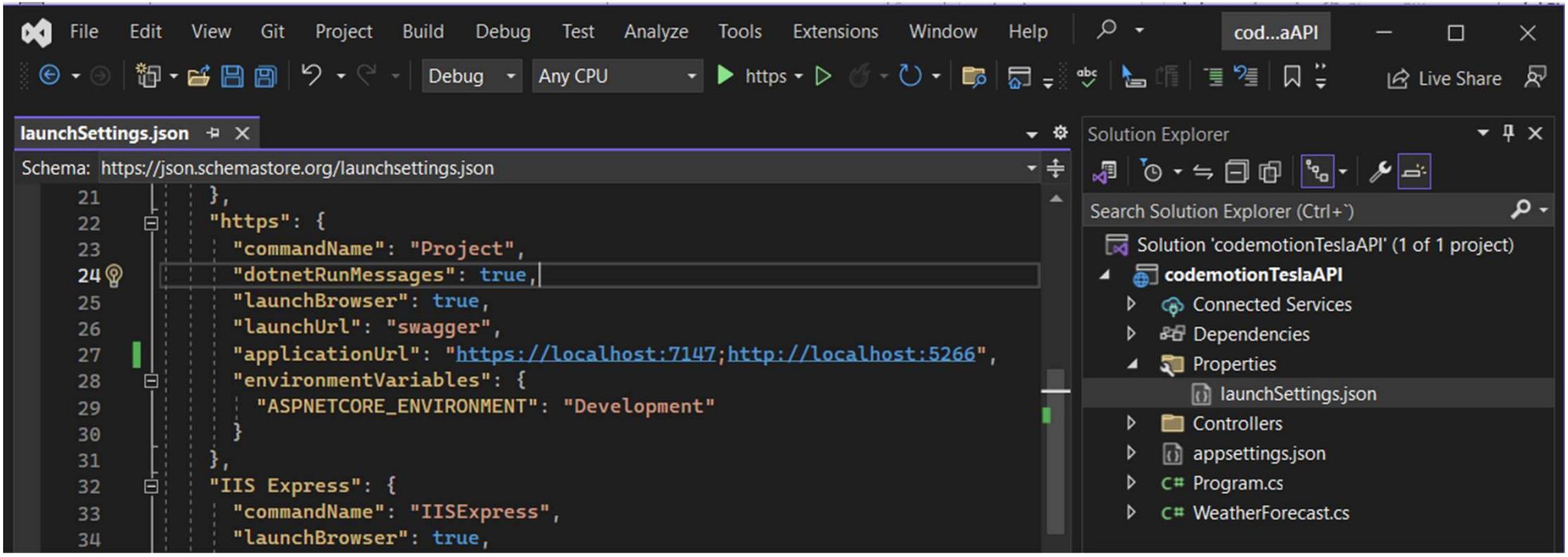
Instalamos las siguientes librerías con ayuda de **NuGet**:

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



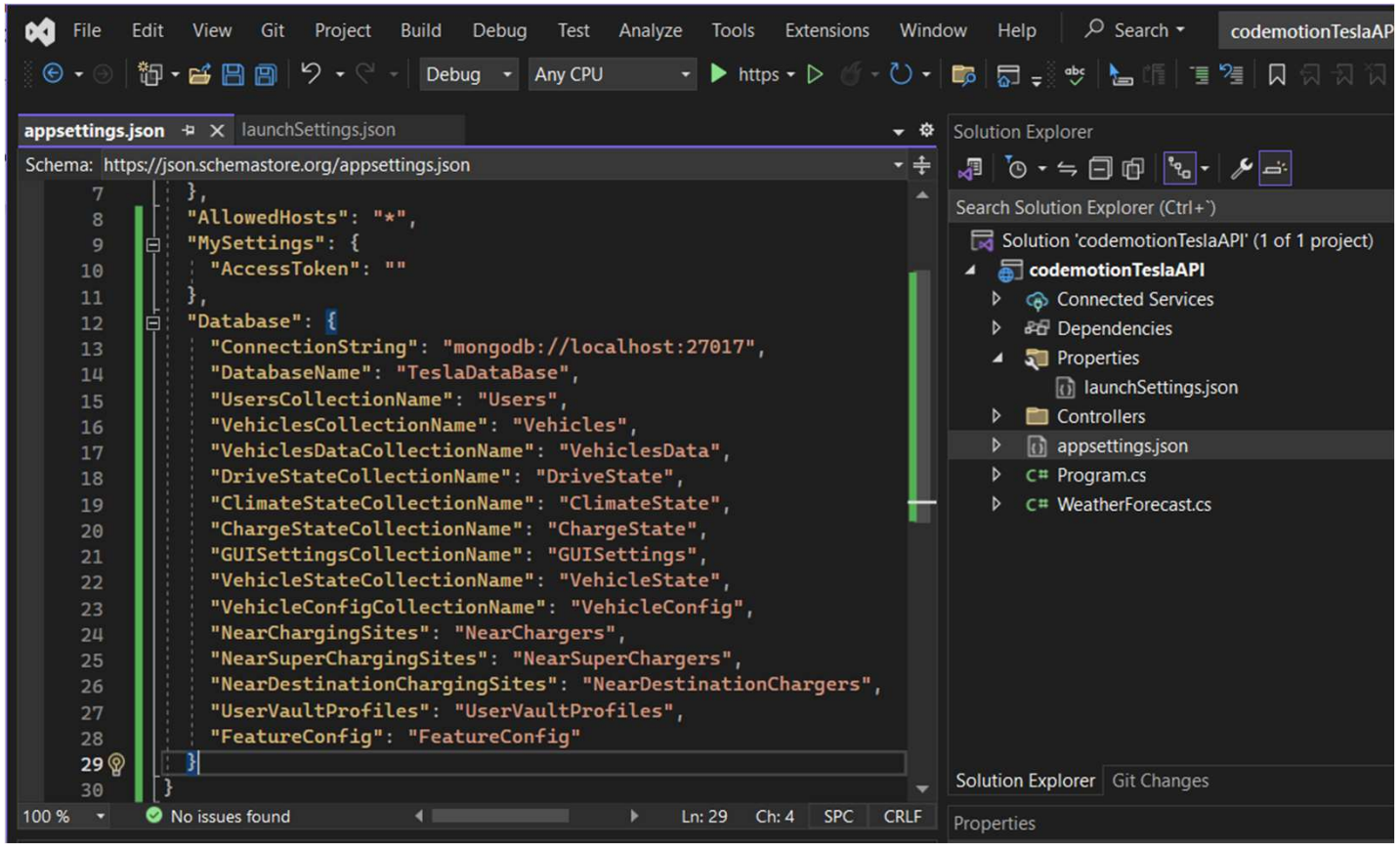
Paso 6. Modificamos el archivo “launchSettings.json”

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



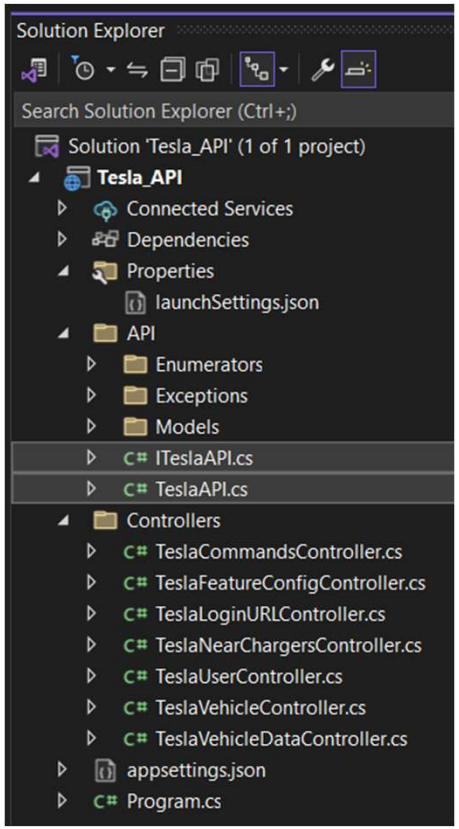
Paso 7. Modificamos el archivo “appSettings.json”

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



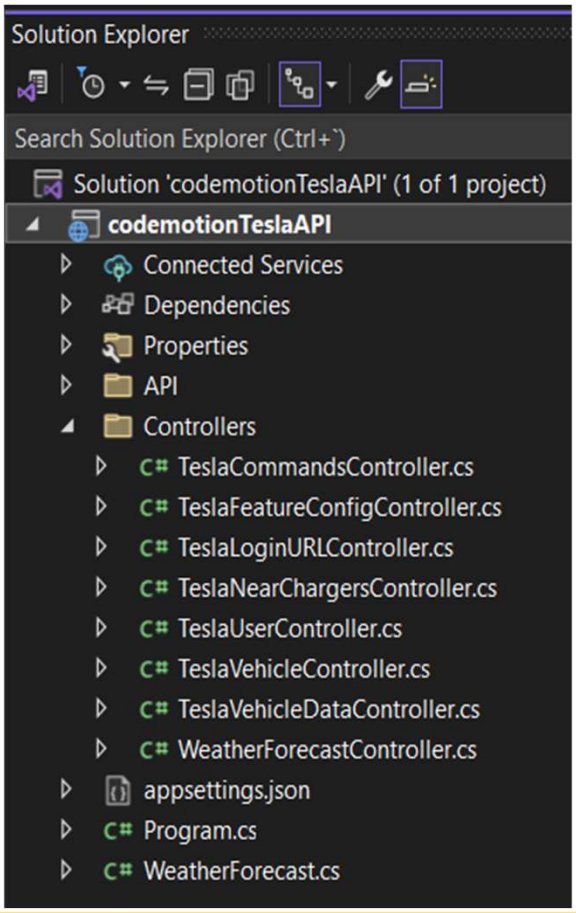
Paso 8. Creamos los modelos de datos y servicio “TeslaAPI”

Creamos una nueva carpeta “**API/Models**” donde ubicaremos los archivos incluyen los modelos de datos de la aplicación. En los archivos “**ITeslaAPI.cs**” y “**TeslaAPI.cs**” definimos los Servicios.



Paso 9. Creamos los controladores de la aplicación

Creamos una nueva carpeta “**Controllers**” donde ubicamos los controladores de nuestra aplicación.



Paso 10 (opcional). Creamos los controladores con inyección de dependecia del servicio ¨TeslaAPI¨

WORKSHOP FEST

```
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 10 (opcional). Creamos los controladores con inyección de dependecia del servicio ¨TeslaAPI¨

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

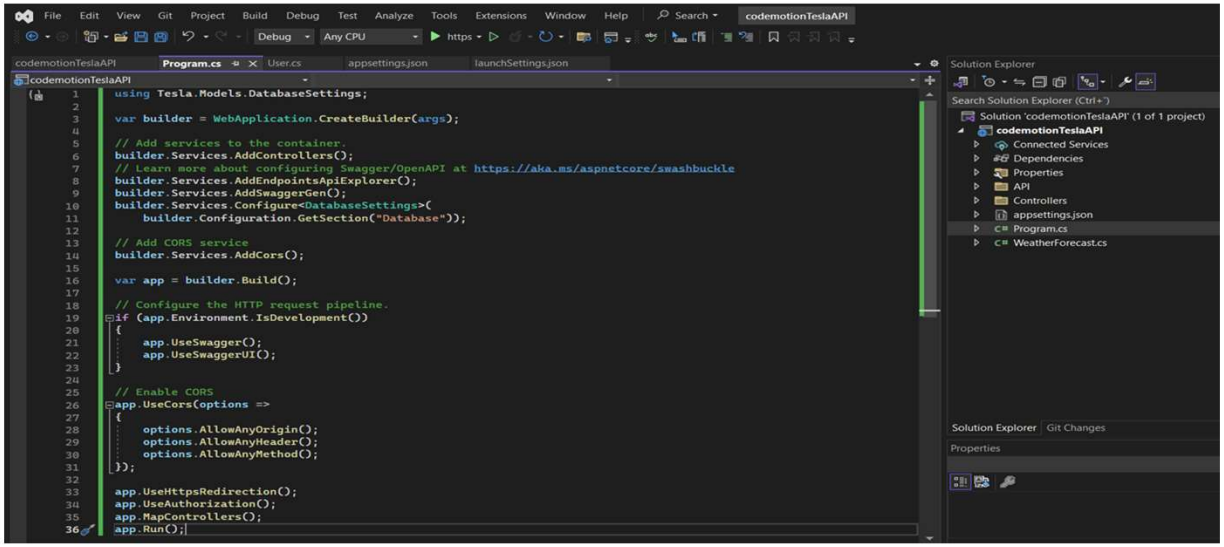
Capacidad de prueba: mejor capacidad de prueba al inyectar la interfaz ITeslaAPIService, puede sustituirla fácilmente con una implementación simulada durante las pruebas unitarias.

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 11. Modificar el archivo program.cs

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




```
using Tesla.Models.DatabaseSettings;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.Configure<DatabaseSettings>(
    builder.Configuration.GetSection("Database"));

// Add CORS service
builder.Services.AddCors();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

// Enable CORS
app.UseCors(options =>
{
    options.AllowAnyOrigin();
    options.AllowAnyHeader();
    options.AllowAnyMethod();
});

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

Si preferimos realizar la **injección de dependecias** del servicio **TeslaAPI** en los constructors de los controlladores, entonces incluir el siguientes código en el archive **program.cs**:

(opción 1)
services.AddScoped<ITeslaAPI, TeslaAPI>();

(opción 2)
services.AddTeslaApi();

Paso 12. 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:

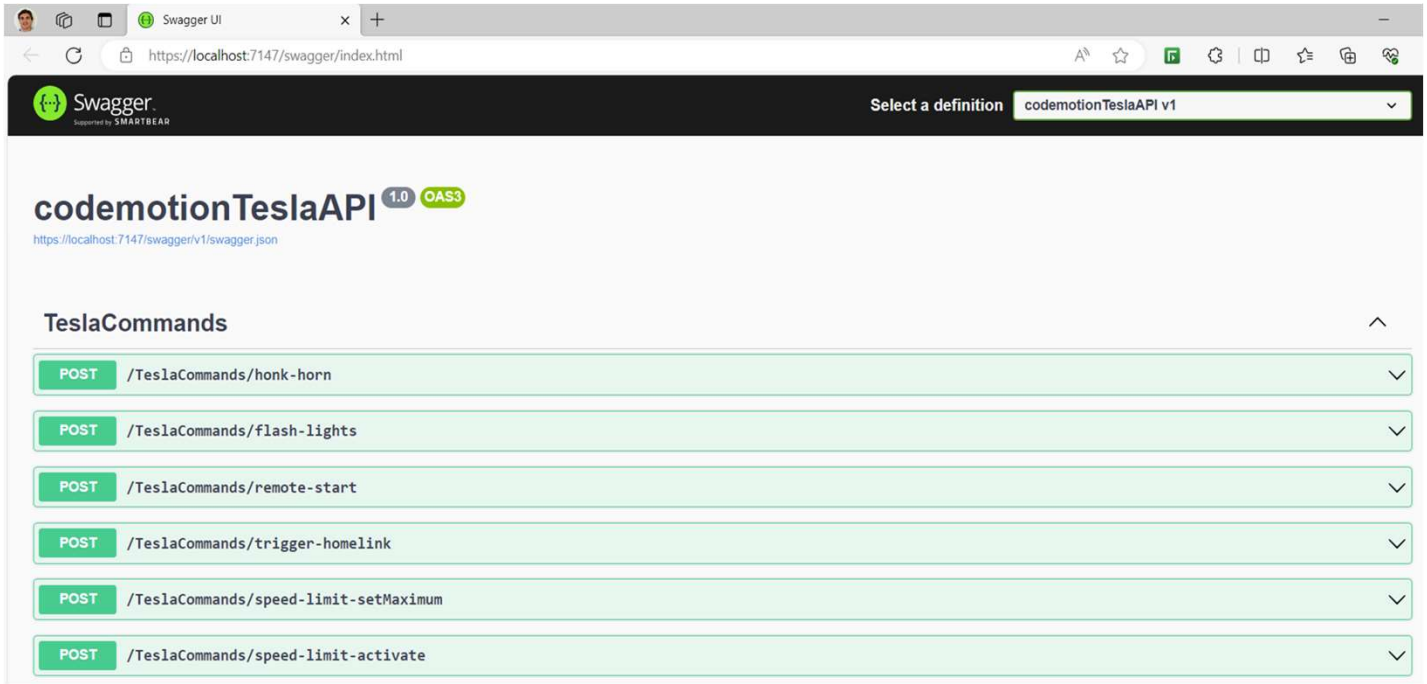
```
docker run -d --rm -p 27017:27017 --name mongo1 --network mongoCluster mongo:latest mongod --replSet myReplicaSet --bind_ip localhost,mongo1
docker run -d --rm -p 27018:27017 --name mongo2 --network mongoCluster mongo:latest mongod --replSet myReplicaSet --bind_ip localhost,mongo2
docker run -d --rm -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 13. Compilar y Ejecutar la aplicación

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

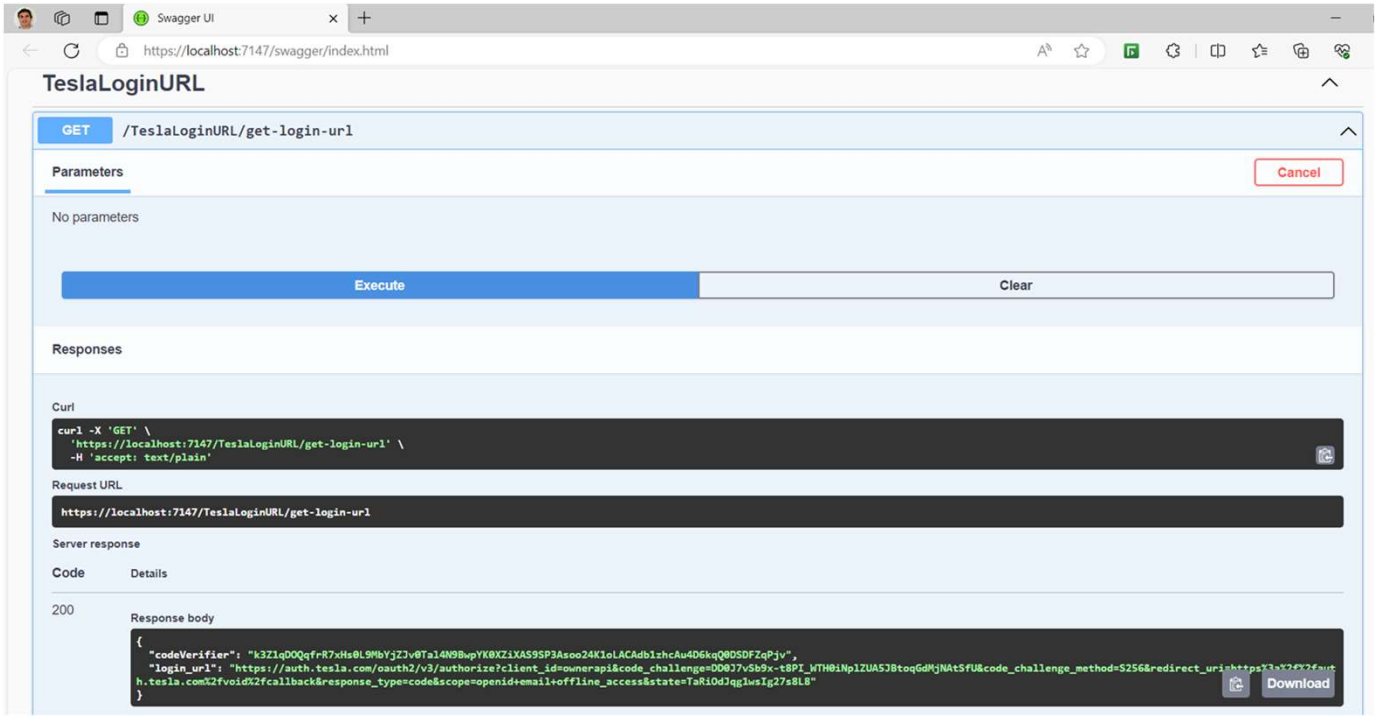
Compilamos y ejecutamos la aplicación .NET Web API con el comando: dotnet run



Paso 14. Login

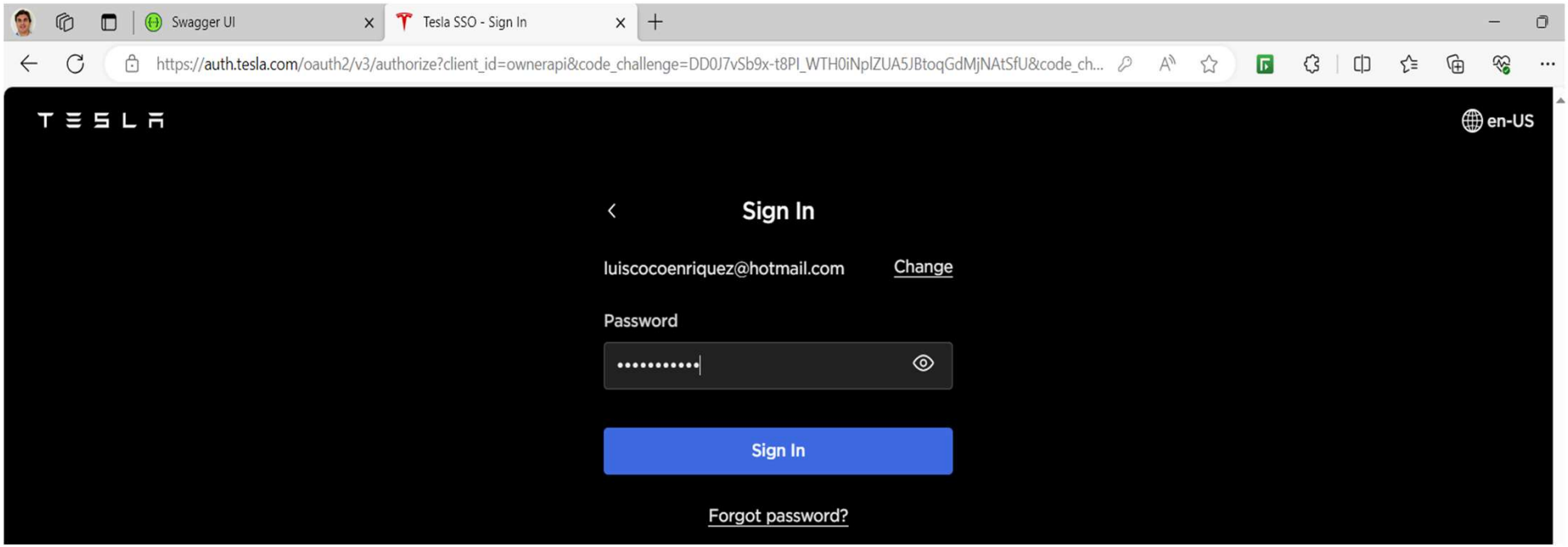
En este paso temenos que ejecutar la API “**/TeslaLoginURL/get-login-url**”. Como respuesta recibidos el “**codeVerifier**” y el “**login_url**”.

Copiamos el valor del “**login_url**” y lo pegamos en la barra de nuestro buscador de internet para acceder a la **página de Login de TESLA**.



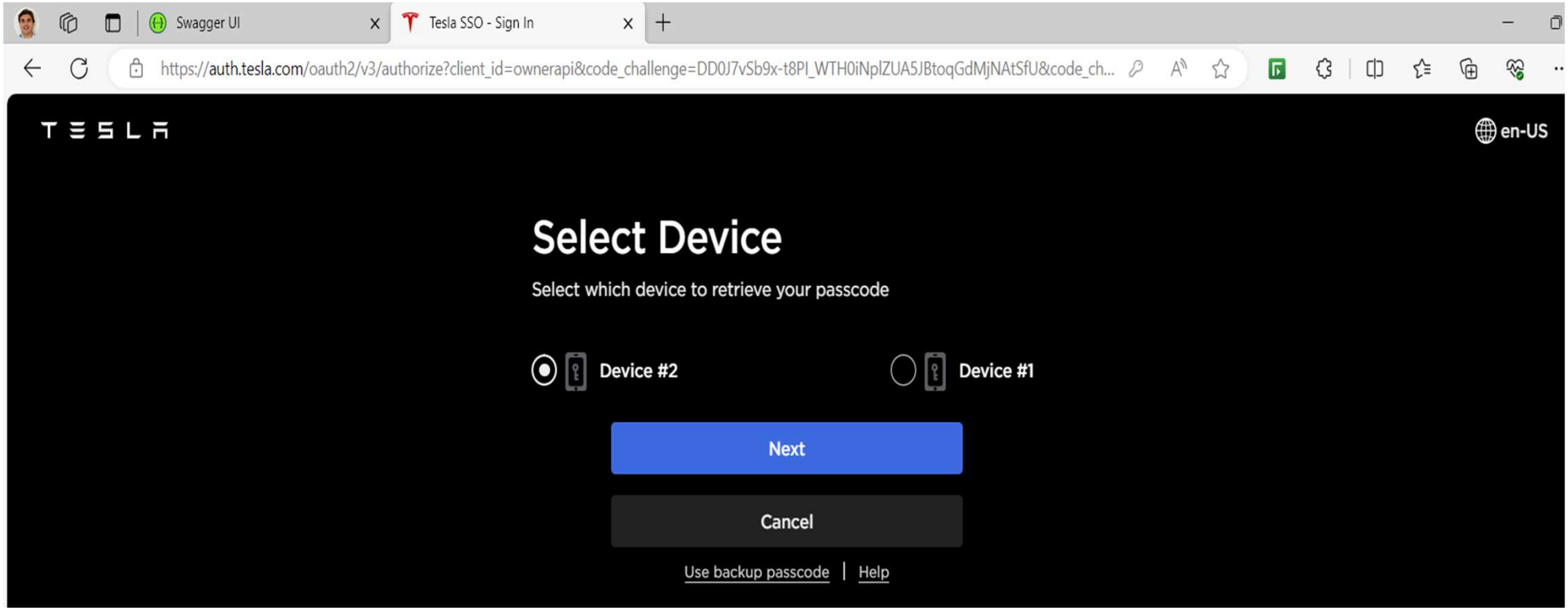
Paso 15. Continuamos con el Login

Introducimos nuestro nombre de usuario (dirección de correo electrónico) y el password de nuestra cuenta TESLA.



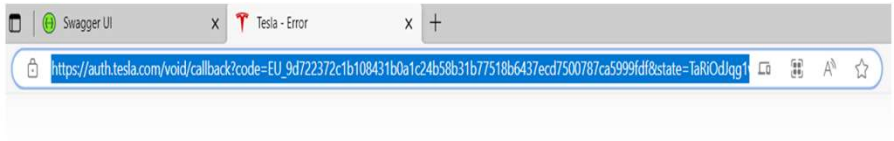
Paso 16. MFA(Multi-Factor Authentication)

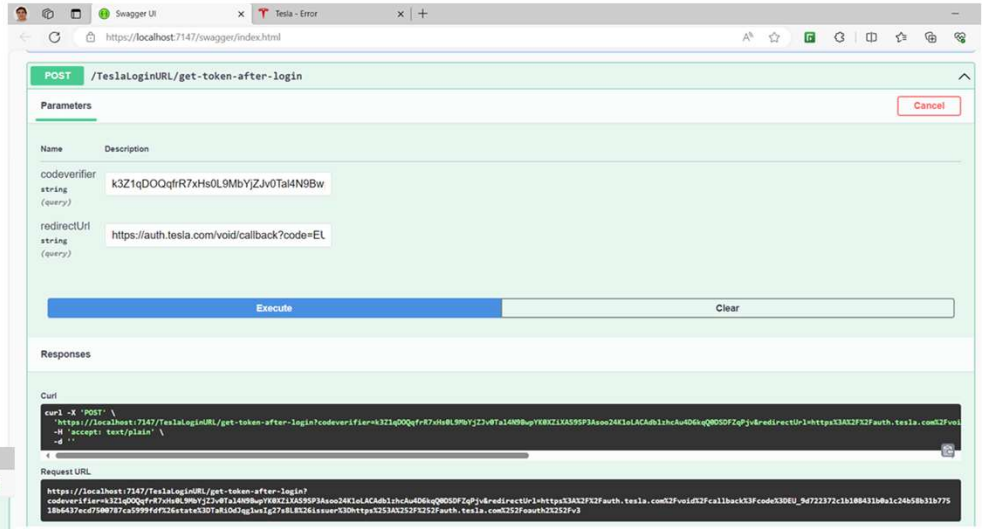
En caso de que hayamos activado el MFA en nuestra cuenta de TESLA, temenos que realizar la autenticación MFA. En mi caso activé el MFA con la aplicación ¨**Authenticator**¨ en mi teléfono móvil.

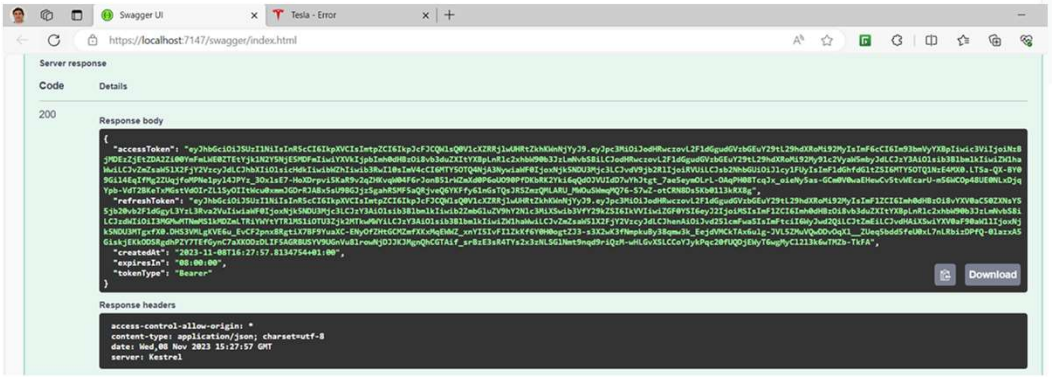


Paso 17. Obtenemos el access token

Copiamos la URL a la que nos ha redirigido la página de Login de TESLA y la pegamos en el campo ¨**redirectUrl**¨ y también cumplimentamos el campo ¨**codeverifier**¨, obtenido en el paso 14 de esta presentación. Una vez introducidos ambos campos ¨**redirectUrl**¨ y ¨**codeverifier**¨ ejecutamos la API ¨**TeslaloginURL/get-token-after-login**¨

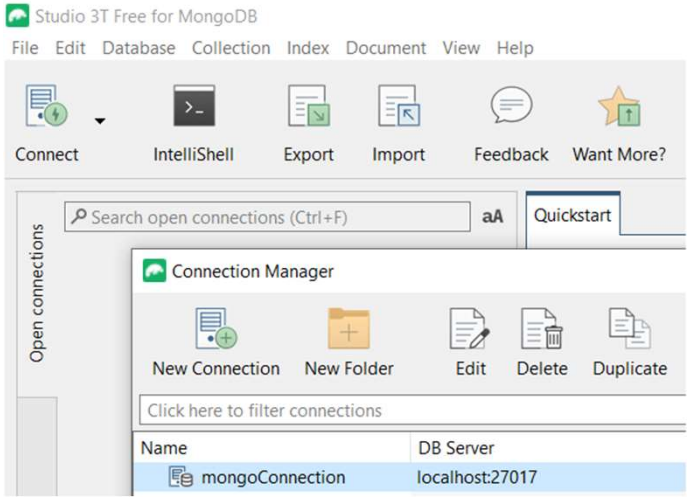


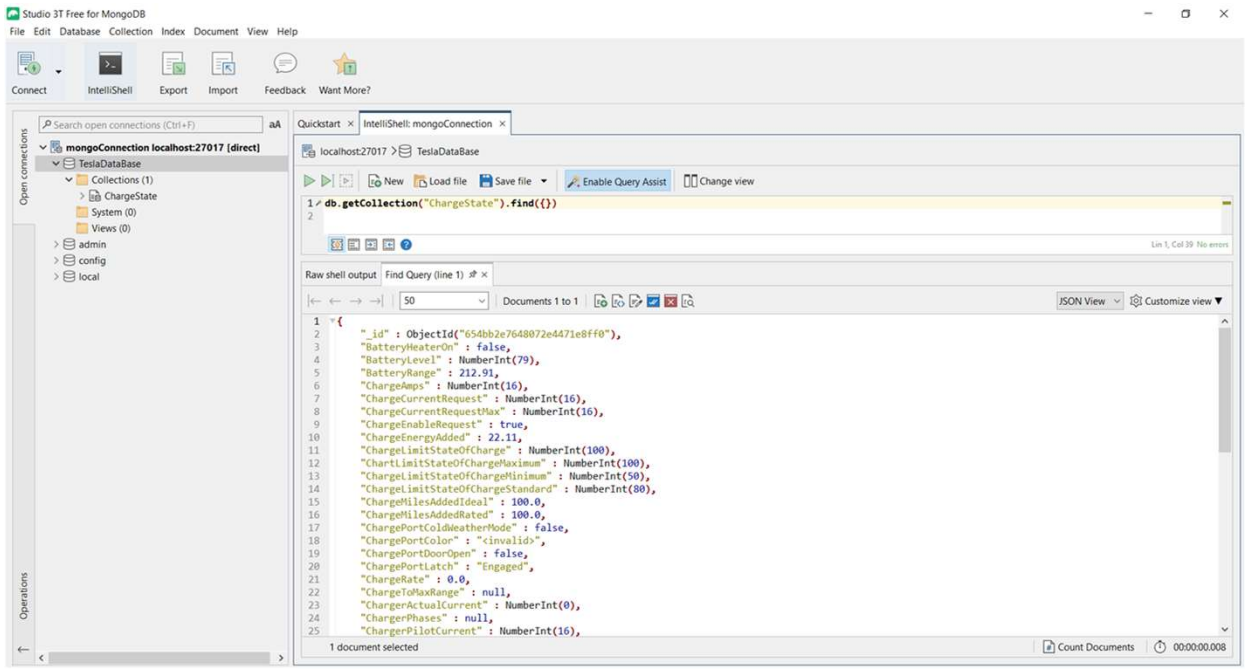




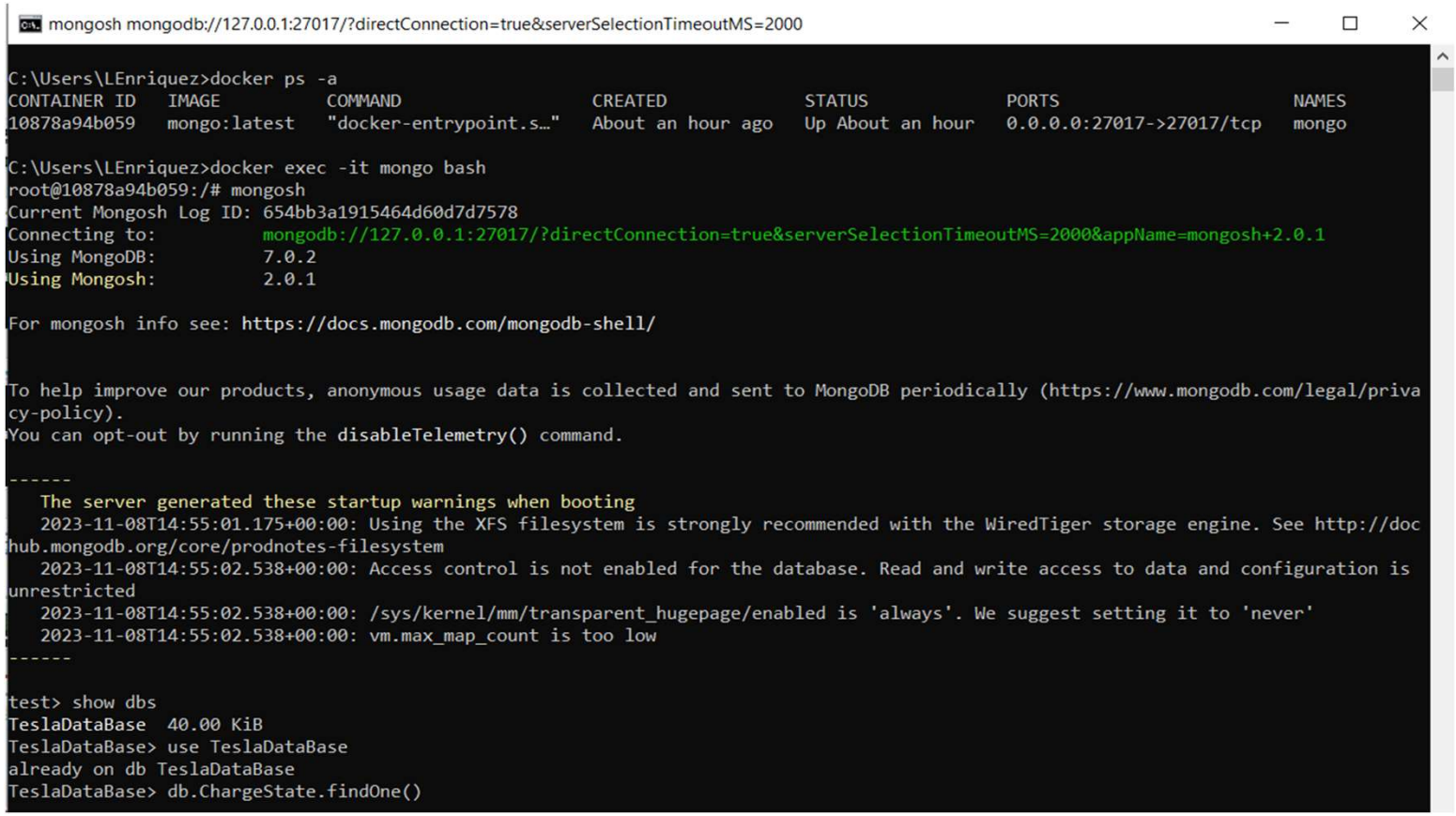
Esta función guarda en la variable **AccessToken** del **appSettings.json** el token que temenos que incluir en el Header de las APIs para tener acceso a la nube TESLA. Este token va a ser leido por todas las acciones de los controladores para solicitar la autorización de acceso a la API de Tesla.

Paso 18. Comprobamos los datos escritos en MongoDB con Studio 3T



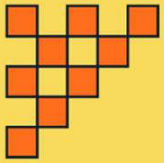


Paso 19. Comprobamos los datos escritos en MongoDB con comandos

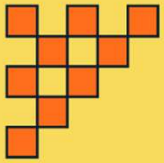


docker exec -it mongo bash

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



PREGUNTAS?



REPOSITORIOS DE CÓDIGO Y OTROS...

WORKSHOP FEST

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 mobilidad 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

WORKSHOP FEST

¡Contacta conmigo!

@GitHub

@Linkedin


WORKSHOP FEST
¡Gracias!

WORKSHOP FEST

¡Recuerda puntuar el Workshop!