# Guía paso a paso – HelloApi (.NET 8 + SQL Server en Docker)

Esta guía documenta cómo preparar el entorno, configurar la base de datos, ejecutar la API, crear migraciones, probar endpoints y conectarse desde TablePlus. Puedes copiar este contenido a un archivo de Word si lo necesitas.

## 1) Requisitos

* .NET 8 SDK (ARM64 para Mac M1/M2/M3)
* Docker Desktop
* (Opcional) TablePlus o Azure Data Studio para ver la BD

## 2) Levantar SQL Server en Docker

1. Crear contenedor (ejemplo con contraseña YourStrongPassword123):

* docker run -d -e ACCEPT\_EULA=Y -e MSSQL\_SA\_PASSWORD=YourStrongPassword123 \  
   -p 1433:1433 --name sqlserver mcr.microsoft.com/mssql/server:2022-latest

1. Verificar que corre:

* docker ps

## 3) Configurar la cadena de conexión

Editar appsettings.json en la raíz del proyecto para apuntar al SQL Server del contenedor:

{  
 "ConnectionStrings": {  
 "DefaultConnection": "Server=localhost;Database=MessageDb;User Id=sa;Password=YourStrongPassword123;TrustServerCertificate=True;"  
 },  
 "AllowedOrigins": [  
 "http://localhost:3000",  
 "https://myfrontend.com"  
 ]  
}

El proyecto lee esta cadena en Program.cs al registrar DbContext:

// Program.cs  
builder.Services.AddDbContext<AppDbContext>(options =>  
 options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

## 4) Migraciones de Entity Framework Core

1. Instalar herramientas EF (si hace falta):

* dotnet tool install --global dotnet-ef

1. Crear una migración (si aún no existe o quieres una nueva):

* dotnet ef migrations add InitialCreate  
  # o  
  dotnet ef migrations add NombreDeTuMigracion

1. Aplicar migraciones a la base de datos:

* dotnet ef database update

Estructura resultante (ejemplo) en la base MessageDb: - Tabla Messages con columnas: Id, MessageText, CreatedAt, UpdatedAt. - Tabla \_\_EFMigrationsHistory.

## 5) Ejecutar la API

Desde la carpeta del proyecto:

dotnet run

Observa en consola los puertos (típicamente http://localhost:5000 y https://localhost:5001).

Swagger UI: - http://localhost:5000/swagger - https://localhost:5001/swagger

## 6) Endpoints principales

Controlador: Controllers/MessageController.cs - GET /api/Message – Lista todos los mensajes - GET /api/Message/{id} – Obtiene un mensaje por id - POST /api/Message – Crea un mensaje - PUT /api/Message/{id} – Actualiza un mensaje - DELETE /api/Message/{id} – Elimina un mensaje

DTO para crear: Models/DTOs/MessageCreateDto.cs

public class MessageCreateDto  
{  
 public string Message { get; set; } = string.Empty;  
}

## 7) Probar guardado de datos

### Opción A: Swagger (recomendado)

1. Abrir Swagger UI.
2. Elegir POST /api/Message → Try it out.
3. Enviar body:

* {  
   "message": "Hola, este es un mensaje de prueba"  
  }

1. Recibirás 201 Created con el JSON del recurso creado.

### Opción B: Curl

* HTTP:
* curl -X POST "http://localhost:5000/api/Message" \  
   -H "Content-Type: application/json" \  
   -d '{"message":"Hola, este es un mensaje de prueba"}'
* HTTPS (si hay advertencia de certificado, agrega -k):
* curl -k -X POST "https://localhost:5001/api/Message" \  
   -H "Content-Type: application/json" \  
   -d '{"message":"Hola, este es un mensaje de prueba"}'

### Consultar datos

curl "http://localhost:5000/api/Message"  
# o  
curl -k "https://localhost:5001/api/Message"

## 8) Conectarse desde TablePlus

* Tipo: Microsoft SQL Server
* Host: 127.0.0.1
* Port: 1433
* User: sa
* Password: YourStrongPassword123
* Database: (primero master, luego cambias a MessageDb)
* SSL: Encrypt = Yes, Trust Server Certificate = Yes

Cadena ADO.NET de referencia:

Server=localhost,1433;Database=MessageDb;User Id=sa;Password=YourStrongPassword123;TrustServerCertificate=True;

## 9) Comandos útiles dentro del contenedor (sqlcmd)

Entrar al contenedor y usar sqlcmd (confiando el certificado):

docker exec -it sqlserver bash  
/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa -P YourStrongPassword123 -C

Consultas útiles:

SELECT name FROM sys.databases; -- Ver bases  
GO  
USE MessageDb; -- Cambiar de base  
GO  
SELECT \* FROM INFORMATION\_SCHEMA.TABLES; -- Ver tablas  
GO  
SELECT \* FROM Messages; -- Ver registros  
GO

## 10) Solución de problemas frecuentes

* Error 18456 (login failed): verifica usuario sa y contraseña; intenta conectar a master primero.
* Certificados SSL: activa “Trust Server Certificate” o usa -C en sqlcmd y -k en curl con HTTPS.
* Contenedor no corre: docker start sqlserver y docker ps.
* Error de cadena de conexión: confirmar appsettings.json y puertos expuestos 1433.

## 11) Arquitectura (resumen)

* Program.cs: registra controladores, Swagger, CORS, y AppDbContext.
* Data/AppDbContext.cs: contexto EF Core.
* Models/Message.cs: entidad de dominio.
* Models/DTOs/: DTOs para crear/leer/actualizar.
* Repositories/: IMessageRepository y MessageRepository para acceso a datos.
* Services/: IMessageService y MessageService con lógica de negocio.
* Controllers/MessageController.cs: expone endpoints REST.

## 12) Flujo de trabajo típico

1. Levantar SQL Server en Docker.
2. Ajustar appsettings.json (conexión).
3. dotnet ef database update para aplicar migraciones.
4. dotnet run para iniciar la API.
5. Probar con Swagger o curl.
6. Ver datos en TablePlus/Azure Data Studio.

¡Listo! Con esto tienes el paso a paso del proyecto y cómo operarlo.