**https://docs.microsoft.com/en-us/learn/modules/build-web-api-minimal-database/4-add-sqlite-database-provider**

**Use the SQLite database provider with EF Core**

One of the benefits of performing database access through an abstraction layer like Entity Framework (EF) Core is that it decouples your application from the database provider. You can change the database provider without rewriting your database access code. You shouldn't expect to be able to switch database providers without any effect to your application code, but the changes will be minimized and localized.

A related advantage of using EF Core is that you can reuse your code, experience, and data access libraries to work with any other EF Core database provider.

For this tutorial, you'll use [SQLite database](https://www.sqlite.org/index.html), but you might also use one that works better for you. EF Core currently supports more than 20 database providers. The providers are listed in the [documentation](https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli%3Fazure-portal%3Dtrue" \t "az-portal).

In general, you'll use the following steps to implement a new database provider:

1. Add one or more NuGet packages to your project to include the database provider.
2. Configure the database connection.
3. Configure the database provider in the ASP.NET Core services.
4. Perform database migrations.

* : Can access many different databases through plug-in libraries called [database providers](https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli%3Fazure-portal%3Dtrue" \t "az-portal). The following package is the SQLite database provider for Entity Framework (EF) Core.

**SQLite EF Core Database Provider** (applies for these [database providers](https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli%3Fazure-portal%3Dtrue" \t "az-portal). )

dotnet add package Microsoft.EntityFrameworkCore.Sqlite --version 6.0

**EF Core tools**

dotnet tool install --global dotnet-ef

**Microsoft.EntityFrameworkCore.Design**

dotnet add package Microsoft.EntityFrameworkCore.Design --version 6.0

To enable database creation, you need to complete two steps:

1. Set the database connection string.
2. Migrate your data model to a SQLite database.

**Set the database connection string**

En appsetting.json

  "ConnectionString": {

      "Conn": "-----"

  },

In *Program.cs*, under your app builder

Elimine

implementation builder.Services.AddDbContext<PizzaDb>(options => options.UseInMemoryDatabase("items"));

Agregue

using Microsoft.EntityFrameworkCore;

using Test.Models;

var builder = WebApplication.CreateBuilder(args);

var Conn = builder.Configuration.GetConnectionString("Conn");

Para SQLite

var connectionString = builder.Configuration.GetConnectionString("Conn") ?? "Data Source=Pizzas.db";

Para SQL Server (PizzasDb es Pizzas DbContext)

builder.Services.AddDbContext<**PizzasDb**>(options =>

    options.UseSqlServer(Conn));

**Migrate your data model**

With the EF Core migration tool, you can now start your first migration, InitialCreate. In a terminal window, run the migrations command:

dotnet ef migrations add InitialCreate

EF Core creates a folder named *Migrations* in your project directory that contains two files with the code that represents the database migrations.

**Create your database and schema**

Now that you've completed the migration, you can use it to create your database and schema.

In a terminal window, run the following database update command to apply migrations to a database:

**CREARA, ACTUALIZARA LA BASE EN SQLITE**

dotnet ef database update

You should see a newly created *Pizzas.db* file in your project directory.

**Run and test the application**

Now that you have a backing database, your changes will be persisted.

run the dotnet run command in your terminal window and interact with the API in the Swagger UI.

You can stop the application by using the **Ctrl+C** command. Use **Cmd+C** on macOS. Then run it again and verify that your changes are still persisted.

MsSQL

Instale:

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Tools