Usuario GitHub: **serraguti**

Hot Reload. Lo que permite es visualizar las Vistas en ejecución y ver los cambios sin volver a cargar, simplemente guardando.

Es una característica que está activada ya por defecto y es igual a lo que sucede con React o Angular o Vue.

Utilizar una aplicación de Xamarin. Es una aplicación Multiplataforma que está orientada a dispositivos móviles.

Xamarin ya no existe (según Microsoft), ahora está Maui. Todavía no han sacado la versión, pero las diferencias son mínimas, simplemente que permite también incluir Apps para Linux o MacOS.

Vamos a trabajar con un Api publicado que consumirá una base de datos de Azure.

Utilizaremos un CRUD y os iré poniendo en los proyectos, este documento y también el Script de bases de datos.

Comenzamos creando un nuevo proyecto Net Core Web Api llamado **ApiProductosLabs**

Agregamos los Nuget de acceso a datos

Texto

Descripción generada automáticamente

Creamos una carpeta llamada **Models** y una clase llamada **Producto**

**PRODUCTO**

{

[Table("PRODUCTOSLABS")]

public class Producto

{

[Key]

[Column("IDPRODUCTO")]

public int IdProducto { get; set; }

[Column("NOMBRE")]

public string Nombre { get; set; }

[Column("DESCRIPCION")]

public string Descripcion { get; set; }

[Column("PRECIO")]

public int Precio { get; set; }

[Column("IMAGEN")]

public string Imagen { get; set; }

}

Creamos una carpeta llamada **Data** y una clase llamada **ProductosContext**

**PRODUCTOSCONTEXT**

public class ProductosContext: DbContext

{

public ProductosContext(DbContextOptions<ProductosContext> options) : base(options) { }

public DbSet<Producto> Productos { get; set; }

}

Creamos una carpeta llamada **Repositories** y una clase llamada **RepositoryProductos**

**REPOSITORYPRODUCTOS**

public class RepositoryProductos

{

private ProductosContext context;

public RepositoryProductos(ProductosContext context)

{

this.context = context;

}

public List<Producto> GetProductos()

{

return this.context.Productos.ToList();

}

public Producto FindProducto(int id)

{

return this.context.Productos.FirstOrDefault(x => x.IdProducto == id);

}

public int GetMaxIdProducto()

{

if (this.context.Productos.Count() == 0)

{

return 1;

}

else

{

return this.context.Productos.Max(x => x.IdProducto) + 1;

}

}

public async Task AddProducto(string nombre, string descripcion, int precio, string imagen)

{

Producto producto = new Producto()

{

Nombre = nombre,

IdProducto = this.GetMaxIdProducto(),

Descripcion = descripcion,

Precio = precio,

Imagen = imagen

};

await this.context.Productos.AddAsync(producto);

await this.context.SaveChangesAsync();

}

public async Task UpdateProducto(int id, string nombre, string descripcion

, int precio, string imagen)

{

Producto producto = this.FindProducto(id);

producto.Nombre = nombre;

producto.Descripcion = descripcion;

producto.Precio = precio;

producto.Imagen = imagen;

await this.context.SaveChangesAsync();

}

public async Task DeleteProducto(int id)

{

Producto producto = this.FindProducto(id);

this.context.Productos.Remove(producto);

await this.context.SaveChangesAsync();

}

}

Sobre la carpeta **Controllers**, agregamos un nuevo controlador llamado **ProductosController**

**PRODUCTOSCONTROLLER**

[Route("api/[controller]")]

[ApiController]

public class ProductosController : ControllerBase

{

private RepositoryProductos repo;

public ProductosController(RepositoryProductos repo)

{

this.repo = repo;

}

[HttpGet]

public ActionResult<List<Producto>> GetProductos()

{

return this.repo.GetProductos();

}

[HttpGet("{id}")]

public ActionResult<Producto> FindProducto(int id)

{

return this.repo.FindProducto(id);

}

[HttpPost]

public async Task<ActionResult> AddProducto(Producto producto)

{

await this.repo.AddProducto(producto.Nombre, producto.Descripcion

, producto.Precio, producto.Imagen);

return Ok();

}

[HttpDelete("{id}")]

public async Task<ActionResult> DeleteProducto(int id)

{

await this.repo.DeleteProducto(id);

return Ok();

}

[HttpPut]

public async Task<ActionResult> UpdateProducto(Producto producto)

{

await this.repo.UpdateProducto(producto.IdProducto, producto.Nombre

, producto.Descripcion, producto.Precio, producto.Imagen);

return Ok();

}

}

Incluimos nuestra cadena de conexión dentro de **appsettings.json**

**APPSETTINGS.JSON**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"AzureSql": "Data Source=azurecastraininglabs.database.windows.net;Initial Catalog=AZUREDATABASE;Persist Security Info=True;User ID=adminsql;Password=Admin123"

}

}

Resolvemos las dependencias dentro de **Program.cs**

**PROGRAM.CS**

using ApiProductosLabs.Data;

using ApiProductosLabs.Repositories;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddTransient<RepositoryProductos>();

string connectionString = builder.Configuration.GetConnectionString("AzureSql");

builder.Services.AddDbContext<ProductosContext>

(options => options.UseSqlServer(connectionString));

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

}

app.UseSwagger();

app.UseSwaggerUI(

options =>

{

options.SwaggerEndpoint("/swagger/v1/swagger.json", "Api Microsoft Labs v1");

options.RoutePrefix = "";

});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();