Resumen del laboratorio Azure

Crear una aplicación Mvc Net Core (Model-View-Controller)

para la creación de usuarios.

Usuario: adminsql

Password: Admin123

1) Azure SQL Server, poder conectarnos desde nuestro equipo y crear una Tabla para poder trabajar.

2) Crear una aplicación Net Core y consumir la base de datos Azure SQL Server.

GetUsuarios(): Mostrar todos los usuarios

FindUsuario(int idUser): Buscar un usuario por su ID

AddUsuario(): Crear un nuevo usuario

3) Crear cuenta Azure Storage y trabajar con Blobs (imágenes)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

4) Implementar la funcionalidad en nuestra App

5) Crear un Azure Logic Apps con un flujo para poder enviar correos desde Azure.

6) Validar los usuarios con un enlace y activarlos en la base de datos.

Recursos necesarios para la sesión:

1) Visual Studio 2022 Enterprise/Community

2) Cuenta de Azure, podéis utilizar perfectamente mis credenciales para el Laboratorio.

3) SQL Server Management Studio

Tabla

Descripción generada automáticamente

Abrimos Visual Studio **2022** y creamos una nueva aplicación Mvc Net Core (Model-View-Controller) llamada **MvcMicrosoftLabsPadel**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Agregamos los Nuget para el acceso a la base de datos Azure SQL Server

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Comenzamos creando una nueva clase, dentro de **Models**, llamada **Usuario**

**USUARIO**

[Table("USUARIOSPADEL")]

public class Usuario

{

[Key]

[Column("IDUSUARIO")]

public int IdUsuario { get; set; }

[Column("NOMBRE")]

public string? Nombre { get; set; }

[Column("APELLIDOS")]

public string Apellidos { get; set; }

[Column("IMAGEN")]

public string Imagen { get; set; }

[Column("FECHA\_NACIMIENTO")]

public DateTime FechaNacimiento { get; set; }

[Column("EMAIL")]

public string Email { get; set; }

[Column("ACTIVO")]

public bool Activo { get; set; }

}

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

**USUARIOSCONTEXT**

public class UsuariosContext: DbContext

{

public UsuariosContext(DbContextOptions<UsuariosContext> options)

: base(options) { }

public DbSet<Usuario> Usuarios { get; set; }

}

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

**REPOSITORYUSUARIOS**

public class RepositoryUsuarios

{

private UsuariosContext context;

public RepositoryUsuarios(UsuariosContext context)

{

this.context = context;

}

//METODO PARA RECUPERAR TODOS LOS USUARIOS

public List<Usuario> GetUsuarios()

{

return this.context.Usuarios.ToList();

}

//METODO PARA BUSCAR UN USUARIO PARA LOS DETALLES

public Usuario FindUsuario(int idUser)

{

return this.context.Usuarios.FirstOrDefault(x => x.IdUsuario == idUser);

}

//METODO PRIVADO PARA RECUPERAR EL MAXIMO ID

//DEL USUARIO AL INSERTAR

private int GetMaxIdUsuario()

{

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

{

return 1;

}

else

{

return this.context.Usuarios.Max(x => x.IdUsuario) + 1;

}

}

//METODO PARA CREAR NUEVOS USUARIOS

//Y DEVOLVERA EL ID DEL NUEVO USUARIO INSERTADO

//PARA FUTURAS ACCIONES

public async Task<int> AddUser(string nombre, string apellidos

, string email, DateTime fechaNacimiento)

{

Usuario user = new Usuario();

user.IdUsuario = this.GetMaxIdUsuario();

user.Nombre = nombre;

user.Apellidos = apellidos;

user.Email = email;

user.FechaNacimiento = fechaNacimiento;

user.Activo = false;

await this.context.Usuarios.AddAsync(user);

await this.context.SaveChangesAsync();

return user.IdUsuario;

}

}

Sobre **Controllers**, agregamos un nuevo controlador llamado **UsuariosController** de tipo Mvc Empty

**USUARIOSCONTROLLER**

public class UsuariosController : Controller

{

private RepositoryUsuarios repo;

public UsuariosController(RepositoryUsuarios repo)

{

this.repo = repo;

}

public IActionResult Index()

{

List<Usuario> usuarios = this.repo.GetUsuarios();

return View(usuarios);

}

public IActionResult Details(int iduser)

{

Usuario user = this.repo.FindUsuario(iduser);

return View(user);

}

public IActionResult NewUser()

{

return View();

}

[HttpPost]

public async Task<IActionResult> NewUser(Usuario usuario)

{

await this.repo.AddUserAsync(usuario.Nombre

, usuario.Apellidos, usuario.Email, usuario.FechaNacimiento);

return RedirectToAction("Index");

}

}

Recuperamos la cadena de conexión Azure del Server Explorer

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Dentro del archivo **appsettings.json** incluimos la cadena de conexión

**APPSETTINGS.JSON**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

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

}

}

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

**PROGRAM.CS**

using Microsoft.EntityFrameworkCore;

using MvcMicrosoftLabsPadel.Data;

using MvcMicrosoftLabsPadel.Repositories;

var builder = WebApplication.CreateBuilder(args);

//AÑADIMOS EL REPOSITORIO

builder.Services.AddTransient<RepositoryUsuarios>();

//RECUPERAMOS LA CADENA DE CONEXION

string connectionString =

builder.Configuration.GetConnectionString("AzureSqlServer");

//INYECTAMOS EL DBCONTEXT DENTRO DE LOS SERVICIOS

builder.Services.AddDbContext<UsuariosContext>

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

// Add services to the container.

builder.Services.AddControllersWithViews();

//LA INYECCION ANTES DE ESTA LINEA

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

Creamos las vistas utilizando **Scaffolding**

Voy a subir el proyecto a **GitHub** y podéis visualizar en casa si lo deseáis.

Mi usuario es **serraguti**

**IMPLEMENTACION FUNCIONALIDAD CON AZURE STORAGE**

Vamos a crear una cuenta de almacenamiento y utilizaremos un **Container** para subir **Blobs** (imágenes) de los usuarios.

Para cada imagen, incluiremos un nombre único, que será el ID del Usuario dado de alta y el nombre de la imagen.

Modificamos el método **AddUser** del **Repositorio** incluyendo dicho cambio.

**REPOSITORYUSUARIOS**

public async Task<int> AddUserAsync(string nombre

, string apellidos

, string email

, string imagen,

DateTime fechaNacimiento)

{

Usuario user = new Usuario();

user.IdUsuario = this.GetMaxIdUsuario();

user.Nombre = nombre;

user.Apellidos = apellidos;

user.Email = email;

user.Imagen = user.IdUsuario + "\_" + imagen;

user.FechaNacimiento = fechaNacimiento;

user.Activo = false;

await this.context.Usuarios.AddAsync(user);

await this.context.SaveChangesAsync();

return user.IdUsuario;

}

**DESCANSO DE 10 MINUTOS…**