Gestión Reservas Restaurante

Portada

Título: Implementación de un Sistema de Reservas para Restaurantes con .NET y Entity

---

Índice

---

1. Introducción

Esta guía describe el desarrollo de un sistema de reservas para restaurantes utilizando .NET 6, Entity Framework Core y un cliente en consola para la interacción con la API. El objetivo es facilitar la gestión de reservas mediante una API REST robusta y un cliente sencillo para su consumo.

---

2. Arquitectura del Sistema

El sistema está compuesto por:

- **Backend:** API REST construida con ASP.NET Core 6.
- **Base de Datos:** In-memory para pruebas, con posibilidad de migrar a SQL Server.
- **Cliente:** Aplicación en consola para la gestión de reservas.
- **Patrón de Diseño:** Arquitectura en capas con separación de responsabilidades.

---

3. Configuración del Entorno de Desarrollo

Requisitos:

- .NET SDK 6+
- Visual Studio / VS Code
- Postman (opcional, para pruebas API)
- SQL Server (opcional, para persistencia)

Instalación:

1. Clonar el repositorio del proyecto.
2. Configurar Entity Framework Core con `dotnet ef`.
3. Ejecutar el servidor con `dotnet run`.

---

4. Implementación del Backend

Configuración de Entity Framework Core

Se utiliza un contexto `RestaurantContext` para gestionar las reservas.

```
Unset
public class RestaurantContext : DbContext
{
    public DbSet<Reservation> Reservations { get; set; }
    public
RestaurantContext(DbContextOptions<RestaurantContext> options)
: base(options) { }
}
```

Creación de Modelos y DbContext

Modelo `Reservation`:

```
Unset
public class Reservation
{
    public long Id { get; set; }
    public string CustomerName { get; set; }
    public DateTime ReservationDateTime { get; set; }
    public int NumberOfGuests { get; set; }
    public int TableNumber { get; set; }
    public bool IsConfirmed { get; set; }
}
```

Implementación de Controladores

Se utiliza un controlador `ReservationsController` para gestionar las reservas.

```
Unset
[Route("api/[controller]")]
[ApiController]
public class ReservationsController : ControllerBase
{
    private readonly RestaurantContext _context;
    public ReservationsController(RestaurantContext context)
    {
        _context = context;
    }

    [HttpGet]
    public async Task<ActionResult<IEnumerable<Reservation>>>
GetReservations()
    {
        return await _context.Reservations.ToListAsync();
    }
}
```

Configuración de Swagger y CORS

```
Unset
builder.Services.AddSwaggerGen();
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAll", builder =>
    {

builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();
    });
});
```

---

5. Implementación del Cliente

El cliente en consola permite gestionar las reservas interactuando con la API REST mediante `HttpClient`.

Ejemplo de consumo de API:

```
Unset
private async Task ViewAllReservationsAsync()
{
    var reservations = await
_httpClient.GetFromJsonAsync<List<Reservation>>(API_BASE_URL);
    foreach (var res in reservations)
    {
        Console.WriteLine($"ID: {res.Id} | Cliente:
{res.CustomerName} | Mesa: {res.TableNumber}");
    }
}
```

El cliente en consola permite la interacción con la API mediante solicitudes HTTP.

Funciones principales:

- Visualizar todas las reservaciones.
- Consultar mesas disponibles.
- Realizar y cancelar reservaciones.
- Ver detalles de las reservas

Uso de System, HttpClient y JsonElement

El cliente de consola hace uso de diversas bibliotecas del espacio de nombres System, en particular:

- **HttpClient**: Utilizado para realizar solicitudes HTTP a la API.
- **JsonElement**: Usado en la clase ReservationNotification para manejar datos en formato JSON sin necesidad de deserializarlos a un tipo específico.
- **System.Threading.Tasks**: Implementa operaciones asíncronas con async/await para mejorar la eficiencia del cliente.

---

6. Pruebas y Validaciones

- Se realizaron pruebas unitarias con `xUnit` se usó `Postman` para validar los endpoints y se verificó la persistencia de datos en memoria.

Pruebas con Postman

Para interactuar con la API, se recomienda usar Postman con las siguientes URLs:

- **Ver todas las reservas:** `GET http://localhost:5021/api/reservations`
- **Ver una reserva específica:** `GET http://localhost:5021/api/reservations/{id}`
- **Ver mesas disponibles:** `GET http://localhost:5021/api/tables/available`
- **Crear reserva:** `POST http://localhost:5021/api/reservations`
- **Actualizar reserva:** `PUT http://localhost:5021/api/reservations/{id}`
- **Cancelar reserva:** `DELETE http://localhost:5021/api/reservations/{id}`

---

8. Conclusiones

Este proyecto demuestra cómo construir una API REST con ASP.NET Core y un cliente en consola para la gestión de reservas. Se destaca:

- Uso de Entity Framework Core.
- Configuración de CORS y Swagger.
- Implementación de un cliente en consola.

---

9. Deudas Técnicas

Aspectos que podrían mejorarse en futuras versiones:

- Implementación de autenticación y autorización.
- Migración a una base de datos persistente como SQL Server.
- Creación de una interfaz gráfica en lugar del cliente de consola.
- Implementación de notificaciones en tiempo real con SignalR.

---

10. Github

- Este es el repositorio en el que se incluye el video de la demo y el proyecto:

  https://github.com/martinPenalva/TRABAJO_PSP.git
