# Instituto Politécnico Nacional
# Escuela Superior De Cómputo

Licenciatura de Ciencia de Datos

Desarrollo de Aplicaciones Web

Documentación de Back-End

# Aquality

**Grupo:**

4AM1

**Integrantes:**

- Ambriz Rangel Alexia M
- Mosco Salgado Cristian
- Reyes Cruz Jorge Jurgen
- Rodriguez Nuñez Diego Eduardo

**Documentación de Back-End**

El backend de la aplicación, implementado en C# utilizando el framework ASP.NET Core, se encarga de gestionar la lógica del negocio y proporcionar servicios a través de una API RESTful. Este backend interactúa con una base de datos MySQL utilizando Entity Framework Core para administrar la persistencia y recuperación de datos relacionados con carritos de compra, especies, historiales, órdenes, productos, tiendas y usuarios, utiliza Entity Framework Core para configurar la base de datos y definir las relaciones entre las entidades del dominio, ofrece una API RESTful que permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los diferentes recursos de la aplicación, gestiona la autenticación y autorización de usuarios para acceder a recursos protegidos, garantizando un adecuado control de acceso, proporciona un punto de conexión para que el frontend interactúe con la lógica del negocio y obtenga o envíe datos según sea necesario y configura rutas específicas para cada controlador, permitiendo una navegación clara y estructurada dentro de la aplicación.

- **Solución "Aquality_api.sln**
  - **ApiGateway**
    - Ocelot.json

Este archivo se utiliza para configurar cómo las solicitudes del cliente al API Gateway deben ser redirigidas a los diferentes servicios downstream basados en la ruta y otros criterios definidos.

- ✓ **Routes:** Define las rutas y configuraciones para redireccionar las solicitudes del API Gateway a diferentes controladores (downstream services).
  - **DownstreamPathTemplate:** La ruta en el servicio downstream (API destino).
  - **DownstreamScheme:** El protocolo utilizado para el servicio downstream (generalmente "https").
  - **DownstreamHostAndPorts:** La dirección y el puerto del servicio downstream.
  - **UpstreamPathTemplate:** La ruta en el servicio upstream (API gateway).
  - **UpstreamHttpMethod:** Los métodos HTTP permitidos para la ruta upstream.
- ✓ **GlobalConfiguration:** Configuración global del API Gateway.
  - **BaseUrl:** La URL base para el API Gateway.

- Program.cs

Este código está configurado para construir y ejecutar una aplicación ASP.NET Core con el middleware Ocelot para la creación de una puerta de enlace de API. A continuación, se detalla la funcionalidad de cada sección:

- ✓ **Agregación de Servicios:** Se añaden los servicios necesarios para controladores, enrutamiento, generación de documentación Swagger y configuración de Ocelot.
- ✓ **Configuración de Ocelot:** Se agrega el archivo de configuración "Ocelot.json" y se inyecta el middleware de Ocelot.
- ✓ **Configuración para Desarrollo:** En el entorno de desarrollo, se habilita la interfaz Swagger para facilitar la documentación de la API.
- ✓ **Uso de Ocelot Middleware:** El middleware de Ocelot se utiliza para gestionar la puerta de enlace de la API y enrutar las peticiones a los servicios correspondientes.
- ✓ **Configuración Adicional:** Se incluye redirección HTTPS, configuración de autorización y mapeo de controladores.
- ✓ **Ejecución de la Aplicación:** La aplicación se construye y ejecuta.

- Aquality_api
  - AqualityContext.cs x

Este archivo define la clase **AqualityContext**, que hereda de **DbContext** de Entity Framework Core. Representa el contexto de la base de datos para la aplicación Aquality y contiene conjuntos de entidades para cada modelo de datos (carrito, especie, historial, orden, producto, tienda y usuario).

- **using Aquality_api.Model;**: Importa el espacio de nombres que contiene las clases de modelo utilizadas en la aplicación.
- **using Microsoft.EntityFrameworkCore;**: Importa las clases necesarias de Entity Framework Core para trabajar con bases de datos.
- **using MySql.EntityFrameworkCore;**: Importa el proveedor MySQL para Entity Framework Core.
- **using System.Reflection.Emit;**: Importa las clases necesarias para generar código en tiempo de ejecución (no utilizadas directamente en este código).
- **namespace Aquality_api.Context**: Define el espacio de nombres y comienza la declaración de la clase **AqualityContext**.
- **public class AqualityContext : DbContext**: Declara la clase **AqualityContext** que hereda de **DbContext**.
- **public DbSet<EntidadModel> NombreDeLaEntidad { get; set; }**: Define un conjunto de entidades para cada modelo en la aplicación.
- **protected override void OnConfiguring(DbContextOptionsBuilder builder)**: Método que se llama al configurar el contexto de la base de datos. Aquí se establecen las opciones de conexión.

- **protected override void OnModelCreating(ModelBuilder builder)**: Método que se llama al crear el modelo de datos. Aquí se configuran las propiedades, claves primarias y relaciones de las entidades en el modelo.

  - CarritoController.cs
  - HistorialController.cs
  - OrdenController.cs
  - ProductoController.cs
  - TiendaController.cs
  - EspecieController.cs
  - UsuarioController.cs

En resumen, estos controladores proporcionan una interfaz para realizar operaciones CRUD en las entidades **CarritoModel, HistorialModel, OrdenModel, ProductoModel, TiendaModel, EspecieModel y UsuarioModel** a través de una API RESTful.

- ✓ **using Directivas:**
  - **Microsoft.AspNetCore.Mvc**: Proporciona funcionalidades para construir aplicaciones web y APIs usando el patrón MVC.
  - **Microsoft.EntityFrameworkCore**: Proporciona acceso a bases de datos utilizando Entity Framework Core.
  - **Aquality_api.Context**: Namespace que contiene la definición del contexto de la base de datos **AqualityContext**.
  - **Aquality_api.Model**: Namespace que contiene las definiciones de los modelos de datos.
  - **Microsoft.EntityFrameworkCore.Scaffolding.Metadata**: Contiene clases que representan la información de metadatos de esquema de base de datos.
- ✓ **Clase XController:**
  - Decorada con **[ApiController]**: Indica que esta clase es un controlador de API.
  - Decorada con **[Route("XController")]**: Especifica la ruta base para las rutas de este controlador.
- ✓ **Método XGet:**
  - Decorado con **[HttpGet]**: Indica que este método maneja solicitudes HTTP GET.
  - Obtiene la lista de carritos desde la base de datos y la devuelve en formato JSON.
- ✓ **Método XPost:**

- Decorado con **[HttpPost]**: Indica que este método maneja solicitudes HTTP POST.
- Agrega un nuevo carrito a la base de datos y devuelve una respuesta JSON indicando el resultado.

✓ **Método XUpdate:**

- Decorado con **[HttpPatch]**: Indica que este método maneja solicitudes HTTP PATCH.
- Actualiza un carrito existente en la base de datos y devuelve una respuesta JSON indicando el resultado.

✓ **Método XDelete:**

- Decorado con **[HttpDelete]**: Indica que este método maneja solicitudes HTTP DELETE.
- Elimina un carrito existente de la base de datos y devuelve una respuesta JSON indicando el resultado.

- CarritoModel.cs
- HistorialModel.cs
- OrdenModel.cs
- ProductoModel.cs
- TiendaModel.cs
- EspecieModel.cs
- UsuarioModel.cs

Estos archivos pertenecen al espacio de nombres **Aquality_api.Model** y definen la clases **XModel**. Estas clases sirven como un modelo de datos para representar la información de un usuario, especie, tienda, producto, historial y carrito en el contexto de una aplicación Aquality.

- Program.cs

Este archivo representa la configuración y la inicialización de una aplicación web API en ASP.NET Core. A continuación, se proporciona un resumen de las principales secciones y funciones del código:

✓ **Creación de la Aplicación:** Se utiliza **WebApplication.CreateBuilder(args)** para crear un constructor de la aplicación web.

✓ **Configuración de Servicios:** Se agregan servicios al contenedor de inyección de dependencias, como controladores y configuración de Swagger/OpenAPI para documentación.

- ✓ **Construcción de la Aplicación:** Se construye la aplicación a partir del constructor.
- ✓ **Configuración del Pipeline de Solicitudes HTTP:** Se configuran las etapas del pipeline de solicitudes HTTP, incluida la habilitación de Swagger en entornos de desarrollo y la configuración de CORS.
- ✓ **Aseguramiento de la Base de Datos:** Se asegura de que la base de datos esté creada mediante **database.Database.EnsureCreated()**.
- ✓ **Redirección HTTPS y Configuración de Autorización:** Se establece la redirección HTTPS y se configura la autorización.
- ✓ **Mapeo de Controladores:** Se mapean los controladores de la API.
- ✓ **Ejecución de la Aplicación:** La aplicación se ejecuta con **app.Run()**.