DOCUMENTACION SAKILA CRUD ORM API

Michael Rafael Ventura Bautista

Fernielis Vicioso Rodrguez

Julian Alejandro Núñez Toribio

Universidad Autonoma de Santo Domingo

Ciencia de Datos 1 | INF-7303-C1

Dr. Silverio del Orbe

17 de mayo de 2025

Resumen

Esta solucion de API responde a un esquema de CRUD desarrollado con ASP.NET Core, utilizando Entity Framework Core como ORM y MySQL como sistema de gestión de base de datos. La API permite la gestión de datos relacionados con Categorias, Direccion de Clientes, Rentas de categorías, Pais y sus ciudades, y está diseñada siguiendo buenas prácticas de arquitectura limpia y principios de diseño REST.

El objetivo principal de esta API es proporcionar una interfaz estándar y eficiente para interactuar con los datos almacenados en MySQL desde aplicaciones cliente, como aplicaciones web, móviles o servicios externos que para este escenario fuese conveniente para el tema de análisis de datos con algún cliente determinado.

Se pretende con esta solución Exponer una interfaz HTTP para operaciones CRUD, Entity Framework Core para el mapeo entre objetos y tablas, Aplicar principios de diseño RESTful y separación de responsabilidades, Facilitar la extensión y el mantenimiento del proyecto.

El proyecto sigue una arquitectura de capas común en aplicaciones .NET de Controladores que reciben las solicitudes HTTP y delegan la lógica, Servicios o Repositorios que contienen la lógica de negocio y de acceso a datos, Modelos que representan las estructuras de datos que se mapean a las tablas de MySQL y todo esto operado por el DbContext que es la clase central que administra las conexiones.

Palabras clave: API REST, Entity Framework Core, ORM, MySQL, CRUD, DbContext, Controlador, Repositorio, JSON, HTTP Methods

Contenido

[Resumen 2](#_Toc200148796)

[DOCUMENTACION SAKILA CRUD ORM API 4](#_Toc200148797)

[API REST 4](#_Toc200148798)

[Entity Framework Core 4](#_Toc200148799)

[Base de Datos MySQL 5](#_Toc200148800)

[Modelo Final de Endpoints 5](#_Toc200148801)

[Esquema de Ficheros 6](#_Toc200148802)

[Referencias 7](#_Toc200148803)

DOCUMENTACION SAKILA CRUD ORM API

# API REST

Transferencia de estado representacionl (Representational State Transfer) es una interfaz de comunicación que permite a diferentes aplicaciones interactuar entre sí a través del protocolo HTTP siguiendo principios arquitectónicos. Cada recurso (por ejemplo, un usuario o un producto) se representa mediante una URL y puede ser manipulado usando métodos HTTP estándar:

• GET – Obtener información

• POST – Crear nuevos recursos

• PUT / PATCH – Actualizar recursos existentes

• DELETE – Eliminar recursos

Las API REST son ampliamente utilizadas por su simplicidad, escalabilidad y compatibilidad con múltiples lenguajes y plataformas.

# Entity Framework Core

Es un ORM (Object-Relational Mapper) de código abierto desarrollado por Microsoft. Permite interactuar con bases de datos relacionales utilizando objetos de C#, eliminando la necesidad de escribir SQL directamente en la mayoría de los casos.

EF Core facilita:

• El mapeo entre clases de C# y tablas de la base de datos.

• La generación automática de migraciones y esquemas de base de datos.

• La ejecución de consultas LINQ (Language Integrated Query) para acceder y modificar datos.

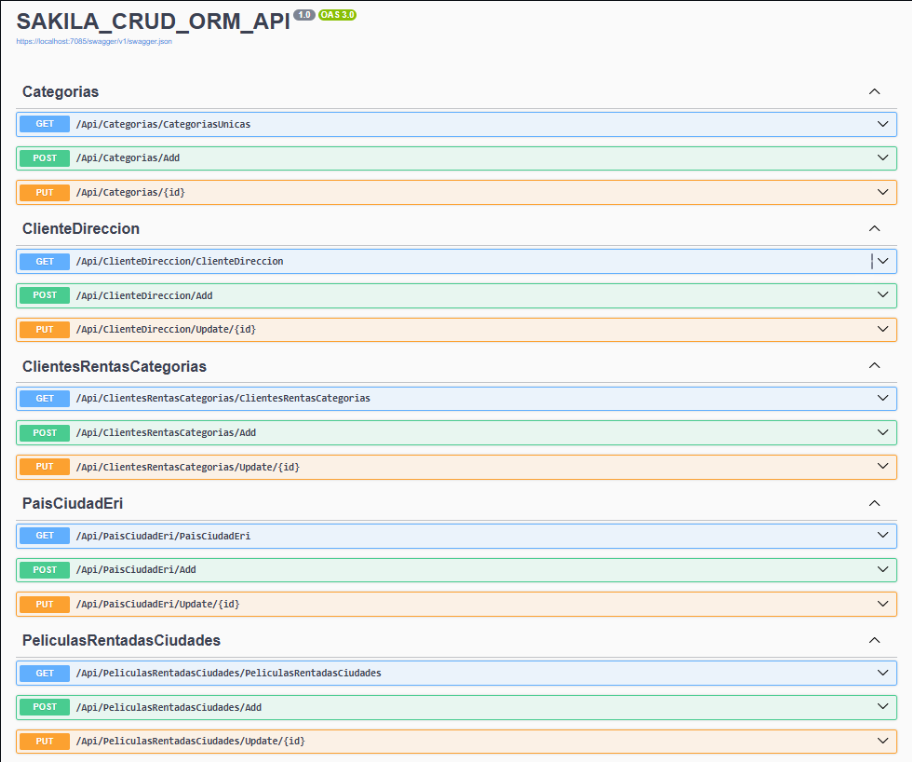
# Base de Datos MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) ampliamente utilizado en aplicaciones web y empresariales por su rendimiento, fiabilidad y compatibilidad con múltiples entornos.

La integración de EF Core con MySQL permite que las clases del dominio se sincronicen con la base de datos mediante proveedores específicos como Pomelo.EntityFrameworkCore.MySql.

# Modelo Final de Endpoints

Estos serían las consignias finales después de la configuración final y apertura en servidor



# Esquema de Ficheros

Considerando que la solución cuenta con un esquema de API REST, se hizo una abstracción utilizando el concepto de los principios SOLID respondiendo al primero de responsabilidad única, determinando en que carpeta va la segmentación determinada para de esta manera cada modelo cuente con su propio dominio, el cual debe ser consumido por el servicio que responda a este modelo y ser canalizado vía su controlador. Además, los archivos comunes respondiendo a un solo bloque de credenciales el cual es consumido vía mecanismos de manejo de string, y esto al estar en una composición de un patrón estructural MVC se ve manejado via un archivo de orquestación



**Enlace a Repositorio:** https://github.com/michaelventura01/SAKILA\_CRUD\_ORM\_API/tree/master

# Referencias

Fielding, R. (s.f.). *https://www.ics.uci.edu*. Obtenido de Roy Fielding’s Dissertation (2000): https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\_arch\_style.htm

Martin, R. C. (s.f.). *Clean Code.*

Microsoft Developer Network. (s.f.). *https://learn.microsoft.com*. Obtenido de Concepto general de SOLID: https://learn.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles

Microsoft Docs. (s.f.). *https://learn.microsoft.com*. Obtenido de Microsoft Docs – ASP.NET Core MVC: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0

Microsoft Learn. (s.f.). *https://learn.microsoft.com*. Obtenido de Arquitectura de software en .NET: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures