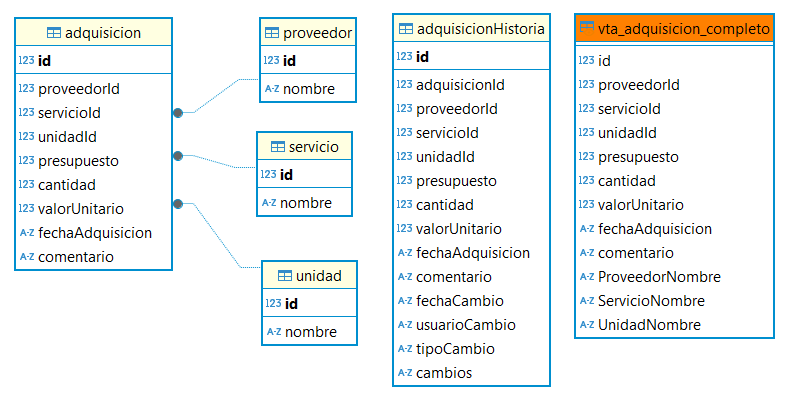
El desarrollo de la prueba técnica se realizó en 3 partes:

**Base de datos** : se escogió SQLite ya que es una base de datos relacional que se integra perfectamente a pequeños desarrollos y no requiere la instalación de aplicativos externos

La base de datos para este ejemplo tiene 5 tablas y una vista



**Tablas**

**Adquisicion**: es la tabla principal en esa se guarda la información básica de la adquisición

**Proveedor**: Contiene la información de las Entidades proveedoras

**Servicio**: Contiene la información de los Bienes o Servicios

**Unidad**: Contiene la información de las unidades administrativa responsables

**AdquisicionHistoria:** Almacena la bitácora de las acciones efectuadas (Adición, Edición, Eliminación) guarda la información del registro antes de la acción, la información de quien cambio el registro, la fecha, el tipo de acción y una descripción de los cambios realizados



**Vista**

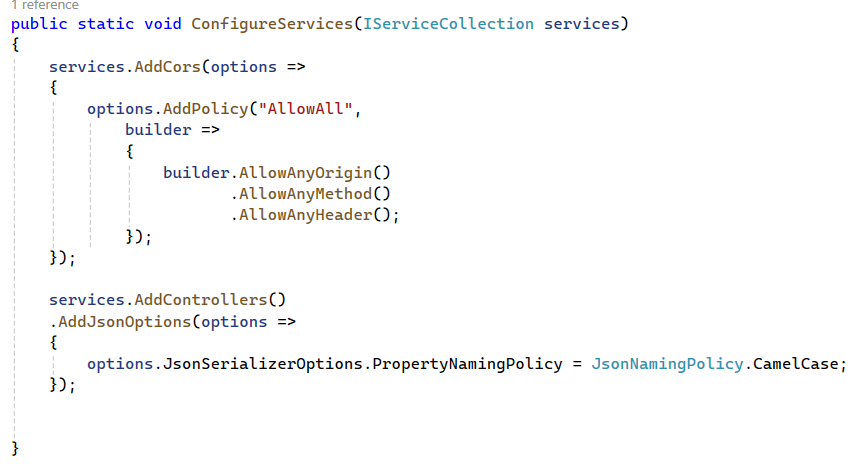
**vta\_adquisicion\_completo** : contiene la información de la adquisición y los nombres de las tablas foráneas, solo para proceso de consulta

**App Cliente**: Como se solicito para el aplicativo cliente se utilizo HTM, CSS JavaScript Vanilla y JQuery y toastr.js para el manejo de ventanas modales y mensajes al cliente

|  |  |
| --- | --- |
|  | Se utilizó Visual Studio Code como IDE, se utilizó una plantilla básica para activar un servidor con Express (Node) (facilidad para el desarrollo) pero para su despliegue se realizaría copiando la carpeta public en el servidor  Si se quiere utilizar el servidor Express se los pasos a ejecutar son los siguientes:   1. Abrir una terminal en la carpeta del proyecto. 2. Instalar dependencias:   Ejecutar **npm install**   1. Inicia el servidor:   Ejecutar npm start   1. Abre tu navegador en:   http://localhost:3000 |
| Dentro de la carpeta js se encuentra un archivo llamado adquisicion.js  En su cabecera se encuentra la definición de el path base de los servicios    Por defecto visual estudio utiliza el puerto 5107 para hacer despliegue de los desarrollos web, si al momento de desplegar los WebApi y estos quedan en otra ruta se debería actualizar esta constante | |
|  |  |

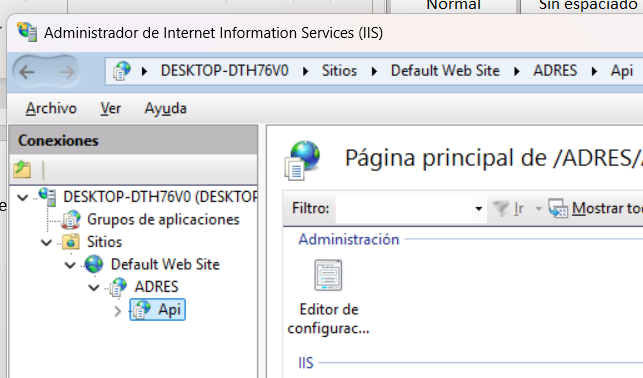
**WebApi:** El desarrollo del API se realizó en .NET ( C# ) dependiente del FrameWork .NET 6.0

|  |  |
| --- | --- |
|  | Se desarrollo utilizando Visual Studio la solución consta de 3 proyectos :  **AdresLib (Class Library):** Contiene todo el código necesario para acceder al repositorio (SQLite), las validaciones, al ser una librería independiente se puede utilizar para el despliegue en entornos Web, Forms o Consola. Tiene dos dependencias una a Dapper y otra a Microsoft.Data.Sqlite (proveedor para la base de datos SQLite)  **AdresWeb (WebApi):** Sirve como EndPoint Web para los servicios que se encuentran en **AdresLib** , tiene dos controladores :  AdquisicionController : controlador para los métodos relacionados con la entidad Adquisición (adicionar, eliminar, modificar, recuperar)  UtilController : Controlador para listar entidades diferentes a la Adquisición tales como Proveedores, Servicios, unidades.  **AdresConsole (Consola):** Sirve como pruebas de desarrollo rápido de los métodos desarrollados en **AdresLib** |

Para evitar problemas de CORS se adiciono una directiva en la WEB Api 

Esto debería desactivarse en producción.

También se podría limitar a localhost y/o si en IIS (internet Information server) se publica el apli como un subdominio y el cliente se publica en el Dominio



Para este ultimo caso tocaría cambiar la constante del URL de las apis



a “/api/” pero esto solo aplica si se publica en IIS

para validar que el API está ejecutándose correctamente se puede visitar la url que crea Swagger

