Prueba Técnica Aranda Software
Desarrollador FullStack

Diego Alejandro Pineda Silva
CC: 1022426254
Fecha: 02/02/2022

CONTEXTO

Actualmente existe un catálogo de productos que cuenta con las siguientes características:

- Nombre
- Descripción breve
- Categoría
- Imagen del producto

Desarrolle una API Restful que permita:

1. Crear un nuevo producto
2. Modificar un producto existente
3. Borrar un producto existente
4. Listar los productos existentes teniendo en cuenta que se puede buscar por:
    a. Nombre
    b. Descripción
    c. Categoría

    Es necesario que se puedan ordenar los resultados de forma ascendente o descendente ya sea por nombre o por categoría.

    Tenga en cuenta que el catálogo de productos puede crecer, hasta llegar a tener más de 100 mil registros.

Tener en cuenta los siguientes requerimientos técnicos:

- Lenguaje C# (net framework 4.7+ / net6.0)
- Base de datos SQL server 2014+
- Usar Entity Framework como ORM
- Deben reflejarse los principios RestFul en el API
- Usar una arquitectura claramente definida (3 capas / dominio).
- Debe usarse inyección de dependencias.
- Debe existir evidencia del manejo de errores.
- Entregar el código fuente en un repositorio de GitHub (público) o en su defecto un archivo ZIP.

PROCESO

Para la realización del proyecto se tomaron como punto de partida ciertos pasos importantes de acuerdo con las necesidades del cliente (debido a que los requerimientos ya están especificados, éstos no se tienen en cuenta dentro del análisis); dichos puntos son los siguientes:

1. Análisis de las arquitecturas a implementar y las tecnologías a utilizar.
2. Análisis y diseño de la base de datos.
3. Construcción del servicio de productos (API.
4. Construcción de la interfaz de usuario.

A continuación, se van a describir cada uno de los puntos anteriormente expuestos con el tiempo estimado para cada uno de estos teniendo en cuenta el tiempo total de 48 horas.

1. Análisis de las arquitecturas y tecnologías. (1 Hora)

Para el desarrollo del proyecto se van a utilizar las siguientes arquitecturas y patrones de diseño:

- Arquitectura de 3 capas, para la organización del código.
- Patrón de diseño de Repositorios y Unidad de trabajo.

También se van a implementar las siguientes tecnologías:

- .Net 6.0 y Entity Framework 6 para el servicio de productos (API).
- Microsoft Sql Server 2019 (Base de datos).
- ReactJS (Interfaz de usuario).
- Git/Github (Control de código fuente).

Además, se van a utilizar las siguientes herramientas:

- Visual Studio Code (Desarrollo del código).
- MySQL Workbench (Diseño de la base de datos).
- Postman (Pruebas del servicio o API).
- SQL Server Management Studio (Administración de la base de datos).

2. Análisis y diseño de la base de datos. (1 Hora)

Dentro del desarrollo de una aplicación el análisis y diseño de la base de datos es uno de los puntos más importantes. Para el análisis se debe tener en cuenta los requerimientos solicitados.

Debido a que solo se tiene conocimiento de un catálogo de productos, se va a crear una tabla de productos y adicionalmente una tabla categoría relacionada a la tabla de productos, con el fin de tener categorías fijas en los productos.

A continuación, en la figura 2 se observar la representación el modelo relacional de la base de datos que se va a utilizar en donde se pueden dos tablas, la primera tabla “Categorie” la cual va a contener las categorías y también esta la tabla “Product” en donde se van a almacenar los datos de los productos, dicha tabla tiene la información del nombre del producto, una descripción, una categoría (está relacionado con la tabla “Categorie”),

una imagen (ruta donde se va a guardar la imagen), y un estado el cual indica si el producto está o no disponible.

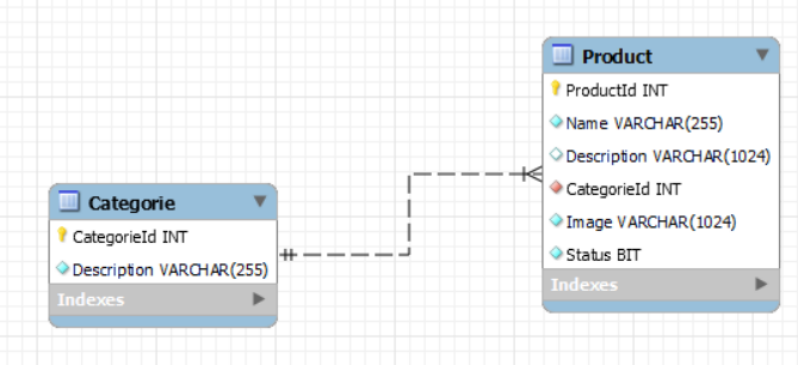

Figura 1. Modelo de la base de datos.

Para la creación y la carga de datos de la base de datos se utilizan dos scripts, *1_createDB.sql* para crear la base de datos y sus correspondientes tablas y llaves, y *2_populateDB.sql* para agregar los datos de las categorías y algunos productos. A continuación, en las figuras 2 y 3 se puede observar la estructura básica de los scripts en donde se valida que cada uno de los objetos no existan antes de ser creados y la carga de los datos.

```
USE [master];
GO

IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'Catalog')
BEGIN
    CREATE DATABASE [Catalog]
END
GO

USE [Catalog];
GO



-----------------------------------------Table dbo.Categorie-----------------------------------------
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO

IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Categorie]') AND type IN (N'U'))
BEGIN
    IF NOT EXISTS (SELECT * FROM [dbo].[Categorie])
    BEGIN
        DROP TABLE [dbo].[Categorie];
    END
END

IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Categorie]') AND type IN (N'U'))
BEGIN
    CREATE TABLE [dbo].[Categorie] (
        [CategorieId] [int] IDENTITY(1, 1) NOT NULL,
        [Description] [varchar](255) NOT NULL,
        CONSTRAINT [PK_Categorie] PRIMARY KEY CLUSTERED
        (
            [CategorieId] ASC
        )
    );
END
GO
```

Figura 2. Script de creación de la base de datos.

```sql
USE [Catalog];
GO

DECLARE @ResetIdentity INT = 0;

----------------------------------------Table dbo.Categorie----------------------------------

DECLARE @Categorie_temp TABLE
(
    [CategorieId] [int] NOT NULL,
    [Description] [varchar] (255) NOT NULL
);

INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (1, 'Vehiculos');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (2, 'Supermercados');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (3, 'Computación');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (4, 'Celulares');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (5, 'Cámaras');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (6, 'Televisores');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (7, 'Electrodomesticos');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (8, 'Deportes');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (9, 'Construcción');
INSERT INTO @Categorie_temp ([CategorieId], [Description]) VALUES (10, 'Moda');

SET IDENTITY_INSERT [dbo].[Categorie] ON;

MERGE INTO [dbo].[Categorie] AS [Des]
USING
    (
        SELECT
            [CategorieId],
            [Description]
        FROM @Categorie_temp
    ) AS [Ori]
ON [Ori].[CategorieId] = [Des].[CategorieId]
WHEN NOT MATCHED BY TARGET
THEN INSERT
    (
        [CategorieId],
        [Description]
    )
    VALUES
    (
        [Ori].[CategorieId],
        [Ori].[Description]
    )
WHEN MATCHED AND TRIM([Des].[Description]) <> TRIM([Ori].[Description])
    THEN UPDATE SET
        [Des].[Description] = TRIM([Ori].[Description]);
```

Figura 3. Script de carga de datos.

Los scripts anteriormente mencionados se encuentran almacenados en el proyecto, en la ruta “src/Data”, además se encuentra un diagrama del modelo relacional de la base de datos.

3. Construcción del servicio de productos

Para la creación del servicio (API Restful) de productos se va a utilizar la versión .Net 6 y además para el mapeo de la base de datos se va a utilizar la versión mas reciente de Entity Framework. El servicio se va a almacenar en la ruta “src/Services/Products”.

La configuración de la aplicación se encuentra en “src/Infraestructure”, en donde se puede observar las entidades, el contexto de la base de datos, los repositorios y la unidad de trabajo como se puede observar a continuación.

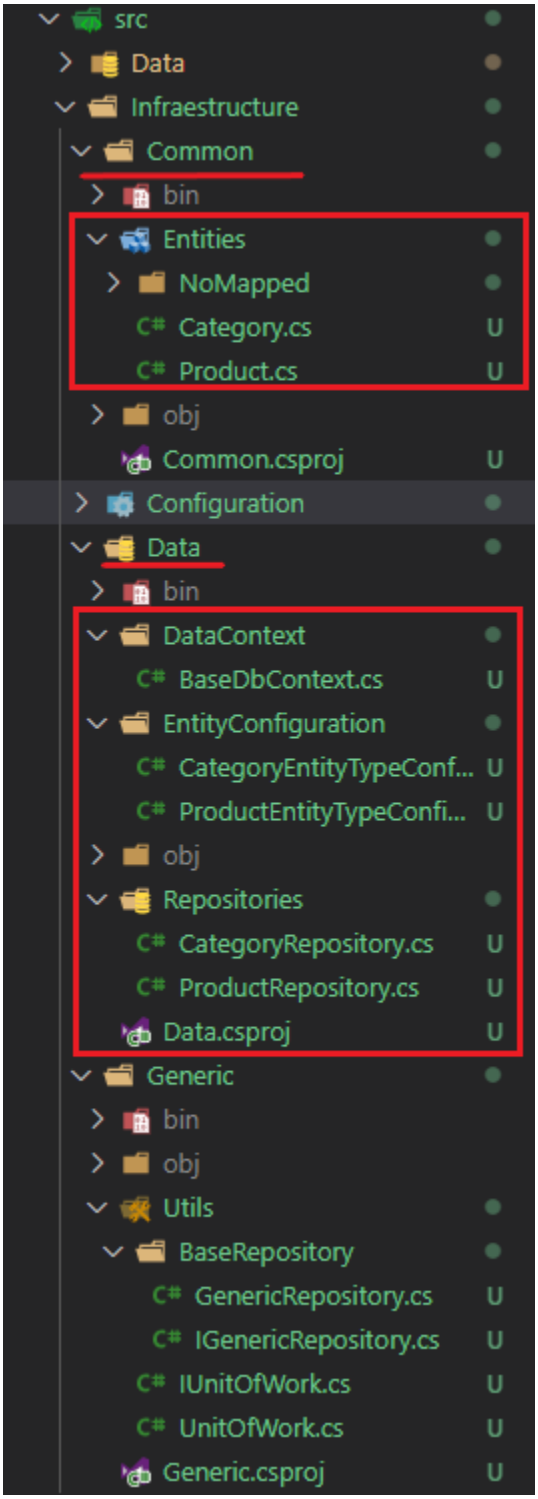

Figura 4. Estructura de las capas.

Además de las capas de la imagen anterior se tiene una capa de Negocio la cual otorga acceso a los métodos expuestos por la API. En la figura 4 se puede observar la estructura de la capa de negocio mencionada anteriormente.

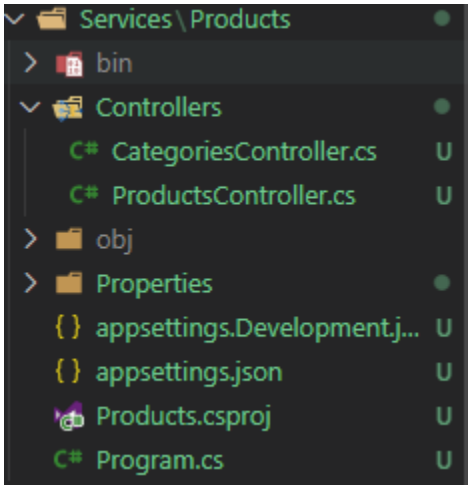

Figura 5. Estructura de la capa de negocio.

Los Endpoints expuesto se pueden ver al correr la aplicación con Swagger como se puede ver en la siguiente imagen.

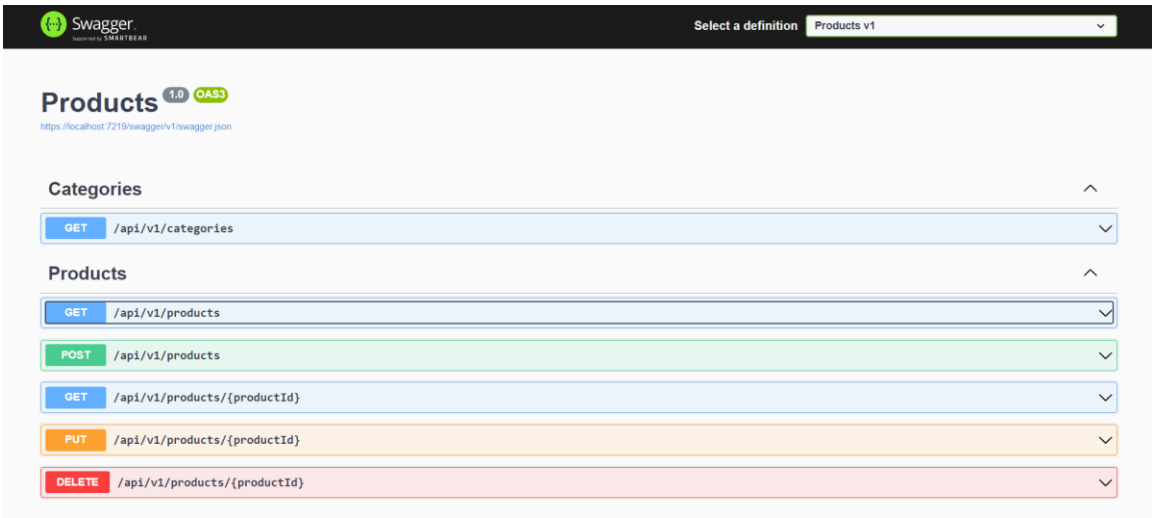

Figura 6. Endpoinst en Swagger.

4. Construcción de la interfaz de usuario

Para la interfaz gráfica se utilizó ReactJS y se dividió en tres grandes ítems, el primero ítem son los componentes, el segundo son los modelos y el tercero es el consumo de los servicios como se muestra a continuación.

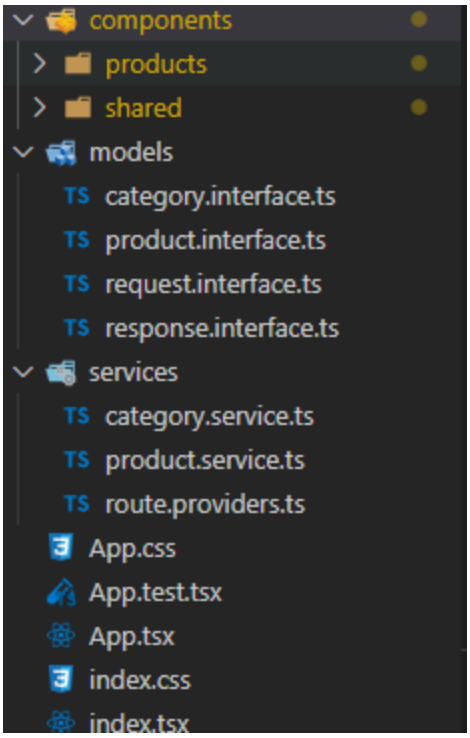

Figura 7. Estructura del FrontEnd.

Al correr la aplicación se puede ver el listado de productos con su correspondiente información y además se pueden observar los filtros que se pueden realizar, así como la paginación.

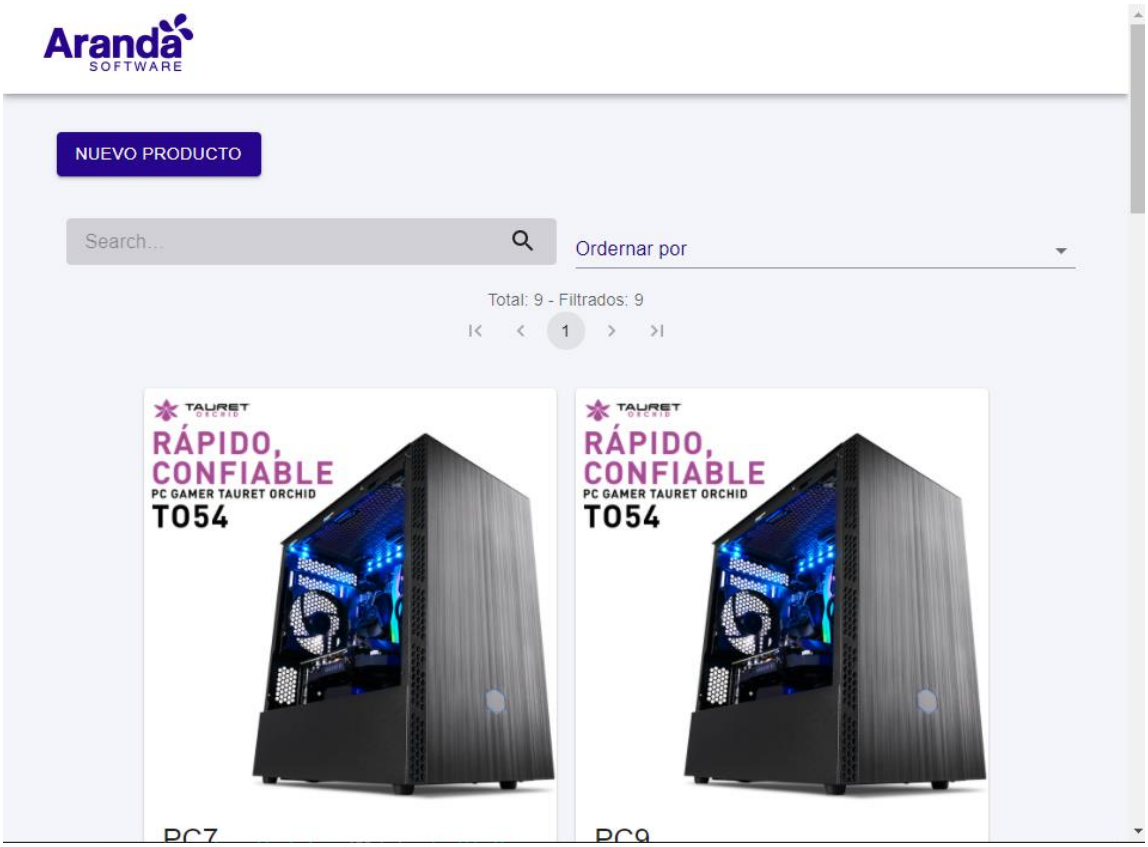

Figura 8. Pantallazo principal

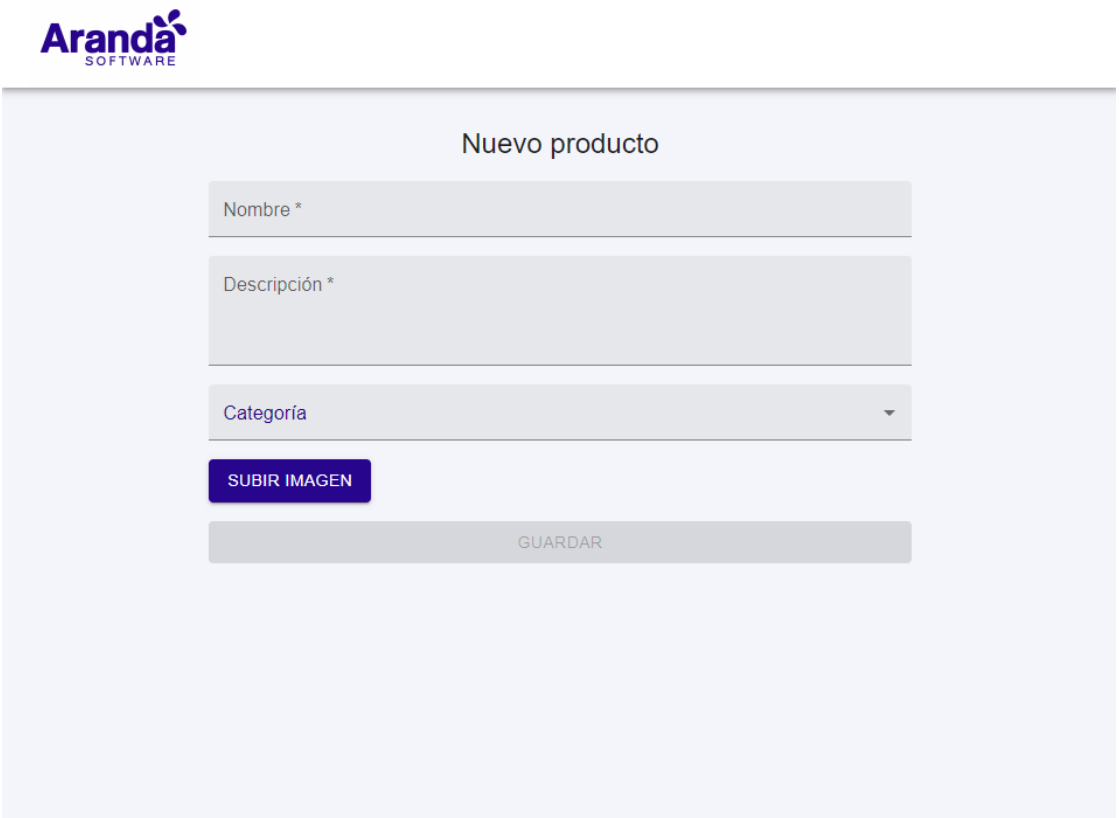

Figura 9. Pantallazo crear producto

La aplicación permite crear, editar, eliminar y filtrar los productos de acuerdo con los parámetros establecidos, además contiene una paginación que permite navegar entre los ítems y permite solo mostrar los productos pertinentes de dicha página, sin tener que obtener todos los productos en una sola petición.