**Arquitectura y patrones.**

La API se realizó utilizando la arquitectura Onion la cual interactúa a través de interfaces, se utiliza el principio de inversión de control, el cual me permite controlar el flujo de ejecución a través del framework o add-on externo.

Gráfico, Diagrama

Descripción generada automáticamente

Domain Model: Contiene las entidades del dominio.

Domain Services: Se definen Interfaces necesarias para almacenar o recuperar los objetos, aunque se implementa en la capa de Infraestructure.

AplicationServices: Se implementan las interfaces core para dar respuesta a las funcionalidades del UI – Infraestructure -Test.

Se utilizo el patron CQRS, el cual permite separar la forma de leer y escribir los datos, y Mediator Pattern, el cual permite reducir dependencias entre componentes y crea un componente mediador para que evitar estas dependencias.

Captura de pantalla de computadora

Descripción generada automáticamente

**Capa de Aplicación (Aplication Services)**

**Texto

Descripción generada automáticamente con confianza baja**

**Behaviours:** se realiza la canalización, captura, validaciones entre todas los Request y Response (pipeline).

**DTOs:** se realiza la declaración de objetos con los que se mapean entidades del dominio, usuarios y token del JWT.

**Enums:** se realiza la declaración de los Roles por defecto de la aplicación.

**Exceptions:** se realiza el manejo de las excepciones y las validaciones de la API.

**Features:** se realiza la definición de las características de la aplicación, como el manejo de las operaciones de las entidades con su mapeo y las diferentes parametrizaciones de validaciones.

**Interfaces:** se realiza la creación de interfaces genéricas para utilizar el patron repositorio, generación de fecha de acciones y manejo de usuarios.

**Mappings:** Se realiza el mapeo entre las entidades y los DTO.

**Parameters:** se realiza la definición de los parámetros genéricos para la paginación, el número de páginas y la cantidad de registros, por defecto 10 por página.

**Specifications:** se realiza la definición de que parámetros se van a utilizar para listar y filtrar todas las entidades.

**Wrappers:** se realiza la definición de la respuesta para la paginación, cuando es succeeded con su respectivo mensaje de forma estándar.

**Capa de Domino (Domain Model and Services)**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Commond:** se realiza la definición de entidades o modelos comunes o que se utilizan para heredar.

**Entities:** se realiza la definición de las entidades de toda la lógica del negocio.

**Settings:** se realiza la definición de la entidad para el uso del JWT.

**Capa de Infraestructura (Infraestructure)**

**Identity:** se maneja todo lo relacionado a la autenticacion y seguridad con JWT

Imagen que contiene Texto

Descripción generada automáticamente

**Context:** se realiza la creación del dbcontext para el manejo de autenticación.

**Helpers:** se realiza la clase para identificar la IP de donde se hacen las peticiones.

**Migrations:** se realiza el manejo de las migraciones para el context Identity.

**Models:** se realiza la definición de la clase para autenticar.

**Seeds:** se realiza la generación usuarios y roles por defecto para utilizar la autenticación.

**Services:** se realiza la autenticación, registro y generación del token JWT.

**Persistence:** se maneja todo lo relacionado a la configuración, definición y parametrización de las entidades y el dbContext para que sean migradas a la base de datos.

Texto

Descripción generada automáticamente

**Configuration:** se realiza la definición de la entidad y las propiedades de cada columna, así como las respectivas relaciones entre ellas.

**Context:** se realiza la creación del dbcontext para el manejo de las entidades.

**Migrations:** se realiza el manejo de las migraciones para el context Application.

**Repository:** se realiza la creación de un repositorio genérico para el dbcontext.

**Shared:** un servicio compartido para manejar la fecha en utc-

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

**Services:** se realiza la generación de la fecha en formato UTC.

**Capa de Presentacion (User Interface)**

Texto

Descripción generada automáticamente

**Controllers:** Se realiza la definición de todos los controladores de las entidades y para la autenticacion, los cuales solo enrutan, también se da la opción de manejar un versionamiento para la API, por defecto es versión 1.

**Extensions:** se realiza la inyección de dependencias para utilizar el middleware y el versionamiento.

**Middlewares:** se realiza la creación de un middleware el cual maneja todos los errores del httpcontext.

**Filtros y paginaciones que se pueden utilizar:**

Entidad Owners:

int pageSize, int pageNumber, string name, string email

Entidad Property:

int pageSize, int pageNumber, string name, string addres, double price, string codInternal, int year

Entidad PropertyImage:

int pageSize, int pageNumber, string file, string idProperty

Entidad ProperyTrace:

int pageSize, int pageNumber, string date, string name, double value, double tax

**Ingreso a la aplicación:**

Después de iniciar la aplicación se debe autenticar, actualmente hay 2 usuarios creados por defecto:

Usuario admin con todos los permisos:

"email": "useradminmail@mail.com",

"password": "1234Pa$$word"

Usuario básico con opciones de lectura:

"email": " userbasicmail@mail.com",

"password": "1234Pa$$word"

**Aplicación

Descripción generada automáticamente con confianza baja**

Responderá con un token si los datos ingresados son correctos.

Texto

Descripción generada automáticamente con confianza baja

Para utilizar cualquier api se debe de agregar el token de la respuesta y además en el campo versión se debe colocar 1, ya que es la versión que se tiene actualmente, si se implementaran otras versiones se colocaría el numero en ese campo.

