
OBLIGATORIO 1 - SmartHome

Diseño 2

Descripción del diseño

Link al repositorio:

<https://github.com/IngSoft-DA2/281542-281835-281535>

Integrantes:

- Matias Corvetto (281535)
- Alejo Fraga (281542)
- Sebastian Vega (281835)

Docentes:

- Daniel Acevedo
- Federico Gonzales

Septiembre 2024

Universidad ORT Uruguay

Facultad de Ingeniería

Abstract

En este documento se explican las decisiones de diseño adoptadas para el desarrollo del primer obligatorio de la asignatura **Diseño de Aplicaciones 2**. Estas decisiones fueron tomadas siguiendo principios de arquitectura de software que buscan asegurar una estructura modular, mantenible y escalable. Además, se incluirán diagramas que complementan las explicaciones, facilitando la comprensión de las ideas expuestas.

La aplicación se estructura en tres proyectos principales: **BusinessLogic**, **DataLayer** y **WebApi**, cada uno con responsabilidades definidas y alineadas con los principios de bajo acoplamiento y alta cohesión. Este documento justifica dichas elecciones arquitectónicas, detallando cómo contribuyen al objetivo general del desarrollo.

Índice

Abstract.....	2
1. Descripción general del trabajo.....	4
1.1 Herramientas utilizadas.....	4
2. Diagrama general de paquetes.....	5
3. Tabla de asignación de responsabilidades (namespaces).....	7
4. Justificación del diseño.....	9
5. Modelo de tablas en la Base de Datos.....	10
6. Diagramas de interacción.....	11
5.1 Creación de un hogar.....	11
5.2 Borrar administrador.....	11
7. Diagrama de componentes.....	12
8. Declaracion de autoria.....	12

1. Descripción general del trabajo

Se desarrolló un sistema para la gestión de dispositivos de seguridad en hogares inteligentes, dependiendo el rol del usuario activo en la aplicación se destacan las siguientes funcionalidades principales incluidas en el alcance de esta versión del proyecto:

- ❖ Mantenimiento de cuentas de administrador
- ❖ Creación de cuentas para dueños de empresas
- ❖ Listar todas las cuentas
- ❖ Listar todas las empresas
- ❖ Manejo de notificaciones para miembros de hogares
- ❖ Creación de una empresa
- ❖ Registrar dispositivos soportados
- ❖ Autenticación mediante credenciales:
- ❖ Listar dispositivos soportados
- ❖ Creación de cuenta para dueño de hogar
- ❖ Listado de dispositivos
- ❖ Creación de hogares
- ❖ Agregar miembros al hogar
- ❖ Asociar dispositivos al hogar
- ❖ Listado de miembros de un hogar
- ❖ Listado de dispositivos en un hogar

1.1 Herramientas utilizadas

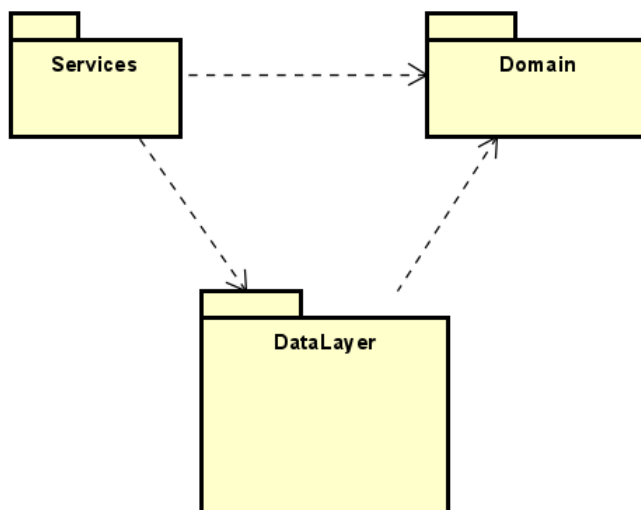
- ❖ C# para el desarrollo
- ❖ MSTest para las pruebas unitarias
- ❖ Visual Studio y Rider como IDE
- ❖ Microsoft SQL Server Express 2017
- ❖ Postman
- ❖ NET Core SDK 8.0/ ASP.NET Core 8.0(C#)
- ❖ Entity Framework Core 8.0
- ❖ Astah UML 2

2. Diagrama general de paquetes

Se tuvo especial consideración para el diseño el cumplimiento de DIP de SOLID.

Inicialmente, nos enfrentamos a la problemática de que nuestra capa lógica (servicios) dependían directamente de la capa de acceso a datos (repositorios) lo cual incumplía el principio mencionado dado que la aplicación no debería de depender de la tecnología (la tecnología es mas volatil que el negocio), es decir, la capa lógica es de mayor nivel que la capa de acceso a datos, por lo cual es un error que una agrupación lógica de alto nivel dependa otro de bajo nivel.

En un principio, nuestro diseño tenía un aspecto similar al ilustrado en la siguiente imagen:

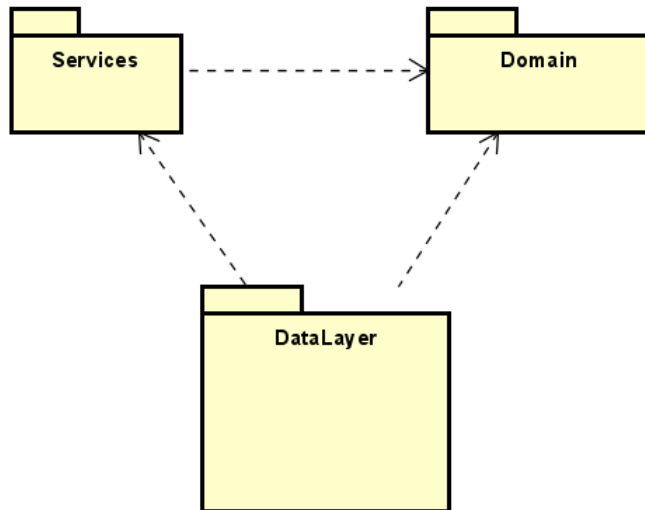


Nos veíamos forzados a separar la lógica de negocio en dos agrupaciones diferentes para evitar una dependencia cíclica (Services y Domain).

Luego de iterar sobre el diseño, logramos la siguiente idea:

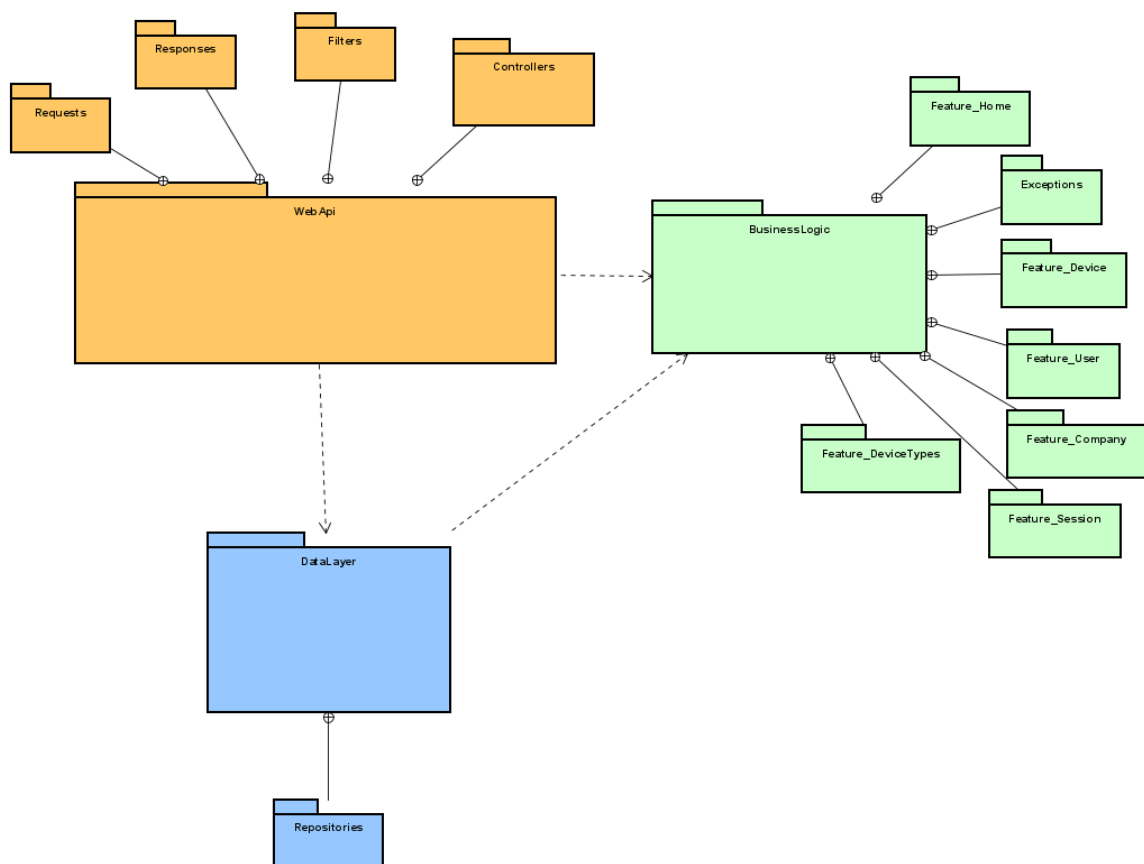
Introducir interfaces del lado de la lógica del negocio y que estas sean implementadas por los repositorios, de esta forma **invertimos la dependencia**.

El diseño a este punto tenía un aspecto similar al ilustrado en la siguiente imagen:



En este momento notamos que ya no tenía sentido que Services y Domain existan por separado por lo cual logramos volver a reunirlos en BusinessLogic.

En la siguiente imagen se puede observar el diseño final:



3. Tabla de asignación de responsabilidades (namespaces)

Project	Namespace	Responsabilidad
BusinessLogic	BusinessLogic	Almacenar toda la lógica de negocio: operaciones y reglas que definen cómo se manejan y procesan los datos.
BusinessLogic	BusinessLogic.Feature_Home	Manejar la lógica de negocio relacionada con el hogar
BusinessLogic	BusinessLogic.Feature_User	Manejar la lógica de negocio relacionada con el usuario
BusinessLogic	BusinessLogic.Feature_Device	Manejar la lógica de negocio relacionada con los dispositivos
BusinessLogic	BusinessLogic.Feature_Device Types	Manejar la lógica de negocio relacionada con los tipos de dispositivos soportados

BusinessLogic	BusinessLogic.Feature_Session	Manejar la lógica de negocio relacionada con las sesiones activas de usuarios
BusinessLogic	BusinessLogic.Feature_Company	Manejar la lógica de negocio relacionada con las empresas
BusinessLogic	BusinessLogic.Exceptions	Definir las excepciones personalizadas
WebApi	WebApi.Controllers	Contener a las clases responsables de interactuar con la capa de lógica de negocio y devolver las respuestas adecuadas a los clientes que consumen la API
WebApi	WebApi.Filters	Contener a las clases responsables de encapsular la información relacionada a los filtros así como la paginación al momento de hacer solicitudes HTTP GET particulares a la API
WebApi	WebApi.Requests	Contener a las clases responsables de encapsular la información transferida a través del body de las peticiones a la API.

WebApi	WebApi.Responses	Contener a las clases responsables de encapsular la información transferida en las respuestas de la API.
DataLayer	DataLayer	Capa de acceso a datos, contiene al contexto (SmartHomeDbContext) y a los repositorios.
DataLayer	DataLayer.Repositories	Contener a las clases encargadas de comunicarse con la base de datos. Concretamente crear, buscar ,actualizar y eliminar registros (CRUD).

4. Justificación del diseño

Para esta aplicación, decidimos utilizar una arquitectura basada en la separación por proyectos. Esta se compone de tres principales: **BusinessLogic**, **DataLayer**, y **WebApi**.

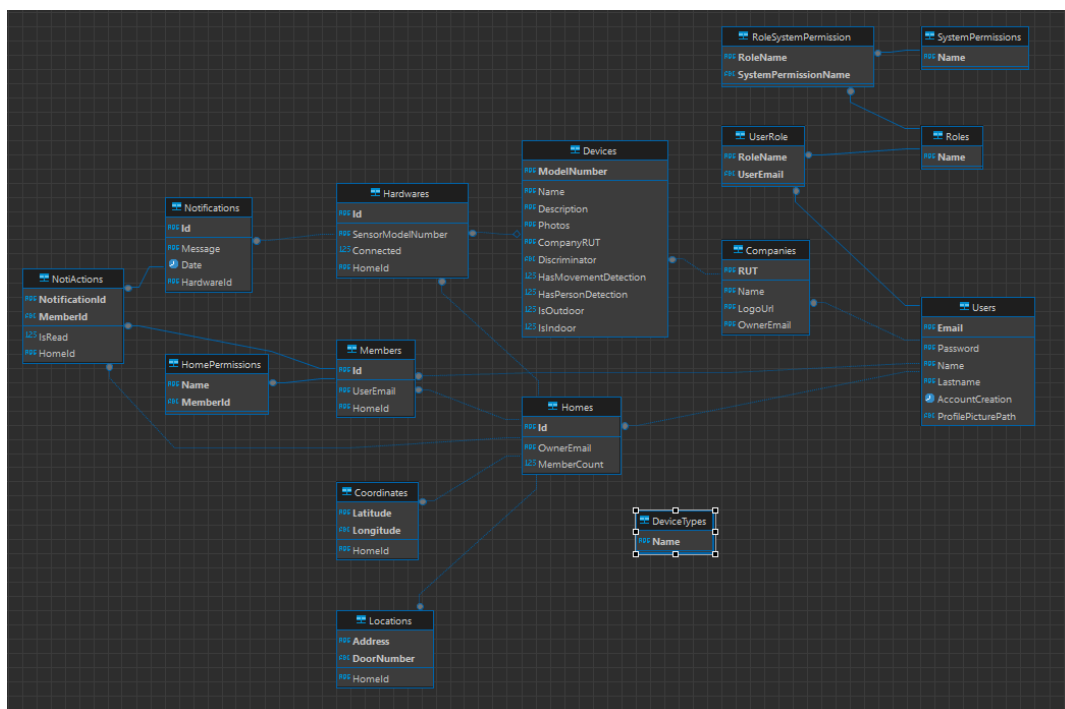
El proyecto **BusinessLogic** contiene la lógica fundamental de la aplicación. Optamos por una estructura organizada por features en lugar de segregar el dominio de los servicios. Esta decisión responde a la necesidad de reducir el acoplamiento entre clases. Al mantener las clases del dominio junto a los servicios que las utilizan, evitamos la complejidad que implicaría tener que acceder a otro proyecto para obtener dichas clases. Asimismo, promovemos una alta cohesión, ya que todas las clases relacionadas a una funcionalidad específica permanecen en el mismo lugar, mientras que una segregación de todos los servicios en un solo espacio podría reunir elementos que, si bien son servicios, no tienen relación directa entre sí.

El proyecto **DataLayer** se encarga de persistir las entidades del dominio mediante el uso de repositorios. Este proyecto es de un nivel más bajo que **BusinessLogic**, por lo que los servicios no deben acceder a los repositorios directamente, ya que esto rompería el principio de inversión de dependencias (DIP). Para abordar esto, decidimos invertir las dependencias utilizando interfaces de repositorio en **BusinessLogic**. De este modo, los servicios utilizan dichas interfaces sin conocer los detalles de su implementación, permitiendo que, si en el futuro se desea cambiar la forma de persistencia (por ejemplo, dejando de usar EF Core para empezar a utilizar Mongo DB), los servicios no requieran modificaciones.

Otra decisión importante en el diseño de los repositorios fue la creación de un repositorio genérico que encapsula las operaciones básicas. Esto evita la duplicación de código y unifica las nomenclaturas para las operaciones CRUD.

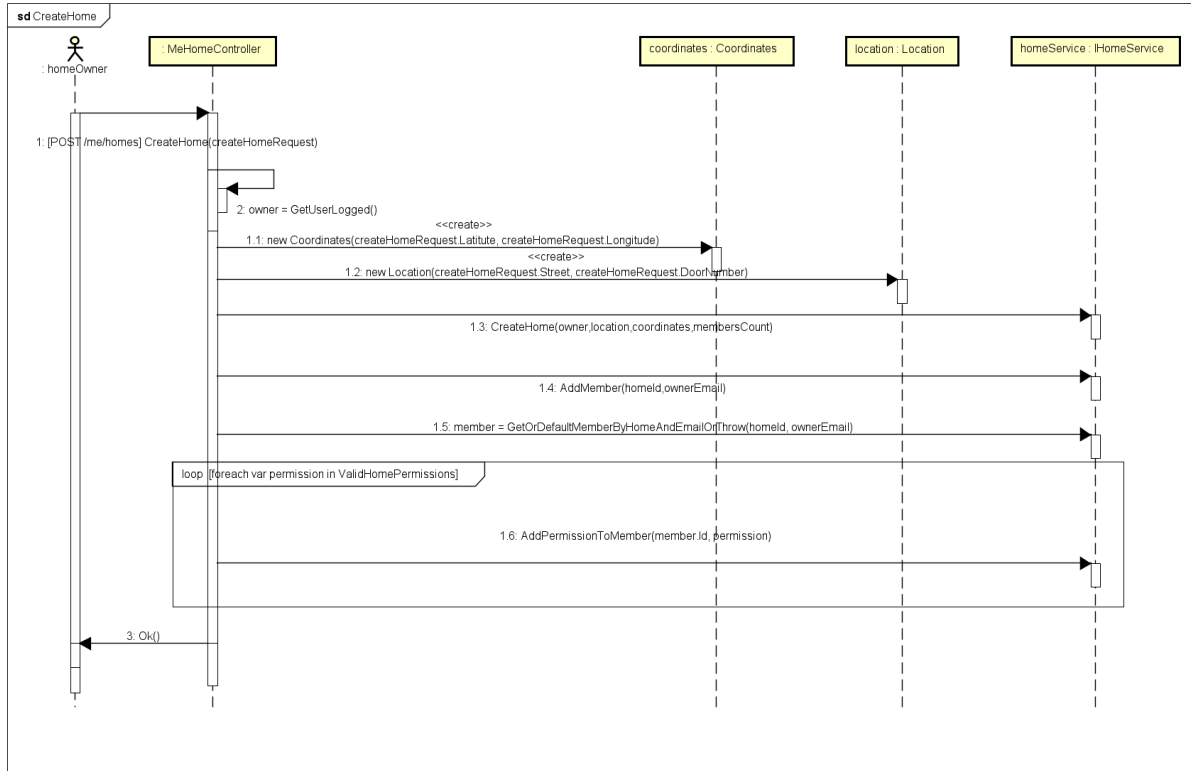
Finalmente, el proyecto **WebApi** media entre el cliente y los servicios. Es fundamental que este proyecto no conozca los detalles de la implementación de los métodos que utiliza. Por lo tanto, los controladores emplean interfaces de los métodos expuestos por los servicios, de manera que se abstraen de la lógica interna, actuando solo como mediadores. Estas interfaces se encuentran en **BusinessLogic**, ya que buscamos que **WebApi** dependa de **BusinessLogic**, al ser este último de un nivel superior.

5. Modelo de tablas en la Base de Datos

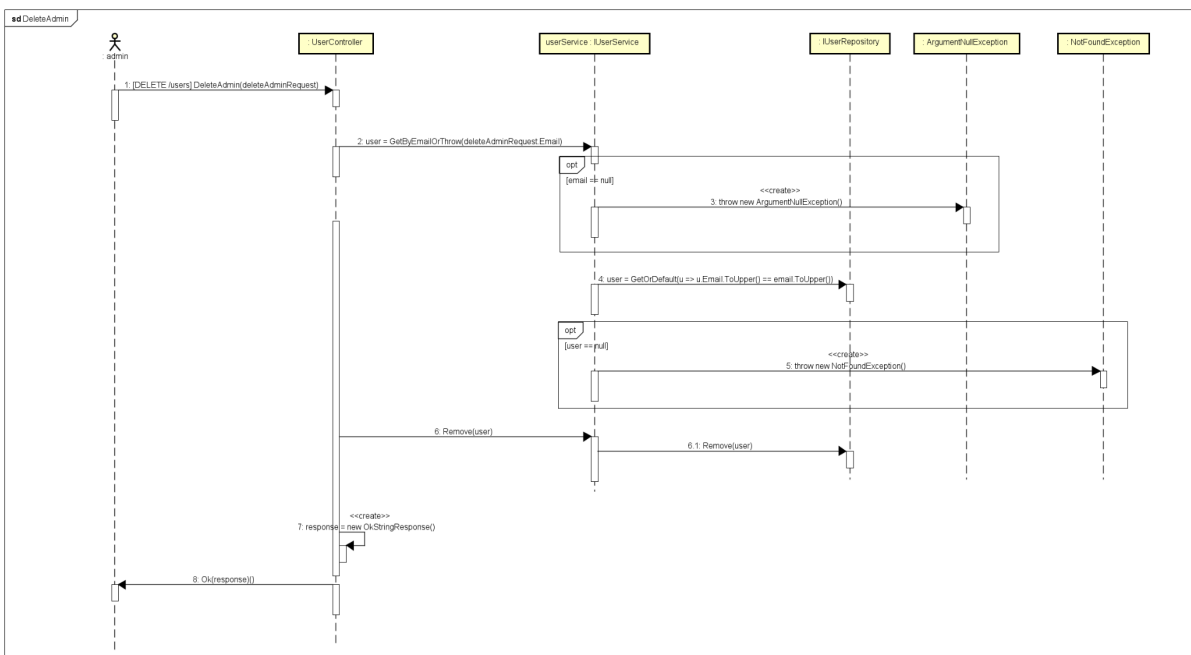


6. Diagramas de interacción

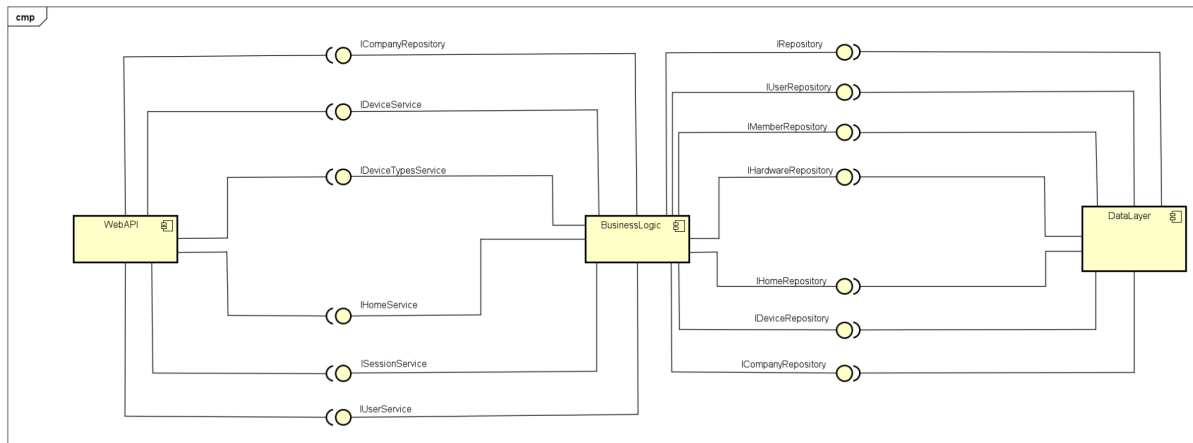
5.1 Creación de un hogar



5.2 Borrar administrador



7. Diagrama de componentes

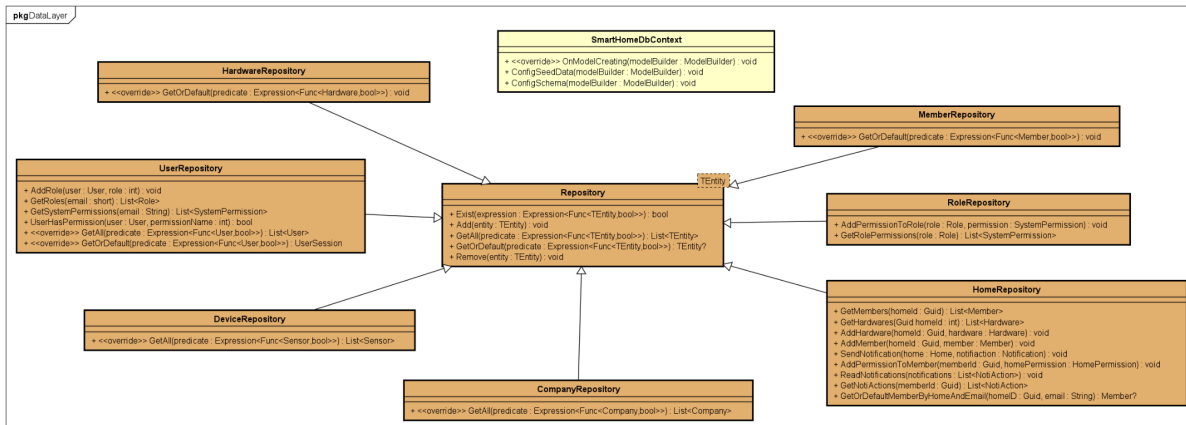


El diagrama refleja la inversión de dependencias entre BusinessLogic y DataLayer, donde las interfaces de los repositorios se definen en BusinessLogic y son implementadas en DataLayer. También se observa que WebAPI depende de las interfaces de los servicios expuestos por BusinessLogic. Este diseño sigue el principio de inversión de dependencias, asegurando que tanto DataLayer como WebAPI dependan de la lógica de negocio, que es el nivel más alto.

8. Diagrama de clases

Debido al tamaño del diagrama no se puede visualizar de una forma correcta en este archivo. Se encuentra en la carpeta Documentación del obligatorio, dentro de la carpeta Diagramas. Los colores distintivos representan diferentes paquetes, utilizándose para visualizar el modelo de mejor manera. Se distinguen los paquetes Feature_Home, Feature_User, Feature_Company, Feature_Device, Feature_DeviceTypes y Feature_Session.

9. Diagrama de repositorios



Los colores distintivos representan diferentes paquetes, utilizándose para visualizar el modelo de mejor manera. Se distingue el paquete Repositories, estando SmartHomeDbContext en la carpeta base del proyecto.

10. Diagrama de controllers

Debido al tamaño del diagrama no se puede visualizar de una forma correcta en este archivo. Se encuentra en la carpeta Documentación del obligatorio, dentro de la carpeta Diagramas. Los colores distintivos representan diferentes paquetes, utilizándose para visualizar el modelo de mejor manera. Se distinguen los paquetes Controllers y Filters, y a su vez se referencian los paquetes Requests y Responses, donde se encuentran las clases correspondientes para el envío y recibimiento de información por la WebApi.

11. Declaracion de autoria

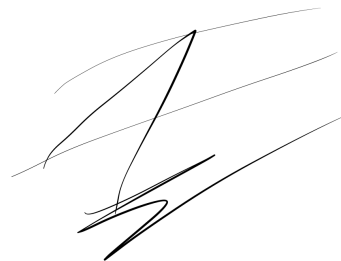
Nosotros, Sebastian Vega, Alejo Fraga y Matias Corvetto, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizamos el Primer Obligatorio de Diseño de Aplicaciones 2;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;

- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue construido por otros, y qué fue construido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Sebastián Vega
8/10/24



Alejo Frogo
8/10/24



Matias Corvetto
8/10/24