
OBLIGATORIO 2 - SmartHome

Diseño de Aplicaciones 2

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 González

Noviembre 2024

Universidad ORT Uruguay

Facultad de Ingeniería

Abstract

El documento aborda los aspectos técnicos y de diseño, siguiendo principios de mantenibilidad y reutilización. Incluye la implementación basada en prácticas de desarrollo ágil como TDD (Desarrollo Guiado por Pruebas) y Clean Code, además de un análisis exhaustivo de métricas de diseño para asegurar la calidad del software. Adicionalmente, se documentan los cambios realizados, el impacto en los componentes del sistema y las pruebas realizadas, con un enfoque en la mantenibilidad, claridad y escalabilidad del código.

La solución entregada persiste todos los datos en una base de datos utilizando Entity Framework Core, incluye documentación detallada de diseño UML, y ofrece tanto una base de datos vacía como otra con datos de prueba. Finalmente, se destaca la capacidad del sistema para ser extendido dinámicamente, garantizando una instalación sencilla y preparada para futuras actualizaciones.

Índice

1. Descripción general del trabajo.....	4
1.1 Herramientas utilizadas.....	4
2. Correcciones más importantes respecto a la entrega anterior.....	5
2.1 Estado de dispositivos en un hogar.....	5
2.2 Persistencia de sesiones.....	5
2.3 Paginación y filtrado corregido.....	5
2.4 Refactorización de Código.....	6
3. Nuevas funcionalidades implementadas e impacto de cambio.....	6
3.1 Asignar nombre a hogar.....	6
3.2 Nombre custom a los dispositivos.....	7
3.3 Administradores y dueños de empresa como dueños de hogar.....	7
3.4 Listado de dispositivos de un hogar.....	8
3.5 Agrupar dispositivos por cuarto.....	8
3.6 Nuevos tipos de dispositivos.....	9
3.7 Validación de modelos en tiempo de ejecución.....	11
3.8 Importar dispositivos inteligentes.....	12
4. Reutilización de componentes y buenas prácticas en frontend.....	14
4.1 Reutilización de componentes.....	14
4.2 Uso de repositorio genérico.....	16
4.3 Uso de interceptor.....	16
4.4 Uso de environments.....	17
5. Aclaraciones y notas adicionales.....	17
5.1 Número de modelo único.....	17
5.2 Incorporación de nuevos importadores.....	17
5.3 Imagen principal de dispositivo.....	17
5.4 Borrado de cuentas de administrador.....	18
5.5 Incorporación de nuevos validadores.....	18
6. Deudas técnicas.....	18
6.1 Migraciones fuera de Smarthome.Datalayer.....	18
6.2 Permitir subir imágenes alojadas en el entorno local del cliente.....	18
7. Diagramas de interacción sobre funcionalidades relevantes.....	18
8. Diagrama de implementación.....	19
8.1 Explicación del diagrama.....	19
9. Tabla de asignación de responsabilidades (namespaces).....	20
10. Aplicación de principios a nivel de paquetes.....	20
11. Aplicación de principios a nivel de clases.....	24
12. Modelo de tablas de la base de datos.....	25
13. Declaración de autoría.....	25
14. Anexo.....	26
14.1 Costo de instalación.....	26
14.2 Seed data.....	26
14.3 Tabla de responsabilidades.....	28
14.4 Obtener Importadores.....	32
14.5 Obtener parámetros de importador.....	32
14.6 Cobertura de código.....	33
14.7 Evidencia de TDD.....	34
14.8 Documentación y Especificación de la API.....	37
14.9 Obtener Cohesión Relaciona a nivel de namespace.....	37
14.10 Agregar foto de perfil al reconvertir cuenta como home owner.....	37

1. Descripción general del trabajo

SmartHome busca implementar una solución escalable y extensible para la gestión de dispositivos inteligentes en los hogares. En esta segunda entrega, se enfocó en corregir problemas detectados en la versión anterior, implementar nuevas funcionalidades, y optimizar la arquitectura del sistema para garantizar su mantenibilidad y capacidad de reutilización. Entre las principales mejoras realizadas se destacan:

Correcciones Críticas: Se abordaron problemas de conexión de dispositivos, persistencia de sesiones y eficiencia en la paginación y filtrado de datos.

Nuevas Funcionalidades: Se añadió la capacidad de personalizar nombres de hogares y dispositivos, agrupar dispositivos por cuartos, incorporar nuevos tipos de dispositivos como lámparas y sensores de movimiento, y validar modelos en tiempo de ejecución.

Importación de Dispositivos Externos: Se implementó un sistema de importadores extensibles, permitiendo la integración de dispositivos desde fuentes externas mediante Reflection y carga dinámica de componentes.

Refactorización y Buenas Prácticas: El código fue optimizado para seguir principios de Clean Code y arquitectura en "vertical slice", maximizando cohesión y minimizando acoplamiento.

Persistencia y Mantenibilidad: Toda la información del sistema se persiste en una base de datos.

1.1 Herramientas utilizadas

- ❖ C# para el desarrollo
- ❖ MSTest para las pruebas unitarias
- ❖ Visual Studio, Rider y Visual Studio Code como IDE
- ❖ Microsoft SQL Server Express 2017, Docker para la base de datos
- ❖ Postman
- ❖ NET Core SDK 8.0/ ASP.NET Core 8.0(C#)
- ❖ Entity Framework Core 8.0
- ❖ Herramientas para modelado de UML (Astah)
- ❖ Angular

2. Correcciones más importantes respecto a la entrega anterior

2.1 Estado de dispositivos en un hogar

En la primera versión de la aplicación no se verifica que los dispositivos que enviaban detecciones estén conectados.

Esto se corrigió en esta segunda versión mediante la implementación de un nuevo endpoint: **PATCH /hardwares/{hardwareId}**, que recibe en el body el nuevo estado del dispositivo (Conectado o Desconectado).

2.2 Persistencia de sesiones

En la entrega anterior, la persistencia de usuarios se limitaba a la memoria volátil. En esta nueva versión, se ha configurado una tabla en la base de datos que permite almacenar los datos de forma persistente.

Con estos cambios, al iniciar sesión, cada usuario generará un ID de sesión único, que permanecerá constante.

SessionId	UserEmail
2D99A8CE-CBD7-491B-AB60-95305D24DD24	alejo@admin.com
D1D25F07-C172-46F3-B315-F8C2D28B04EA	alejo@smarthome.com
E7DE5D4B-EC6E-4695-AFDA-11592F142197	alejo2@smarthome.com
468ADB33-FEC0-4ADF-BDB0-9EB3070AC010	co@smarthome.com
07511BCA-E1D4-43C6-B468-EBCDF6CAA2D0	member@gmail.com
837CC85F-9E32-4A87-9C5E-4A34A1DEC94C	sa@smarthome.com

2.3 Paginación y filtrado corregido







Se optimizó la lógica de filtrado de recursos para reducir la cantidad de datos innecesarios obtenidos de la base de datos, mudando la responsabilidad de esto fuera del servicio.

Ahora, las consultas se construyen directamente en el repositorio, aplicando todos los filtros desde el inicio (incluidos el Offset y Limit). De este modo, se asegura que solo se obtengan de la base de datos los registros exactos que cumplen con los criterios especificados evitando la carga de registros innecesarios.

2.4 Refactorización de Código

Se optimizó la experiencia de Clean Code en toda la aplicación mediante varias mejoras. Se abstrajo la lógica de los controladores, se renombraron métodos con nombres más descriptivos, y se actualizó la estructura de carpetas y namespaces. Además, se revisaron los proyectos para eliminar importaciones innecesarias y se unificaron los formatos en el código, manejo de excepciones, asignaciones, entre otros aspectos.

Además, se refactorizó la lógica para agregar un nuevo usuario, tanto en el servicio como en el controlador de usuario. Luego de estos cambios se logra una extensibilidad mucho mayor para la creación de nuevos usuarios con diferentes roles.

 SmartHome.BusinessLogic	11/14/2024 6:51 PM	File folder
 SmartHome.BusinessLogic.Test	11/11/2024 4:54 PM	File folder
 SmartHome.DataLayer	11/16/2024 5:41 PM	File folder
 SmartHome.DataLayer.Test	11/16/2024 7:18 PM	File folder
 SmartHome.WebApi	11/16/2024 5:16 PM	File folder
 SmartHome.WebApi.Test	11/16/2024 8:15 PM	File folder

3. Nuevas funcionalidades implementadas e impacto de cambio

3.1 Asignar nombre a hogar

Para implementar este requerimiento se agregó una propiedad “Name” a la clase Home, por lo que al momento de crear una instancia de Home se le ofrece al usuario agregar un nombre para la casa.

Este nombre se puede asignar al momento de crear el hogar o posteriormente.

Además de modificar el endpoint **POST me/homes**, ahora se expone el nuevo endpoint **PATCH homes/{homeId}/name** para permitir modificar el nombre del hogar posterior a su creación.

3.2 Nombre custom a los dispositivos

Se permite agregar un alias a un dispositivo en un hogar, al momento de crear un dispositivo al hogar este toma como alias el nombre real del dispositivo, permitiendo luego cambiar este alias ya sea por el dueño del hogar como por un miembro con el permiso adecuado.

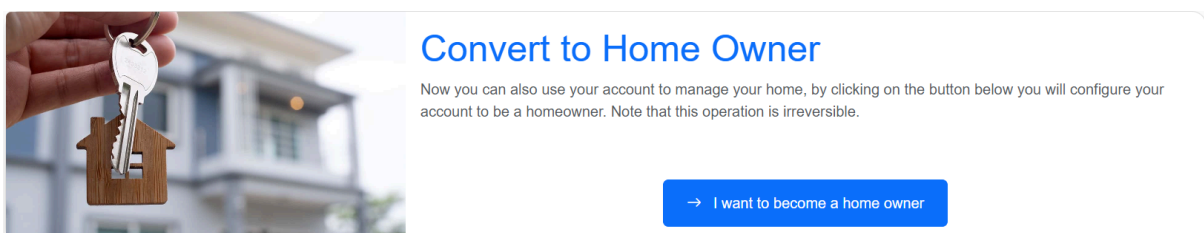
Se implementó el nuevo endpoint **PATCH hardware/{hardwareId}/name**

3.3 Administradores y dueños de empresa como dueños de hogar

Este cambio tuvo un impacto muy bajo debido a que en la versión anterior ya soportábamos que un usuario tenga más de un rol.

Basta con asociar el rol de HomeOwner (agrupación lógica de los permisos de sistema asociados a un dueño de hogar) a los usuarios que desean utilizar su cuenta como dueño de hogar.

En la aplicación presentamos la opción en el menú principal:



Presionando el botón se mostrará un mensaje de confirmación y una vez confirmado se agregara el rol de HomeOwner al usuario activo.

Al momento de confirmar el cambio, se le ofrece al usuario incluir una foto de perfil. (**Ver anexo 14.10**)

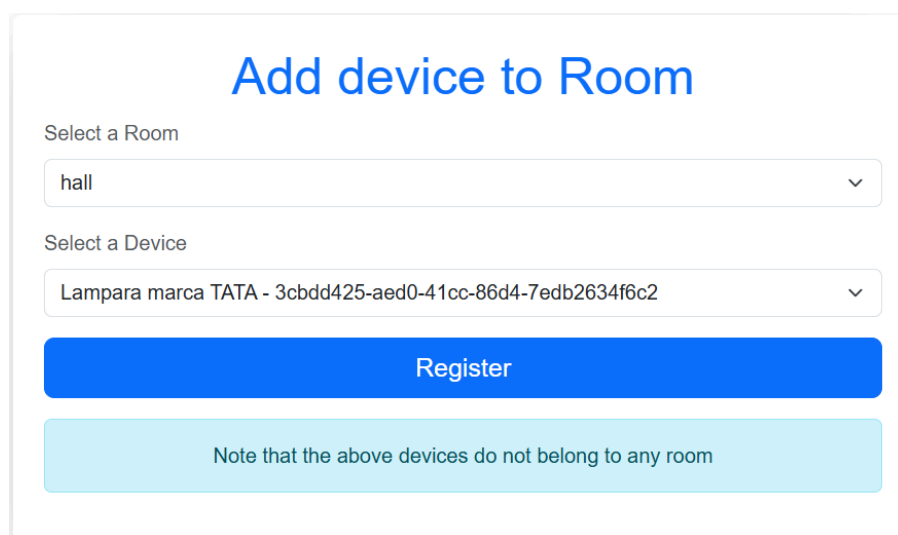
3.4 Listado de dispositivos de un hogar

En esta segunda versión, junto a las nuevas funcionalidades, se actualizó el listado de dispositivos de un hogar, permitiendo ahora filtrar por cuarto.

A su vez, ahora se muestra el estado del dispositivo (en caso de ser una lámpara encendida o apagada y en caso de ser un sensor de puerta si está abierto o cerrado).

3.5 Agrupar dispositivos por cuarto

Ahora el dueño del hogar puede agrupar dispositivos por cuarto asignando a este un nombre. También es posible que miembros con el permiso correspondiente agreguen dispositivos a un cuarto. Para implementar esta funcionalidad se creó una entidad 'Room' el cual está contenido en una 'Home' (Hogar) y tiene una lista de 'Hardwares' (Dispositivos del hogar).



The screenshot shows a web form titled "Add device to Room" in blue text. Below the title, there are two dropdown menus. The first is labeled "Select a Room" and has "hall" selected. The second is labeled "Select a Device" and has "Lampara marca TATA - 3cbdd425-aed0-41cc-86d4-7edb2634f6c2" selected. Below these menus is a prominent blue button labeled "Register". At the bottom of the form, there is a light blue box containing the text: "Note that the above devices do not belong to any room".

Un miembro de un hogar con el permiso correspondiente puede seleccionar un cuarto de entre los existentes al momento y seleccionar un dispositivo a agregar.

El dueño de hogar puede ingresar un nombre (debe ser distinto a cualquier cuarto existente en el hogar) y puede seleccionar los dispositivos que quiera agregar al cuarto (estos dispositivos no deben estar en ningún cuarto al momento de la acción)

3.6 Nuevos tipos de dispositivos

Se agregaron los tipos Lámpara y Sensor de Movimiento, los cuales tienen el siguiente funcionamiento.

Tanto las lámparas como los sensores de movimiento ambos comparten la misma información que los dispositivos anteriores con la integración de funcionalidades características.

Lámpara: las lámparas guardan un estado de encendido o apagado y pueden cambiar de estado a conveniencia del cliente. Cuando una lámpara cambia de estado, ésta envía una notificación a todos los miembros del hogar con el permiso correspondiente.

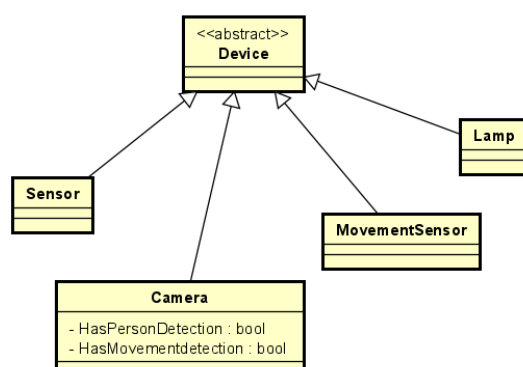
Sensor de Movimiento: similar a las cámaras, estos tienen la funcionalidad de detectar movimiento y al hacerlo notifican a todos los miembros del hogar con el permiso correspondiente.

Sensor: Ahora debe persistir su estado de apertura o cierre.

Para lograr incorporar estos nuevos dispositivos, decidimos modificar el anterior diseño:

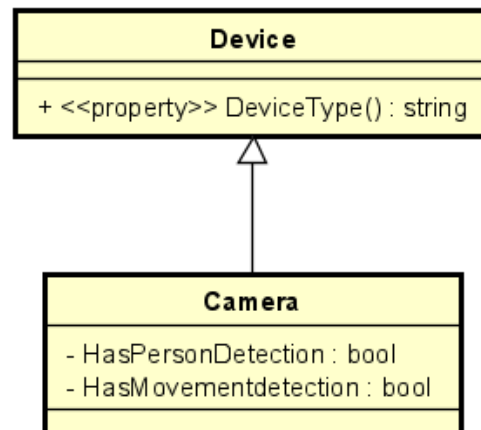
En la primera versión donde solo teníamos cámaras y sensores, nuestro diseño propone que la cámara era un caso particular de un sensor, de ahí la herencia Camara : Sensor.

Dado que ahora tanto la lámpara como el sensor de movimiento no agrega nueva información sobre un sensor, la primera opción sería heredar estos de Sensor pero como ya no queda tan claro que los nuevos dispositivos sean un caso particular de Sensor, un primer refactor fue cambiarle el nombre de Sensor a Device quedando entonces siguiendo esta idea un diseño similar al de la siguiente imagen:



Sin embargo encontramos que no era un buen diseño, debido a que las clases Sensor, Lamp y Movement Sensor quedan vacías lo que incumple clean code (principio DRY), cuesta mantenimiento, conllevan duplicación de pruebas entre otros factores.

Es por esto que se optó por incluir una propiedad “Tipo de dispositivo” en la clase Padre, simplificando el diseño y quitando las clases vacías se logró el diseño final:



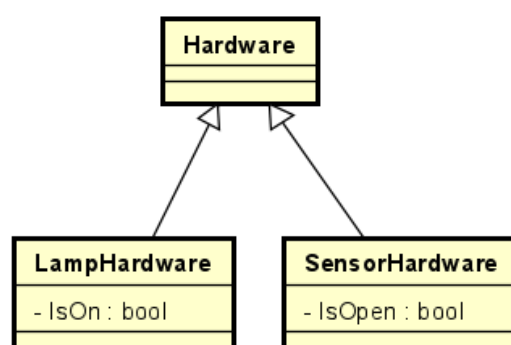
Donde los dispositivos de tipo Sensor tendrán la propiedad “DeviceType” seteada a su nombre y siguiendo la misma idea para Lámpara y Sensor de movimiento.

Nota: Device dejó de ser una clase abstracta.

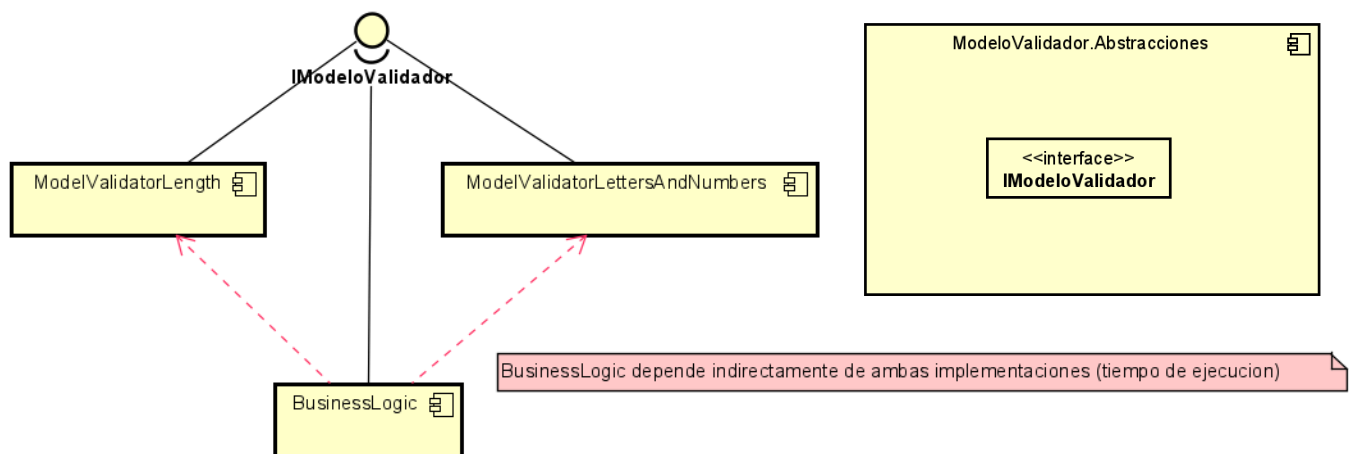
Por otra parte, estas son entidades teóricas siendo que los dispositivos reales que se encuentran en los hogares (llamados hardware en nuestro diseño) son los encargados de asociar a un hogar una “instancia real” de un dispositivo.

Notar que los hardware de lámpara (y ahora sensor) deben persistir su estado (encendido/apagado y abierto/cerrado respectivamente).

Es por esto que se decidió crear clases puntuales para estos tipos de hardware y heredarlos de la clase padre Hardware resultado en lo ilustrado en la siguiente imagen:



3.7 Validación de modelos en tiempo de ejecución



Los usuarios dueños de empresa, ahora pueden elegir una lógica de validación para los modelos de los dispositivos que cree su empresa.

Al momento de crear la empresa, se mostrará un listado de las diferentes opciones de lógicas de validación disponibles en el momento.

Se habilita el desarrollo de lógicas de validación por terceros de forma que añadiendo el archivo de extensión .dll (de la nueva implementación a incorporar en la aplicación) a la carpeta 'Validators' ubicada en la misma dirección que el ejecutable se incorpora el nuevo validador en ejecución sin necesidad de recompilar la aplicación.

❖ Análisis de diagramas

Podemos observar que la interfaz 'IModeloValidador' está definida en el proyecto 'ModeloValidador.Abstracciones' al cual referencia nuestro proyecto BusinessLogic para lograr utilizar el tipo de la interfaz (habilitando el polimorfismo) y así permitir buscar en tiempo de ejecución las implementaciones de los validadores.

```
<Reference Include="..\Reflection\ModeloValidador.Abstracciones.dll" />
```

Implementamos las lógicas de validación 'ValidatorLength' y 'ValidatorLettersAndNumbers' las cuales implementan la interfaz 'IModeloValidador'.

Notar que estas lógicas fueron implementadas en **proyectos independientes**.

Estas lógicas se ubican en la carpeta mencionada anteriormente y se cargan en tiempo de ejecución como se muestra en el diagrama anterior.

❖ **Diagrama de clases de la implementación del requerimiento**

Podemos encontrar el diagrama de clases de este requerimiento en la carpeta **Diagramas/Validacion de modelos**

Nota: Se ejemplifica el uso de la validación en tiempo de ejecución al agregar dispositivos de tipo cámara.

3.8 Importar dispositivos inteligentes

Se puede encontrar el diagrama de clases de este requerimiento en la carpeta **Diagramas/Importador de dispositivos**

Se desarrolló la funcionalidad de importar dispositivos a través de importadores, los cuales pueden ser desarrollados e incorporados por terceros en ejecución (sin necesidad de recompilar).

Para lograr esto se necesitó hacer uso de *Reflection* para obtener todos los importadores disponibles al momento.

El curso de acción es el siguiente:

1. Se ingresa al portal de company owner
2. Se elige la opción de importar dispositivos
3. Se muestran todos los importadores disponibles al momento (*)
4. Se elige uno de los importadores (τ)
5. Se despliega un menú para ingresar los parámetros que necesita el importador elegido (γ)
6. Se presiona el botón para importar los dispositivos. (α)

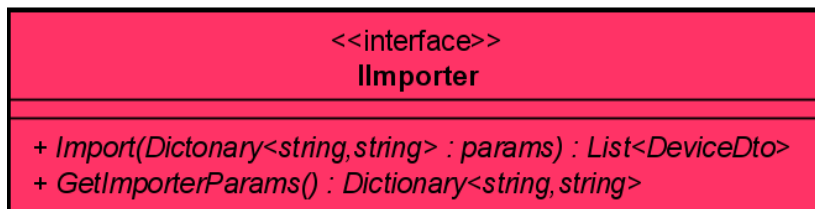
(*):

Para mostrar los importadores disponibles, se efectúa una request al endpoint **GET /importers**.

El mismo comunica al servicio 'DeviceService' la necesidad de listar dispositivos.

Se decidió asignar la responsabilidad de realizar la búsqueda de las implementaciones de los importadores a la clase 'DeviceImporter' la cual se encarga de hacer reflection sobre la carpeta 'Importers' (ubicada en la dirección del ejecutable) para buscar .dll's que contienen implementaciones de la interfaz 'IImporter'.

Interfaz 'IImporter' :



Método para obtener los nombres de los importadores disponibles: [Ver anexo 14.4]

(τ):

Al momento de seleccionar un importador, se dirige al cliente al menú de dicho importador.

Dicho menú debe contener un formulario que solicite el ingreso de todos los parámetros que el importador seleccionado necesite para funcionar correctamente.

Por tanto, la API expone el endpoint **GET /importers/{importerName}/params**

El cual da como respuesta un diccionario con el nombre de cada parámetro y el tipo asociado requeridos por el importador seleccionado.

Para lograr obtener los parámetros que necesita el importador, se lo solicitamos directamente a él, dado que cada importador contiene el método 'GetImporterParams' que devuelve el diccionario mencionado anteriormente.

Siguiendo esta idea, 'DeviceImporter' se encarga de localizar la implementación que coincida con el nombre especificado y que implemente 'IImporter' y a esta le consulta por sus parámetros: [Ver anexo 14.5]

(γ):

Se crea un menú empleando los parámetros obtenidos en el paso anterior.

(α):

En este punto se tomarán todos los valores del formulario y se hará una consulta HTTP al endpoint **POST /importers/{importerName}**

'DeviceImporter' se encargará de localizar nuevamente el importador y ejecutar su método

'Import' con los parámetros recibidos anteriormente.

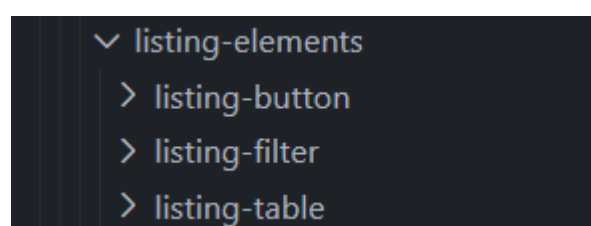
El método devolverá una lista de objetos (DTO's) los cuales finalmente serán mapeados a entidades de dispositivos reales en el servicio 'DeviceService' y agregados a la base de datos.

4. Reutilización de componentes y buenas prácticas en frontend

4.1 Reutilización de componentes

Para ejemplificar la reutilización de componentes que se aplicó en el desarrollo de la aplicación frontend, utilizaremos el listado de elementos.

Para esta funcionalidad, se crearon los elementos 'listing-elements' genéricos, los cuales son 'listing-table', 'listing-button' y 'listing-filter'. Estos componentes luego se utilizaran en un business component, que se encargará de juntarlos para tener una página que permita mostrar, filtrar y paginar los datos traídos por la API.



La tabla se realizó de forma que el usuario que quiera utilizarla no deba preocuparse por cómo crear la tabla, al componente solo hay que pasarle los atributos que tendrán los objetos que les pasaran, y una lista de dichos objetos. Y es el componente el que se encarga de realizar la tabla mostrando todos los objetos, con sus atributos en las columnas correspondientes.

```
export class ListingTableComponent{
  @Input() categories: string[] = [];
  @Input() dataObject: any = {};
```

El botón se va a utilizar para la paginación, por lo que recibe el offset actual, y también cuánto se quiere sumarle a ese offset (se le pueden pasar números negativos en los casos de que se quiera decrementar el offset), también recibe un booleano *disabled*, el cual se utiliza para que el usuario no pueda seguir incrementando el offset cuando ya no hay datos, ni decrementarlo si ya es 0, además recibe un atributo *class* para inyectarle CSS con *Bootstrap*, y un atributo *title* para el texto que irá dentro del botón.

También tiene un *Output*, el cual se encarga de que cuando se haga click sobre él, en caso de ser *disabled* falso, enviará al componente padre el nuevo offset, resultante de sumar el actual con la diferencia establecida; en caso de ser *disabled* verdadero, se enviará el offset actual sin sumarle nada.

El filtro es parecido al botón, este recibe las opciones de filtrado como una lista de string. Por cada una de esas opciones mostrará un *input* (con el input genérico creado anteriormente), el cual para cada uno, en caso de ser actualizado, se le enviará el valor de lo que esté ingresado en el *input* al componente padre, en tiempo real sin necesidad de actualizarlo. Esto nos permite en el componente padre poder filtrar sin necesidad de apretar un botón, únicamente escribiendo ya se va a estar filtrando, todo en tiempo real, aprovechando el *framework* Angular utilizado.

4.2 Uso de repositorio genérico

Se destaca el uso de un repositorio genérico para unificar nomenclatura y reutilizar código.

4.3 Uso de interceptor

Se destaca el uso de un interceptor para agregar el header 'Authorization' al final de cada request, este header sirve para identificar al usuario autenticado en cada request a la API.

```
@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor() {}

  intercept(
    req: HttpRequest<any>,
    next: HttpHandler
  ): Observable<HttpEvent<any>> {
    const token = localStorage.getItem("authToken");

    if (token) {
      const clonedRequest = req.clone({
        setHeaders: {
          Authorization: `${token}`,
        },
      });

      return next.handle(clonedRequest);
    }
    return next.handle(req);
  }
}
```

4.4 Uso de environments

El principal objetivo de usar environments es gestionar las distintas versiones de la API (desarrollo y producción), permitiendo cambiar la versión sin modificar código.

```
▼ environments
  TS environment.development.ts
  TS environment.production.ts
```


5. Aclaraciones y notas adicionales

5.1 Número de modelo único

Se entendió de que el atributo número de modelo ('ModelNumber') de los dispositivos es único y por ende puede ser utilizado como Clave Primaria para la entidad 'Devices'.

Por lo que al momento de registrar dispositivos, además de todas las validaciones necesarias se valida de que el número de modelo sea único.

En la importación de dispositivos, ante una posible duplicación de modelos, se decidió concatenar al número un valor GUID para asegurar la unicidad.

5.2 Incorporación de nuevos importadores

Para incorporar un nuevo importador, se debe implementar la interfaz 'IImporter', compilar el proyecto para obtener el archivo .dll e incorporarlo en la carpeta 'Importers' ubicada en la dirección del ejecutable.

5.3 Imagen principal de dispositivo

La imagen principal de un dispositivo siempre será la cual esté ubicada en la primera posición de la lista.

5.4 Borrado de cuentas de administrador

Al momento de borrar cuentas de administrador, solo se admite borrar las cuentas que sean exclusivamente de administradores, es decir, no se utilicen para ser dueño de hogar (ni ser miembro).

5.5 Incorporación de nuevos validadores

Para incorporar un nuevo importador, se debe implementar la interfaz 'IModeloValidador' ubicada en la dll 'ModeloValidador.Abstracciones' en la dirección del ejecutable, compilar el proyecto para obtener el archivo .dll e incorporarlo en la carpeta 'Validators' también ubicada en la dirección del ejecutable.

Nota : tanto la carpeta ‘Importers’ como ‘Validators’ debe ser creada en la dirección del ejecutable si aún no existe.

6. Deudas técnicas

6.1 Migraciones fuera de Smarthome.Datalayer

Entendemos que las migraciones deberían ubicarse junto a la tecnología, decidimos no moverlas con el objetivo de no perder el historial de migraciones y evitar conflictos.

6.2 Permitir subir imágenes alojadas en el entorno local del cliente

En esta segunda entrega incluimos la funcionalidad de mostrado de imágenes que sean recursos existentes.

Una posible mejora o complemento a esto es permitir el arrastrado de imágenes por parte del cliente y subirlas a un servidor externo para luego renderizarlas.

7. Diagramas de interacción sobre funcionalidades relevantes

Se realizaron diagramas para los cursos de acción de:

- Borrado de administrador
- Generar notificaciones (cambio de estado de una lámpara)

Ambos diagramas se encuentran en la carpeta **Diagramas/Secuencia**

8. Diagrama de implementación

En la carpeta **Diagramas/Componentes** se encuentra el diagrama de implementación de la aplicación.

8.1 Explicación del diagrama

Se separó la arquitectura en tres grandes proyectos: `Smarthome.BusinessLogic`, `Smarthome.DataLayer` y `Smarthome.WebApi`.

❖ **SmartHome.BusinessLogic**

Justificación: Separar la lógica de negocio permite que esta capa sea independiente de la capa de datos y de la capa de presentación. Esto facilita la reutilización de la lógica de negocio en diferentes contextos siguiendo el principio REP (Release/Reuse Equivalency principle) el cual habla de que un componente debe ser **Reusable**.

❖ **SmartHome.DataLayer**

Justificación: Al separar la capa de datos, se logra una abstracción que permite cambiar la implementación de la persistencia de datos sin afectar la lógica de negocio ni la capa de presentación. Esto también facilita la implementación de pruebas unitarias y la migración a diferentes tecnologías de bases de datos si es necesario.

❖ **SmartHome.WebApi**

Justificación: Separar la capa de presentación permite que la interfaz de usuario o los clientes que consumen la API sean independientes de la lógica de negocio y de la capa de datos. Esto facilita la evolución de la interfaz de usuario y la integración con diferentes tipos de clientes (por ejemplo, aplicaciones móviles).

9. Tabla de asignación de responsabilidades (namespaces)

[\(Ver tabla en Anexo 14.3\)](#)

En **Diagramas/Paquetes/Namespaces** se encontrarán diagramas de paquetes para los namespaces más relevantes.

En **Diagramas/Paquetes/Smarthome.png** se encuentra el diagrama general de paquetes de todo el backend de la aplicación.

10. Aplicación de principios a nivel de paquetes

10.1 Cohesión

Desde un principio, el desarrollo de la aplicación se realizó utilizando la ‘vertical slice architecture’ (desarrollo de código por features).

Se destacan los namespaces:

- Smarthome.BusinessLogic.Homes
- Smarthome.BusinessLogic.Companies
- Smarthome.BusinessLogic.Devices
- Smarthome.BusinessLogic.DeviceTypes
- Smarthome.BusinessLogic.Sessions
- Smarthome.BusinessLogic.Users

Cada uno de estos namespaces cumplen tanto **CCP** (Common Closure Principle) como **CRP** (Common Reuse Principle), esto es debido a que cada clase embebida dentro de cada uno de estos namespaces sigue el mismo lineamiento de cambio que el resto de clases que comparten el mismo namespace, siguiendo esta idea, también se cumple que para reutilizar una de estas clases debemos traer a la mayoría de las que están incluidas en el namespace (están ‘atadas’ entre sí).

Para ejemplificar lo mencionado, tomaremos la clase `Hardware` de `Smarthome.BusinessLogic.Homes`. Si impactamos en ella, directamente afectamos a las clases : `Home`, `HardwareData`, `Room`, `HomeService`, `LampHardware`, `SensorHardware`, entre otras. Esto es debido a que todas estas clases siguen el **mismo lineamiento de cambio** (todas estas se crearon debido a la necesidad de manejar casas (Homes) en la aplicación).

De igual forma, si se quisiera utilizar la clase `hardware`, deberíamos llevar con ella a todas las clases mencionadas anteriormente, dado que `hardware` por sí sola no tiene razón de existencia, es decir, **se deben reutilizar en conjunto**.

Teniendo en cuentas la métrica de **cohesión relacional a nivel de assembly** (se muestra el cálculo en el apartado siguiente), podemos observar que Smarthome.WebApi y Smarthome.BusinessLogic tienen un nivel saludable (entre 1.5 y 4), esto es debido al desarrollo guiado por la vertical slice architecture. Por otra parte, notamos que el valor en Smarthome.DataLayer está un poco por debajo de lo deseado, pero es razonable, dado que la mayoría de los repositorios son independientes entre sí, siendo los servicios los encargados de depender de varios repositorios en caso de necesitarlo.

Se consideró que utilizar la métrica de cohesión relacional a nivel de namespace nos podría brindar una mayor claridad respecto a la cohesión de cada namespace y visualizar aún mejor los resultados de la vertical slice architecture.

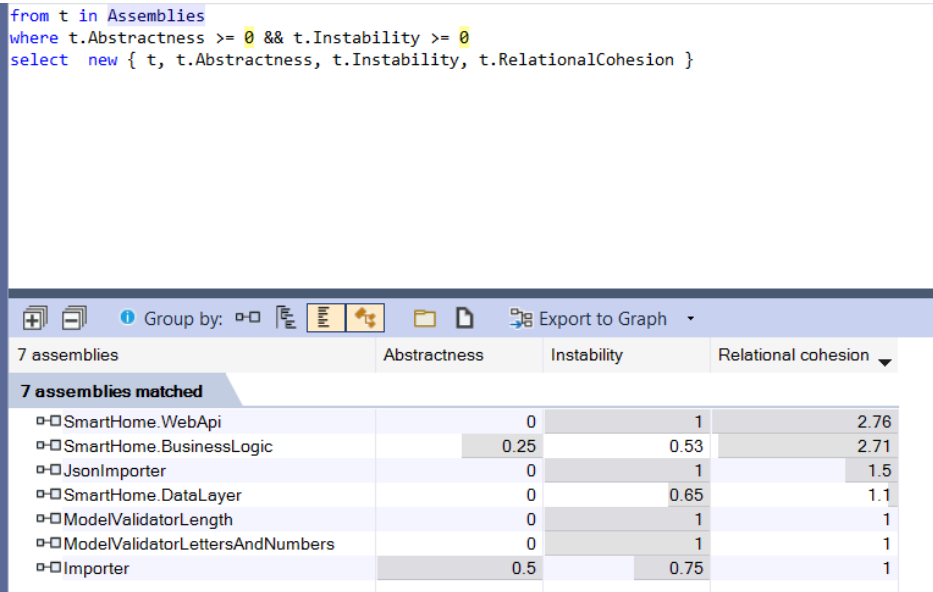
La consulta ejecutada en **ndepend**: <https://www.ndepend.com/> (Ver anexo 14.9) dio como resultado lo ilustrado en la siguiente imagen:

22 namespaces	Cohesion_Relational
22 namespaces matched	
{ } ModelValidator.Length	1
{ } Importer	0.5
{ } Smarthome.BusinessLogic	1
{ } <u>Smarthome.BusinessLogic.Users</u>	2
{ } Smarthome.BusinessLogic.Sessions	1.5
{ } Smarthome.BusinessLogic.Homes	4.5
{ } Smarthome.BusinessLogic.Exceptions	0.5
{ } Smarthome.BusinessLogic.DeviceTypes	0.33
{ } Smarthome.BusinessLogic.Devices	1.75
{ } Smarthome.BusinessLogic.Companies	1
{ } Smarthome.BusinessLogic.Args	0.083
{ }	0.25
{ } <u>Smarthome.DataLayer</u>	0.19
{ }	0.036
{ } Smarthome.WebApi	1
{ } Smarthome.WebApi.Responses	0.05
{ } Smarthome.WebApi.Requests	0.037
{ } Smarthome.WebApi.Migrations	0.019
{ } Smarthome.WebApi.Filters	0.17
{ } <u>Smarthome.WebApi.Controllers</u>	0.83
{ } ModelValidator.LettersAndNumbers	1
{ } JsonImporter	1

Podemos observar particularmente en Smarthome.BusinessLogic que la media de la cohesión relacional entre sus namespace es saludable, con algunas excepciones.

10.2 Acoplamiento

Se obtuvo el cálculo de las métricas Inestabilidad, Abstracción y Cohesión relacional a través de la herramienta **ndepend**: <https://www.ndepend.com/>



Observando el diagrama general de paquetes ubicado en **Diagramas/Paquetes**,

Podemos observar que se cumple **SDP** (Stable Dependencies Principle) en los paquetes más importantes de la aplicación, donde Smarthome.WebApi depende de Smarthome.DataLayer y Smarthome.BusinessLogic (ambos con mayor estabilidad) y Smarthome.DataLayer depende de Smarthome.BusinessLogic (más estable).

Analizando los diagramas independientes de cada namespace ubicados en Diagramas/Paquetes/Namespace podemos observar que se cumple **ADP** (Acyclic Dependencies Principle) al no existir dependencias cíclicas entre namespaces.

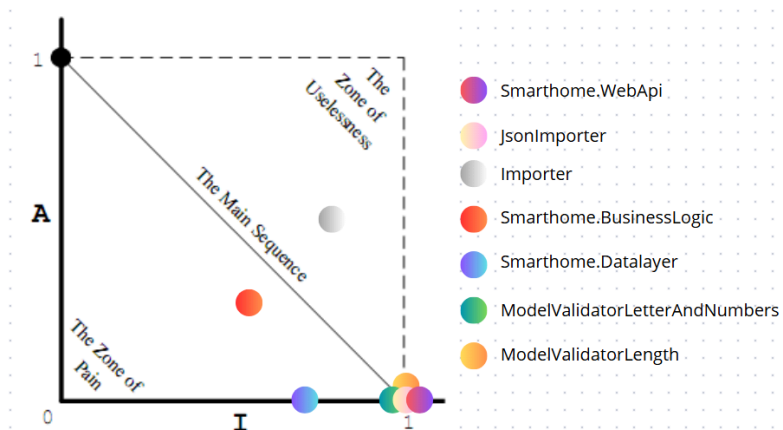
En la siguiente gráfica, podemos ver que los assemblies Smarthome.WebApi, Importer, ModelValidatorLettersAndNumbers y ModelValidatorLength se encuentran en la parte inferior derecha de la gráfica siendo esta una zona saludable.

Hallamos que el resultado de las métricas de Inestabilidad y Abstracción es consistente particularmente en Smarthome.WebApi, dado que es puramente concreto, pero nadie depende de él, siendo este el encargado de depender del resto de los módulos para lograr exponer su funcionalidad.

Por otra parte hallamos el resultado de Smarthome.BusinessLogic coherente, dado que a pesar de que contiene a las clases de dominio (deberían ubicarse en la zona de dolor), también

contiene a la lógica de los servicios y sus interfaces y es por esto, su ubicación intermedia en la gráfica.

Por último, notamos que el resultado de Datalayer también es coherente, dado que es puramente concreto (implementación de repositorios) y Smarthome.WebApi depende de él.



El anterior análisis nos permite decir que se cumple con **SAP** (Stable Abstractions Principle).

11. Aplicación de principios a nivel de clases

En **Diagramas/Clases** se encontraran diagramas de clases para los tres principales namespaces de la aplicación (Smarthome.WebApi, Smarthome.DataLayer, Smarthome.BusinessLogic).

Se cumple con DIP (Dependency inversion principle) dado que tanto Smarthome.Datalayer como Smarthome.WebApi ambos dependen de Smarthome.BusinessLogic, esto se logró invirtiendo las dependencias mediante interfaces.

Para ejemplarizar el cumplimiento de OCP (Open Closed Principle), tomaremos el mecanismo de permisos de sistema en la aplicación, donde no se consulta directamente si el

usuario cuenta con un permiso particular, sino que se comprueba si alguno de los roles que tiene el usuario contiene dicho permiso, habilitando la implementación de nuevos permisos sin modificar el código existente.

Se desarrolló cada parte de la aplicación teniendo en cuenta el balance entre cohesión y acoplamiento, siempre intentando minimizar el acoplamiento y maximizar la cohesión de las clases.

Se destaca la implementación del patrón Controller de GRASP particularmente en Smarthome.WebApi, donde cada clase dentro del namespace Controllers tiene como objetivo manejar eventos de la aplicación frontend.

Es relevante mostrar el uso de indirección de GRASP, por ejemplo en las clases ModelValidator y DeviceImporter, donde se les asigna la responsabilidad de mediar con otros módulos con lineamientos de cambio diferente.

12. Modelo de tablas de la base de datos

Se puede visualizar el diagrama de tablas de la base de datos en **Diagramas/Modelo**

13. Declaración de autoría

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 Segundo 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.



Sebastian Vega
21/11/24



Alejo Fraga
21/11/2024



Matias Corvetto
21/11/24

14. Anexo

14.1 Costo de instalación

En la carpeta “Aplicación” se encuentran dos carpetas: Frontend y WebApi.

Frontend: incluye los archivos necesarios (dist) para poder hacer un deploy de la aplicación en IIS o en algún servidor.

WebApi: Incluye la carpeta publish, con el release de la API.

Connection string: Se debe modificar el archivo appsettings.json con el connection string a utilizar.

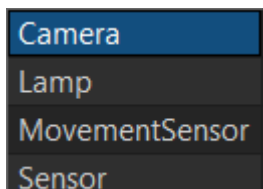
En la carpeta “Base de datos” se encuentran dos carpetas: **DBVacía** y **DBConCasosDePueba**. La primera contiene un archivo .bak con el respaldo de la base de datos solamente con la seed data y también incluye como alternativa los archivos DDL.sql y SeedData.sql.

Por otro lado, **DBConCasosDePrueba** contiene un archivo .bak con el respaldo de la base de datos con datos de prueba y también incluye como alternativa los archivos DDL.sql, SeedData.sql y DML.sql.

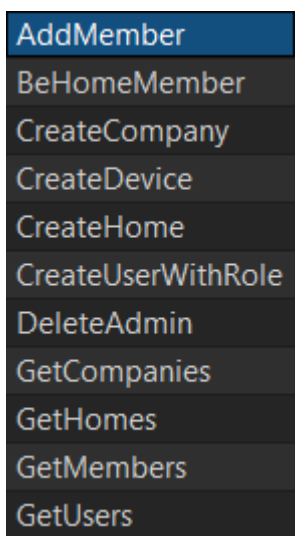
14.2 Seed data

Para iniciar la aplicación, se cuenta con la siguiente información del administrador inicial, permisos existentes, roles y tipos de dispositivos soportados:

❖ Tipos de dispositivos soportados:



❖ Permisos de Sistema:



❖ Roles:

Admin
CompanyOwner
HomeOwner

❖ Permisos de sistema asociados a cada rol:

Admin	CreateUserWithRole
Admin	DeleteAdmin
Admin	GetCompanies
Admin	GetUsers
CompanyOwner	CreateCompany
CompanyOwner	CreateDevice
HomeOwner	AddMember
HomeOwner	BeHomeMember
HomeOwner	CreateHome
HomeOwner	GetHomes
HomeOwner	GetMembers

❖ Administrador inicial:

ABC Email	ABC Password	ABC Name	ABC Lastname	AccountCreation
sa@smarthome.com	sA\$a1234	System	Admin	2024-11-12 12:18:42.918

ABC RoleName	ABC UserEmail
Admin	sa@smarthome.com

14.3 Tabla de responsabilidades

Project	Namespace	Responsabilidad
Smarthome.BusinessLogic	Smarthome.BusinessLogic	Almacenar toda la lógica de negocio: operaciones y reglas que definen cómo se manejan y procesan los datos.
Smarthome.BusinessLogic	Smarthome.BusinessLogic.Args	Define argumentos personalizados a ser utilizados en servicios y repositorios
Smarthome.BusinessLogic	Smarthome.BusinessLogic.Homes	Manejar la lógica de negocio relacionada con el hogar
Smarthome.BusinessLogic	Smarthome.BusinessLogic.Users	Manejar la lógica de negocio relacionada con el usuario
Smarthome.BusinessLogic	Smarthome.BusinessLogic.Devices	Manejar la lógica de negocio relacionada con los dispositivos
Smarthome.BusinessLogic	Smarthome.BusinessLogic.DeviceTypes	Manejar la lógica de negocio relacionada con los tipos de dispositivos soportados

Smarthome.BusinessLogic	Smarthome.BusinessLogic.Sessions	Manejar la lógica de negocio relacionada con las sesiones activas de usuarios
Smarthome.BusinessLogic	Smarthome.BusinessLogic.Companies	Manejar la lógica de negocio relacionada con las empresas
Smarthome.BusinessLogic	Smarthome.BusinessLogic.Exceptions	Definir las excepciones personalizadas
Smarthome.WebApi	Smarthome.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
Smarthome.WebApi	Smarthome.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

Smarthome.WebApi	Smarthome.WebApi.Requests	Contener a las clases responsables de encapsular la información transferida a través del body de las peticiones a la API.
Smarthome.WebApi	Smarthome.WebApi.Responses	Contener a las clases responsables de encapsular la información transferida en las respuestas de la API.
Smarthome.DataLayer	Smarthome.DataLayer	Capa de acceso a datos, contiene al contexto (SmartHomeDbContext) y a los repositorios.
Smarthome.DataLayer	Smarthome.DataLayer.Repositories	Contener a las clases encargadas de comunicarse con la base de datos. Concretamente crear, buscar ,actualizar y eliminar registros (CRUD).
Importer	Importer	Contener a las clases encargadas de encapsular la información necesaria para lograr incorporar nuevos importadores en tiempo de ejecución

14.4 Obtener Importadores

```
public List<string> GetImporterNames()
{
    var path :string = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, folderName);
    const string searchPattern = "*.dll";
    var dllFiles :string[] = Directory.GetFiles(path, searchPattern);

    var assemblies :Assembly[] = dllFiles.Select(Assembly.LoadFrom).ToArray();
    var importerTypes :List<Type> = assemblies // Assembly[]
        .SelectMany(a :Assembly => a.GetTypes())
        .Where(t :Type => typeof(IImporter).IsAssignableFrom(t)) // IEnumerable<Type>
        .ToList();

    if (!importerTypes.Any())
    {
        throw new InvalidOperationException(message: "No importers found");
    }

    return importerTypes.Select(t :Type => t.Name).ToList();
}
```

14.5 Obtener parámetros de importador

```
public Dictionary<string, string> GetImporterParams(string importerName)
{
    var path :string = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, folderName);
    const string searchPattern = "*.dll";
    var dllFiles :string[] = Directory.GetFiles(path, searchPattern);

    var assemblies :Assembly[] = dllFiles.Select(Assembly.LoadFrom).ToArray();
    var importerType = assemblies // Assembly[]
        .SelectMany(a :Assembly => a.GetTypes()) // IEnumerable<Type>
        .FirstOrDefault(t :Type => typeof(IImporter).IsAssignableFrom(t) && t.Name == importerName);
    if (importerType == null)
    {
        throw new InvalidOperationException(message: "Importer not found");
    }

    var importer = (IImporter?)Activator.CreateInstance(importerType);
    if (importer == null)
    {
        throw new InvalidOperationException(message: "caould not create instance of importer"); Typo in string: 'caould'.
    }

    return importer.GetImporterParams();
}
```


14.6 Cobertura de código

Code Coverage 91%			
Package	Line Rate	Branch Rate	Health
ModelValidatorLength	100%	100%	✓
ModelValidatorLettersAndNumbers	100%	62%	✓
SmartHome.BusinessLogic	89%	75%	—
SmartHome.DataLayer	100%	95%	✓
SmartHome.WebApi	92%	66%	✓
Summary	91% (1965 / 2151)	76% (278 / 364)	✓
Minimum allowed line rate is 85%			

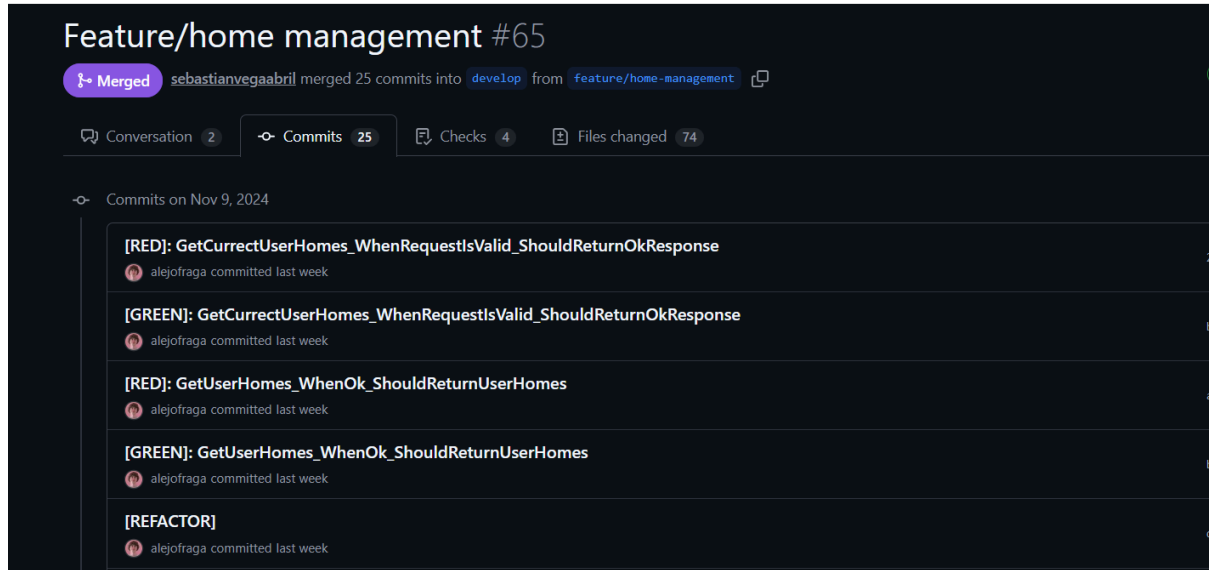
La cobertura obtenida mediante las pruebas automatizadas refleja un 91% de líneas de código ejecutadas, con ciertas variaciones entre módulos.

Una cobertura de código amplia es fundamental en cualquier proyecto, ya que permite extender la aplicación de forma segura, evitando incumplimientos de requisitos y reduciendo el riesgo de introducir errores en código ya desplegado en producción. Aunque el nivel actual de cobertura es aceptable, podría mejorarse aún más. Para una empresa donde cualquier retroceso representa una pérdida significativa, sería ideal aspirar a la máxima cobertura posible, previniendo con mayor eficacia casos de inconsistencias y errores.

En el flujo de trabajo correspondiente a la primera entrega, cometimos el error de no monitorear constantemente la cobertura en cada *pull request*. Esto derivó en la acumulación progresiva de código sin verificar, lo que en algunos casos obligó a realizar iteraciones adicionales para corregir errores en secciones que se consideraban finalizadas.

En esta segunda entrega, nos comprometimos a sí mantener la cobertura de código en todas las instancias como una prioridad, en beneficio del proyecto y su calidad a largo plazo.

14.7 Evidencia de TDD



The screenshot shows a GitHub pull request titled "Feature/home management #65". It was merged by sebastianvegaabril, merging 25 commits into the develop branch from the feature/home-management branch. The pull request has 2 conversations, 25 commits, 4 checks, and 74 files changed. Below the summary, there is a list of commits from November 9, 2024, by alejofraga. The commits are: [RED]: GetCurrentUserHomes_WhenRequestIsValid_ShouldReturnOkResponse, [GREEN]: GetCurrentUserHomes_WhenRequestIsValid_ShouldReturnOkResponse, [RED]: GetUserHomes_WhenOk_ShouldReturnUserHomes, [GREEN]: GetUserHomes_WhenOk_ShouldReturnUserHomes, and [REFACTOR].

- ❖ [RED]: GetCurrentUserHomes_WhenRequestIsValid_ShouldReturnOkResponse: Se agrega nueva prueba automatizada para GetUserHomes del controlador MeHomeController.

```
73     _homeServiceMock.VerifyAll();
74     _httpContextMock.VerifyAll();
75 }
76 +
77 + [TestMethod]
78 + public void GetCurrentUserHomes_WhenRequestIsValid_ShouldReturnOkResponse()
79 + {
80 +     var userLogged = new User()
81 +     {
82 +         Name = "Alejo", Lastname = "Fraga", Email = "alejofraga22v2@gmail.com", Password = "12A2$$#!ssasd"
83 +     };
84 +     var newHome = new Home()
85 +     {
86 +         OwnerEmail = userLogged.Email,
87 +         Location = new Location("Street", "123"),
88 +         Coordinates = new Coordinates("123", "123"),
89 +         MemberCount = 1
90 +     };
91 +     var expectedHomes = new List<Home>() { newHome };
92 +
93 +     _homeServiceMock
94 +         .Setup(hs => hs.GetUserHomes(It.IsAny<User>()))
95 +         .Returns(expectedHomes);
96 +     _httpContextMock.Setup(h => h.Items["UserLogged"])
97 +         .Returns(userLogged);
98 +     _meHomeController.ControllerContext = new ControllerContext { HttpContext = _httpContextMock.Object };
99 +     var controllerResponse = _meHomeController.GetUserHomes();
100 +
101 +     var result = controllerResponse as ObjectResult;
102 +     result.Should().NotBeNull();
103 +
104 +     var responseValue = result.Value.GetType().GetProperty("Message")?.GetValue(result.Value).ToString();
105 +     responseValue.Should().NotBeNull();
106 +
107 +     responseValue.Should().Be("Operation was successfully performed");
108 +     _homeServiceMock.VerifyAll();
109 +     _httpContextMock.VerifyAll();
110 + }
111 }
```

- ❖ [GREEN]: GetCurrentUserHomes_WhenRequestIsValid_ShouldReturnOkResponse: Se agrega implementación para la función GetUserHomes en el controlador.

```
39 +         StatusCode = (int)HttpStatusCode.Created
40 +     };
41 + }
42 +
43 + [HttpGet]
44 + [AuthenticationFilter]
570 + }
571 +
572 + public List<Home> GetUserHomes(User user)
573 + {
574 +     throw new NotImplementedException();
575 + }
576 + }
577 +
578 +
579 +
580 +
581 +
582 +
583 +
584 +
585 +
586 +
587 +
588 +
589 +
590 +
591 +
592 +
593 +
594 +
595 +
596 +
597 +
598 +
599 +
600 +
601 +
602 +
603 +
604 +
605 +
606 +
607 +
608 +
609 +
610 +
611 +
612 +
613 +
614 +
615 +
616 +
617 +
618 +
619 +
620 +
621 +
622 +
623 +
624 +
625 +
626 +
627 +
628 +
629 +
630 +
631 +
632 +
633 +
634 +
635 +
636 +
637 +
638 +
639 +
640 +
641 +
642 +
643 +
644 +
645 +
646 +
647 +
648 +
649 +
650 +
651 +
652 +
653 +
654 +
655 +
656 +
657 +
658 +
659 +
660 +
661 +
662 +
663 +
664 +
665 +
666 +
667 +
668 +
669 +
670 +
671 +
672 +
673 +
674 +
675 +
676 +
677 +
678 +
679 +
680 +
681 +
682 +
683 +
684 +
685 +
686 +
687 +
688 +
689 +
690 +
691 +
692 +
693 +
694 +
695 +
696 +
697 +
698 +
699 +
700 +
701 +
702 +
703 +
704 +
705 +
706 +
707 +
708 +
709 +
710 +
711 +
712 +
713 +
714 +
715 +
716 +
717 +
718 +
719 +
720 +
721 +
722 +
723 +
724 +
725 +
726 +
727 +
728 +
729 +
730 +
731 +
732 +
733 +
734 +
735 +
736 +
737 +
738 +
739 +
740 +
741 +
742 +
743 +
744 +
745 +
746 +
747 +
748 +
749 +
750 +
751 +
752 +
753 +
754 +
755 +
756 +
757 +
758 +
759 +
760 +
761 +
762 +
763 +
764 +
765 +
766 +
767 +
768 +
769 +
770 +
771 +
772 +
773 +
774 +
775 +
776 +
777 +
778 +
779 +
780 +
781 +
782 +
783 +
784 +
785 +
786 +
787 +
788 +
789 +
790 +
791 +
792 +
793 +
794 +
795 +
796 +
797 +
798 +
799 +
800 +
801 +
802 +
803 +
804 +
805 +
806 +
807 +
808 +
809 +
810 +
811 +
812 +
813 +
814 +
815 +
816 +
817 +
818 +
819 +
820 +
821 +
822 +
823 +
824 +
825 +
826 +
827 +
828 +
829 +
830 +
831 +
832 +
833 +
834 +
835 +
836 +
837 +
838 +
839 +
840 +
841 +
842 +
843 +
844 +
845 +
846 +
847 +
848 +
849 +
850 +
851 +
852 +
853 +
854 +
855 +
856 +
857 +
858 +
859 +
860 +
861 +
862 +
863 +
864 +
865 +
866 +
867 +
868 +
869 +
870 +
871 +
872 +
873 +
874 +
875 +
876 +
877 +
878 +
879 +
880 +
881 +
882 +
883 +
884 +
885 +
886 +
887 +
888 +
889 +
890 +
891 +
892 +
893 +
894 +
895 +
896 +
897 +
898 +
899 +
900 +
901 +
902 +
903 +
904 +
905 +
906 +
907 +
908 +
909 +
910 +
911 +
912 +
913 +
914 +
915 +
916 +
917 +
918 +
919 +
920 +
921 +
922 +
923 +
924 +
925 +
926 +
927 +
928 +
929 +
930 +
931 +
932 +
933 +
934 +
935 +
936 +
937 +
938 +
939 +
940 +
941 +
942 +
943 +
944 +
945 +
946 +
947 +
948 +
949 +
950 +
951 +
952 +
953 +
954 +
955 +
956 +
957 +
958 +
959 +
960 +
961 +
962 +
963 +
964 +
965 +
966 +
967 +
968 +
969 +
970 +
971 +
972 +
973 +
974 +
975 +
976 +
977 +
978 +
979 +
980 +
981 +
982 +
983 +
984 +
985 +
986 +
987 +
988 +
989 +
990 +
991 +
992 +
993 +
994 +
995 +
996 +
997 +
998 +
999 +
1000 +
```

Se agrega el método sin cuerpo para satisfacer la interfaz.

- ❖ [RED]: GetUserHomes_WhenOk_ShouldReturnUserHomes: Se agrega nueva prueba automatizada para GetUserHomes del servicio HomeService.

```

1281     }
1282 +
1283 +     [TestMethod]
1284 +     public void GetUserHomes_WhenOk_ShouldReturnUserHomes()
1285 +     {
1286 +         var user = GetValidUser();
1287 +         const string expectedAddress = "Golden street";
1288 +         const string doorNumber = "818";
1289 +         var expectedLocation = new Location(expectedAddress, doorNumber);
1290 +         const string latitude = "123";
1291 +         const string longitude = "456";
1292 +         var expectedCoordinates = new Coordinates(latitude, longitude);
1293 +         const int expectedMemberCount = 3;
1294 +         var home = new Home
1295 +         {
1296 +             Owner = user,
1297 +             OwnerEmail = user.Email,
1298 +             Location = expectedLocation,
1299 +             Coordinates = expectedCoordinates,
1300 +             MemberCount = expectedMemberCount
1301 +         };
1302 +         var expectedHomes = new List<Home> { home };
1303 +         _homeRepositoryMock.Setup(repo => repo.GetAll(It.IsAny<Expression<Func<Home, bool>>>()))
1304 +             .Returns(expectedHomes);
1305 +         var homes = _homeService.GetUserHomes(user);
1306 +         homes.Should().BeEquivalentTo(expectedHomes);
1307 +     }
1308 }

```

- ❖ [GREEN]: GetUserHomes_WhenOk_ShouldReturnUserHomes: Se implementa la función GetUserHomes para cumplir con la funcionalidad.

```

public List<Home> GetUserHomes(User user)
{
    throw new NotImplementedException();
}

```

```

572 + public List<Home> GetUserHomes(User user)
573     {
574 +     return homeRepository.GetAll(h => h.Members.Any(m => m.UserEmail == user.Email));
575     }

```

- ❖ [REFACTOR] : Finalmente se refactoriza el código para poder ser leído y entendido de manera más limpia. Aquí se agrega una clase DTO para devolver una respuesta y así devolver información sin campos innecesarios.

<pre> return new ObjectResult(new { InnerCode = "Ok", Message = "Operation was successfully performed", Data = homes }) { StatusCode = (int)HttpStatusCode.OK } </pre>	<pre> 51 52 + var response = homes.Select(home => new GetUserHomesResponse 53 + { 54 + Id = home.Id, 55 + Name = home.Name! 56 + }).ToList(); 57 + 58 return new ObjectResult(new 59 { 60 InnerCode = "Ok", 61 Message = "Operation was successfully performed", 62 + Data = response 63 }) 64 { 65 StatusCode = (int)HttpStatusCode.OK </pre>
--	---

14.8 Documentación y Especificación de la API

En la carpeta **Documentacion/Segunda Entrega** se encuentra el archivo: '*Especificación API - Segunda Entrega*', que contiene todas las especificaciones sobre los nuevos endpoints implementados, también se puede recurrir en caso de necesitarlo a la especificación de la versión anterior de la API.

14.9 Obtener Cohesión Relacional a nivel de namespace

```
from ns in Application.Namespaces
let types = ns.ChildTypes
let N = types.Count()

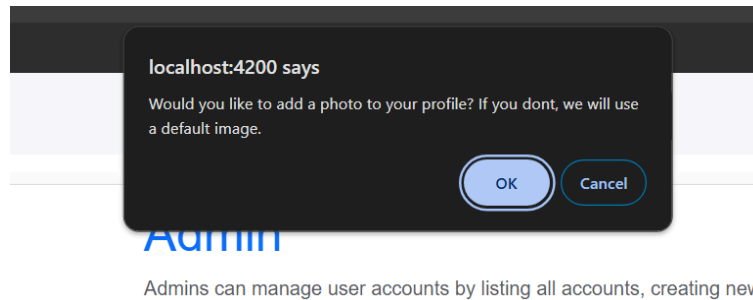
let internalDependencies =
    from t1 in types
    from m in t1.Methods
    from t2 in types
    where t2.IsUsedBy(m) && t1 != t2
    select new { t1, t2 }

let R = internalDependencies.Count()

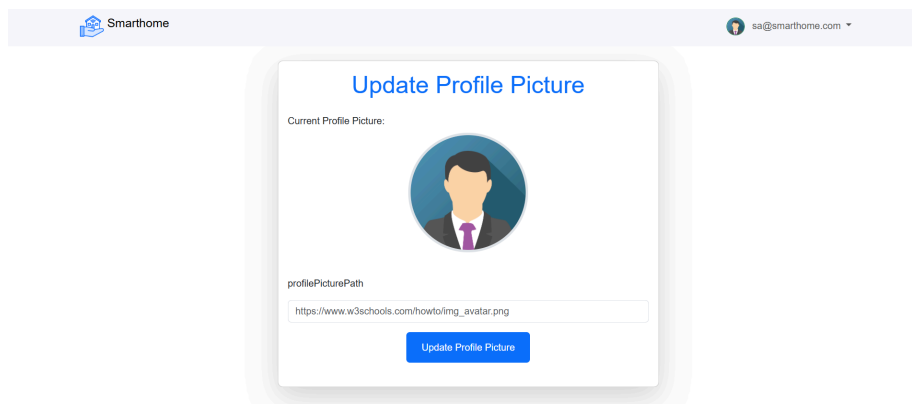
select new
{
    ns,
    Cohesion_Relational = (R + 1.0) / (N == 0 ? 1.0 : N)
}
```

14.10 Agregar foto de perfil al reconvertir cuenta como home owner

Al confirmar la intención de convertirse en Home Owner, se mostrará un mensaje de confirmación solicitando al usuario si desea agregar una imagen de perfil personalizada o utilizar una imagen predeterminada.



En caso de que el usuario decida agregar una imagen personalizada, será redirigido a la página de Actualizar Imagen de Perfil. Esta página permite visualizar la imagen actual del perfil y proporciona un campo para ingresar la URL de la nueva imagen deseada. Además, la imagen del perfil se reflejará dinámicamente en el header de la aplicación. Si el usuario actualiza su imagen en esta página, el cambio se aplicará automáticamente en el header sin necesidad de recargar la interfaz.



Si el usuario opta por no agregar una imagen de perfil en ese momento, se utilizará una imagen predeterminada. Sin embargo, el usuario podrá cambiar su imagen en cualquier momento accediendo a la opción de Cambiar Imagen de Perfil desde el menu desplegable ubicado en el header

