

Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de Aplicaciones 2

Obligatorio 2

Juliette Ruchel - 203942

Francisco Martinez - 233126

Docente: Ignacio Valle

Entregado como requisito de la materia Diseño de
Aplicaciones 2

<https://github.com/ORT-DA2/203942-233126>

25 de junio de 2020

Declaraciones de autoría

Nosotros, Juliette Ruchel y Francisco Martinez, 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 realizábamos 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 contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Resumen

El presente documento tiene el propósito de exponer nuestra justificación del diseño de nuestro sistema para el problema presentado. En esta actualización se presentaran además las métricas actuales del sistema y el impacto de los cambios realizados.

Índice general

1. Descripción del Sistema	3
1.1. Nuevas Funcionalidades	3
1.2. Impacto y Cambios	3
1.3. Aclaraciones	4
1.3.1. Requerimientos previos:	4
1.3.2. Bugs detectados en esta versión	4
2. Diseño	5
2.1. Estructura de Paquetes	5
2.1.1. Dependencias de Paquetes	5
2.1.2. Diagrama de paquetes con nesting	6
2.1.3. Diagrama de Componentes del sistema	8
2.2. Paquete de Dominio	9
2.2.1. Diagrama de Clases	9
2.2.2. Diagramas de Interacción	10
2.3. Paquete de Data Access	12
2.3.1. Diagrama de Clases	12
2.3.2. Diagrama de Entidades	13
2.4. Paquete de Lógica de Negocio	14
2.4.1. Diagrama de Clases	14
2.4.2. Diagramas de Interacción	15
2.5. Paquete Importer	16
2.5.1. Diagrama de Clases	16
2.6. Paquete Web Api	18
2.6.1. Diagrama de Clases	18
2.6.2. Diagramas de Interacción	19
2.7. Deploy de la Aplicacion	19
3. Métricas	21
3.1. Abstracción e Inestabilidad	21
3.2. Cohesión Relacional	22
3.3. Conclusión	23
A. Clean Code	24
A.1. Test Driven Development	24
A.2. Cobertura de pruebas	26

A.3. Clean Code	28
B. Documentación API	31
B.1. Estándares	31
B.2. Mecanismos de Autenticación	31
B.2.1. Authorization Filter	31
B.3. Códigos Utilizados	32
B.4. Especificación de Endpoints	32
B.4.1. Administrator Controller	32
B.4.2. Citizen Request Controller	35
B.4.3. Importer Controller	38
B.4.4. Reports Controller	39
B.4.5. Session Controller	40
B.4.6. Type Controller	41
B.5. Visualizar documentación de la API online	43
C. Tutorial Importador	44
C.1. Aclaraciones Generales	44
C.2. Json Area Importer	45
C.3. Xml Area Importer	46
D. Páginas Front End	47
D.1. Reporte A: Request Summary	47
D.2. Reporte B: Type Summary	48
D.3. Importar XML	48
D.4. Solicitudes	49
D.5. Administradores	50
D.6. Áreas, Temas y Tipos	51
E. Manual Deploy	54
E.1. Deploy Lado Servidor	54
E.2. Deploy Lado Cliente	59
Bibliografía	61

1. Descripción del Sistema

1.1. Nuevas Funcionalidades

1. Se agregó un nuevo tipo de campo adicional "Bool", que puede recibir como valor "True" o "False".
2. Todos los campos adicionales, a excepción del campo *Bool*, se les permite configurar si esta permitido ingresar más de un valor para él mismo en una solicitud.
3. Importador de Áreas. Esta funcionalidad permite ingresar nuevas áreas, sub temas y sub tipos, o ingresar nuevos temas o tipos a áreas existentes. Actualmente, un archivo es requerido para la importación, en el capitulo de justificación sobre el desarrollo de esta funcionalidad encontrará más detalles al respecto. Sección 2.5. También se encuentra un anexo que ejemplifica el uso del importador, para las opciones que vienen incluidas en esta funcionalidad. Apéndice C
4. Nuevos reportes para los administradores

1.2. Impacto y Cambios

Ninguno de los cambios realizados causó un gran impacto sobre el sistema actual, en la gran mayoría fueron nuevos archivos el impacto que se obtuvo.

- Al agregar el nuevo campo adicional, se agregó una nueva clase y se tuvo que modificar el contexto, el binder y agregar un nuevo modelo a la web api para que consideraran el nuevo tipo. La lógica no se vio afectada por el cambio.
- Otro cambio respecto al nuevo campo, fue cambiar algunos métodos abstractos de la clase *BaseField* a virtual, para evitar redefinir métodos que no serían utilizados por la implementación actual, pero permitir que futuras implementaciones si los redefinan.
- Para permitir la múltiple selección de valores, se agregó una Property virtual en la clase *BaseField*, lo que permitió que el nuevo tipo Bool pueda prevenir que se ingrese más de un valor y que los datos ingresados no resulten incoherentes.

- Para la implementación del importador se crearon dos nuevos proyectos: *ImmRequest.Importer* e *ImmRequest.Importer.Interfaces*. La justificación detrás de esto se encuentra en el capítulo de Diseño. Sección 2.5
- Para implementar los reportes, se creó un nuevo controller y se reutilizó la lógica de las solicitudes.
- Se implementó el método *Save* en las clases lógicas para poder hacer uso del patron *Unit Of Work* [1]. De esta forma se realiza una sola vez la operación de guardado de cambios. Esto agregó gran ventaja al importador, al poder cancelar la operación sin dejar algunos datos guardados y otros no.

1.3. Aclaraciones

1.3.1. Requerimientos previos:

Para el correcto funcionamiento del sistema se deberá tener instalado las siguientes tecnologías en la maquina local donde se ejecute el proyecto.

- .NET Core 3.1
- Node.js v12.16.2
- IIS, Internet Information services. Se deberá también tener IIS Scripts and Tools. Refierasé al siguiente articulo para más información sobre como instalar IIS en su maquina [8]
- Al abrir la carpeta del cliente en visual code, en una terminal ejecutar el siguiente comando: *npm install*. Así se instalaran las dependencias necesarias.

1.3.2. Bugs detectados en esta versión

- Una limitación actual del sistema es que el importador no es lo suficientemente dinámico, para aceptar la diversidad de parámetros que las implementaciones puedan presentar. Esto se debe a que la capa de web api espera únicamente un archivo. Fue una decisión basada en el tiempo de desarrollo dado, aunque el equipo cree que su modificación no debería causar mayor impacto en caso de realizarse el cambio en futuras entregas. Más detalle en el capítulo del Importador. Sección 2.5
- Un bug conocido es el poder agregar campos adicionales repetidos en la creación de las solicitudes. Esto implica que por más que se controle que se ingrese un único valor, si la múltiple selección no está permitida, se pueda mandar dos campos con un único valor.
- Otro error encontrado por el equipo se ubica en el front end de Importación. Si se selecciona un archivo, se envía, luego se edita el mismo, y se vuelve a enviar. El navegador tira el siguiente código de error: *ERR UPLOAD FILE CHANGED*

2. Diseño

2.1. Estructura de Paquetes

2.1.1. Dependencias de Paquetes

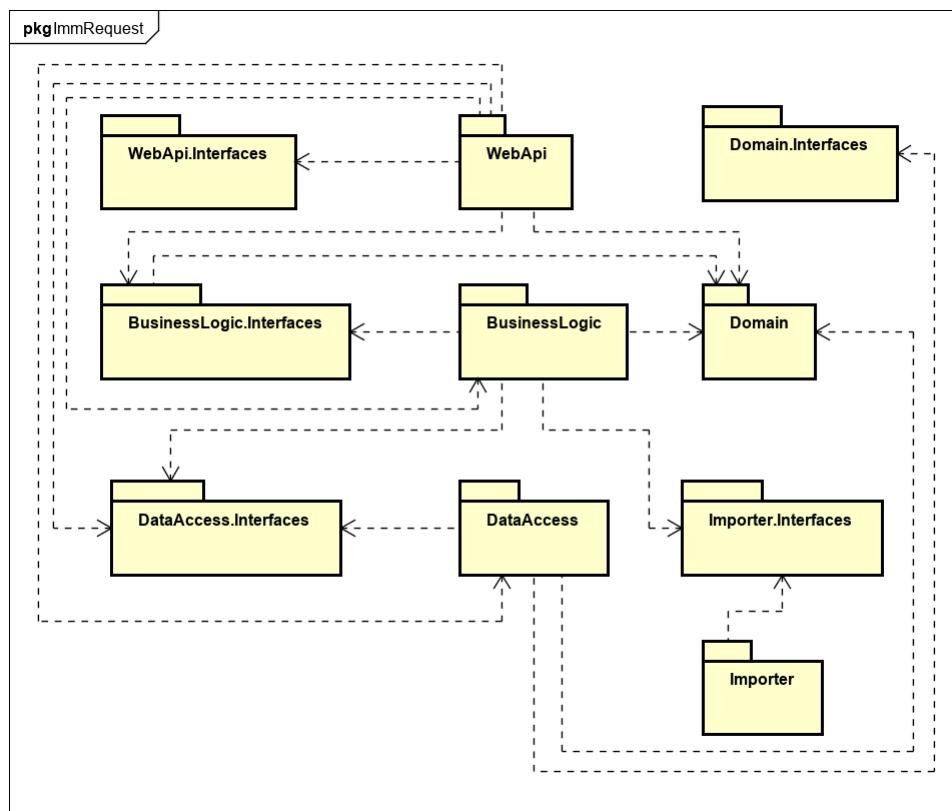


Figura 2.1: Diagrama de dependencias entre paquetes

Nuestra idea principal para el diseño de los paquetes fue generar niveles y conectar cada uno con interfaces intermedias con el objetivo de que aquellos cuya implementación sea mayoritariamente concreta dependan de los más abstractos y por ende más estables. Los paquetes de *DataAccess.Interfaces* y *BusinessLogic.Interfaces* buscan lograr justamente eso.

La otra razón detrás de los paquetes de *.Interfaces* es poder asignar la responsabilidad de establecer los contratos o pre-requisitos que las clases concretas deben

implementar en un lugar específico y poder reusarlos para otros proyectos con requerimientos similares. *Domain.Interfaces* y *WebApi.Interfaces* son ejemplo de esto.

Descripción de cada paquete

- **ImmRequest.Domain:** Es el paquete que contiene las entidades de la solución que proporciona el sistema.
- **ImmRequest.Domain.Interfaces:** Define los contratos que deben proporcionar las entidades del sistema.
- **ImmRequest.DataAccess:** Es el paquete que tiene la responsabilidad de comunicarse con la base de datos. Se encarga de manejar el contexto y realizar las consultas necesarias para la lógica.
- **ImmRequest.DataAccess.Interfaces:** Contiene las interfaces que deben implementar las clases que se comuniquen con la base de datos. Conecta el paquete de la lógica con el de base de datos, para reducir el impacto de cambio.
- **ImmRequest.BusinessLogic:** Contiene la lógica de negocio para poder cumplir con los requerimientos funcionales de la aplicación.
- **ImmRequest.BusinessLogic.Interfaces:** Establece las interfaces necesarias para las clases de lógica. Conecta el paquete de WebApi y el paquete de Lógica.
- **ImmRequest.WebApi:** Es el paquete ejecutable del proyecto, en donde se encuentran los diferentes endpoints del sistema.
- **ImmRequest.WebApi.Interfaces:** Contiene las interfaces necesarias para la implementación de algunas clases del paquete WebApi
- **ImmRequest.Importer:** Es el paquete que se encarga de contener los importadores desarrollados por el equipo.
- **ImmRequest.Importer.Interfaces:** Es el paquete que contiene las interfaces para que terceros puedan implementar sus propios importadores.

2.1.2. Diagrama de paquetes con nesting

En el siguiente diagrama se puede observar cada paquete principal con sus namespaces. El objetivo es motivar el reuso y asignarle una responsabilidad común a las clases de cada uno.

Al igual que la entrega pasada se mantuvo la decisión de ubicar las excepciones del paquete de DataAccess en el de Interfaces. Los mismo se aplicó para el denominado Importer. La razón de estas decisiones es evitar dependencias directas entre paquetes concretos y que las dependencias existentes sean de paquetes abstractos, de mayor estabilidad.

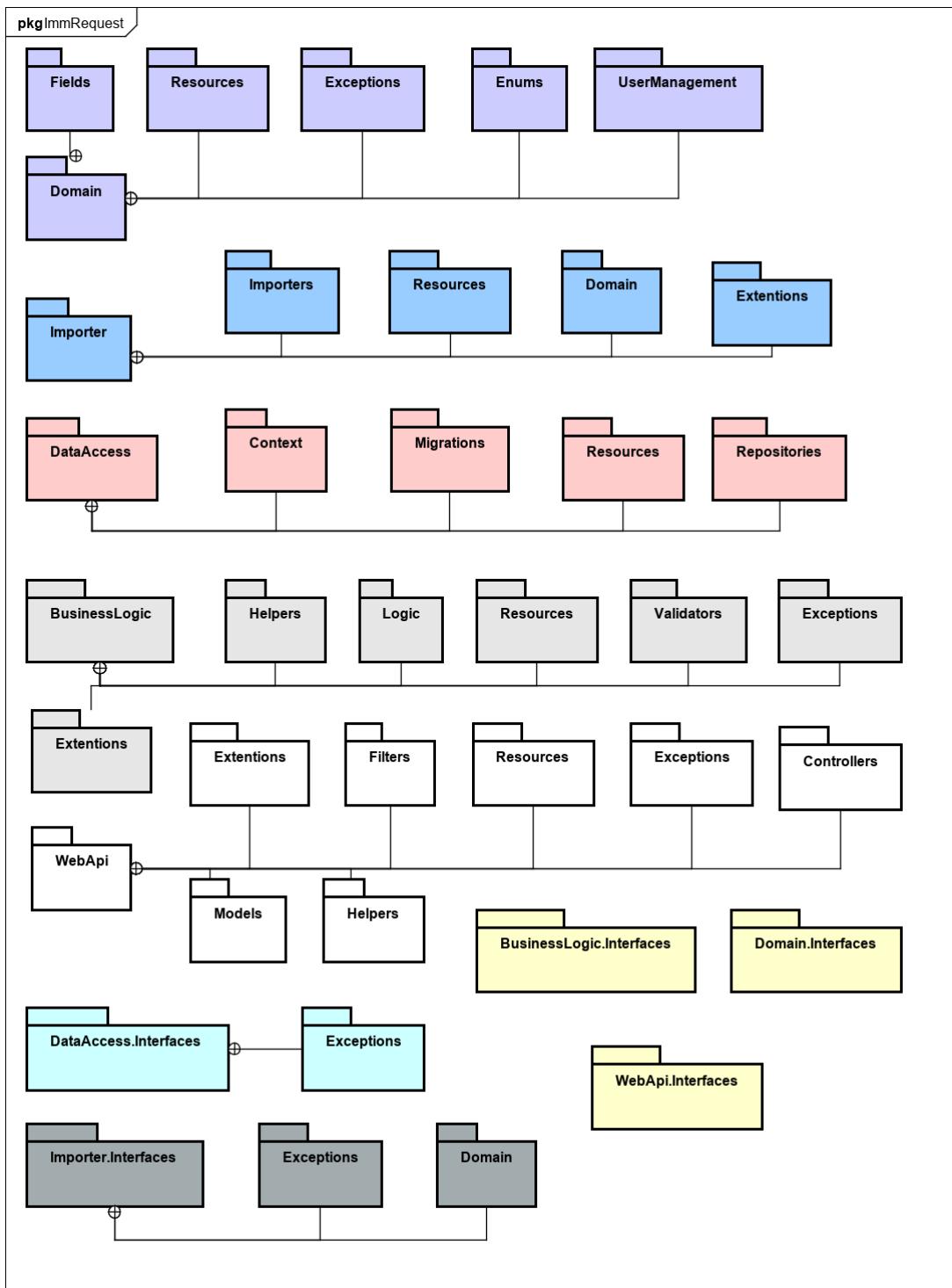


Figura 2.2: Diagrama de paquetes usando conector nesting

2.1.3. Diagrama de Componentes del sistema

En el siguiente diagrama de componentes podemos observar como los distintos componentes de cada paquete se comunican a través de las interfaces establecidas en cada uno de los paquetes de *.Interfaces*.

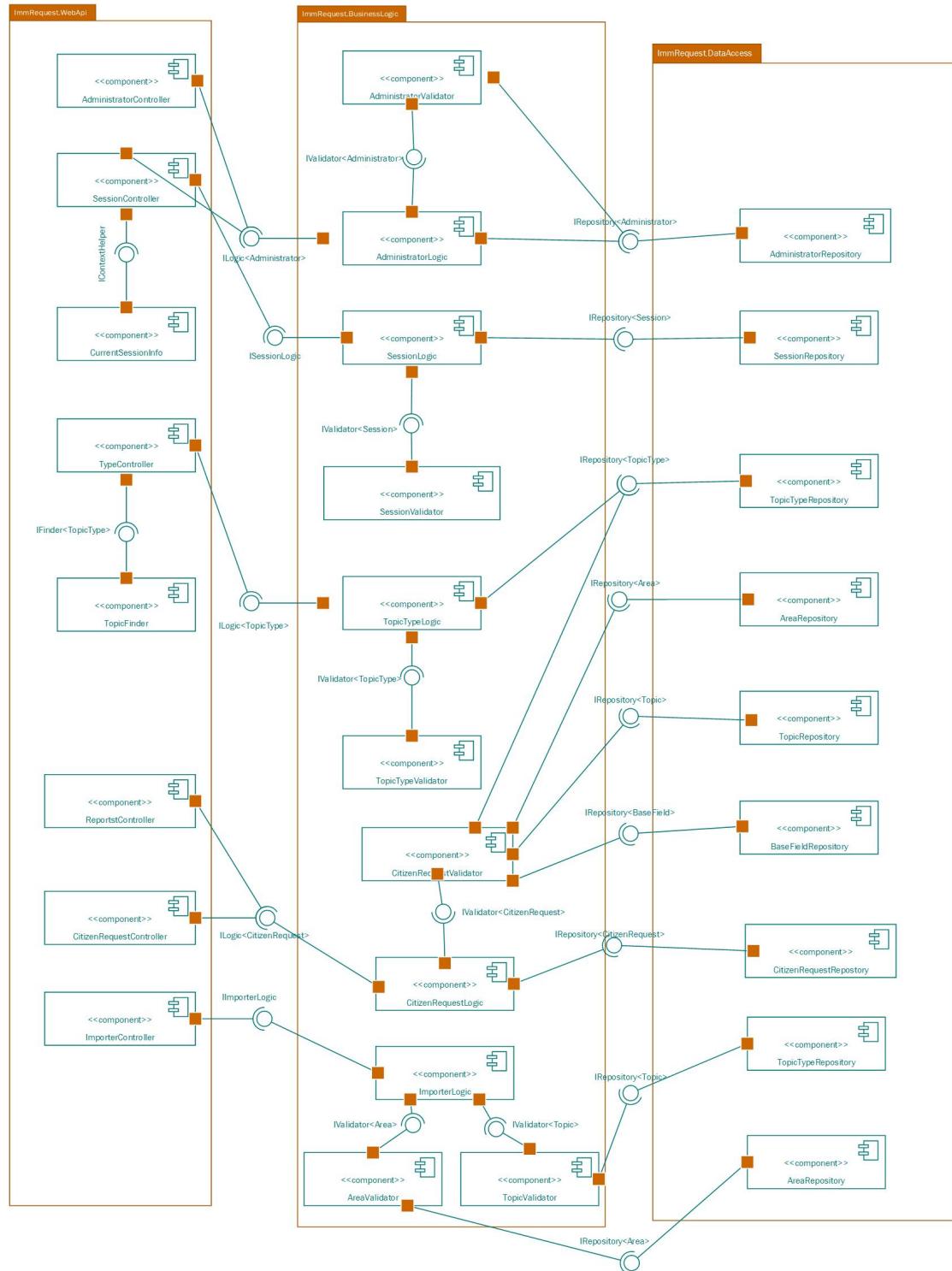


Figura 2.3: Diagrama de componentes del sistema

2.2. Paquete de Dominio

2.2.1. Diagrama de Clases

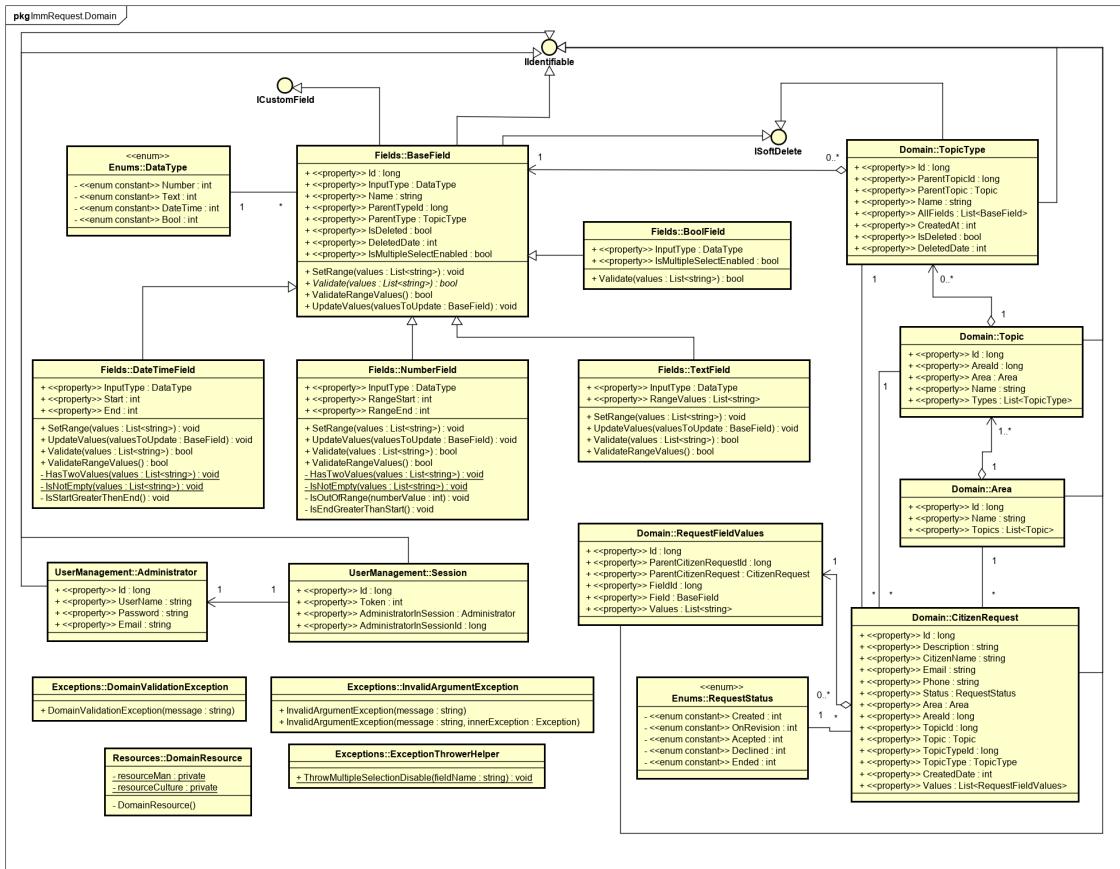


Figura 2.4: Diagrama de clases de domino

La decisiones más importantes respecto al diseño de nuestra solución fueron tres. La primera, es que toda entidad que se fuera a guardar en la base de datos tenía que implementar la interfaz de *Identifiable*. Como forma de establecer un estándar para el ID de las entidades de la base de datos y asegurarse que toda entidad la cumpla.

La segunda, es la utilización de la interfaz *SoftDelete*. Aquellas entidades que no se deban eliminar permanentemente deberán implementar esta interfaz, la cual luego el contexto utilizará para determinar el modo de eliminación, pero además para llevar un control de la fecha de la misma. En caso de algún error, la entidad que se eliminó se puede encontrar por su fecha siempre y cuando se sepa la misma.

Por ultimo, la utilización de polimorfismo para la implementación de los campos adicionales. Usando un patron strategy, en donde una interfaz *ICustomField* define el comportamiento que cualquier tipo de campo adicional debe implementar, nos permite dar la extensibilidad de agregar futuros campos adicionales cuya implementación del comportamiento difiera de la de *BaseField*. Al mismo tiempo se creó la

clase *BaseField* para poder implementar el comportamiento común que tienen los campos adicionales actuales y utilizar el polimorfismo para otorgarle la responsabilidad a cada campo la asignación de su rango y validación del valor que se da en la solicitud. Esto permite que no se tengan que crear clases especiales que según el tipo campo validen o asignen el valor del mismo en la solicitud.

2.2.2. Diagramas de Interacción

A continuación se pueden encontrar diagramas de secuencia y colaboración que buscan exemplificar el uso del polimorfismo para la creación y validación de los campos adicionales.

En ambos casos se esta creando un *Topic Type* que es quien contiene una lista de *BaseFields*. Por cada elemento en esa lista se va a llamar a los métodos de cada implementación de una *BaseField* para validar y definir su rango.

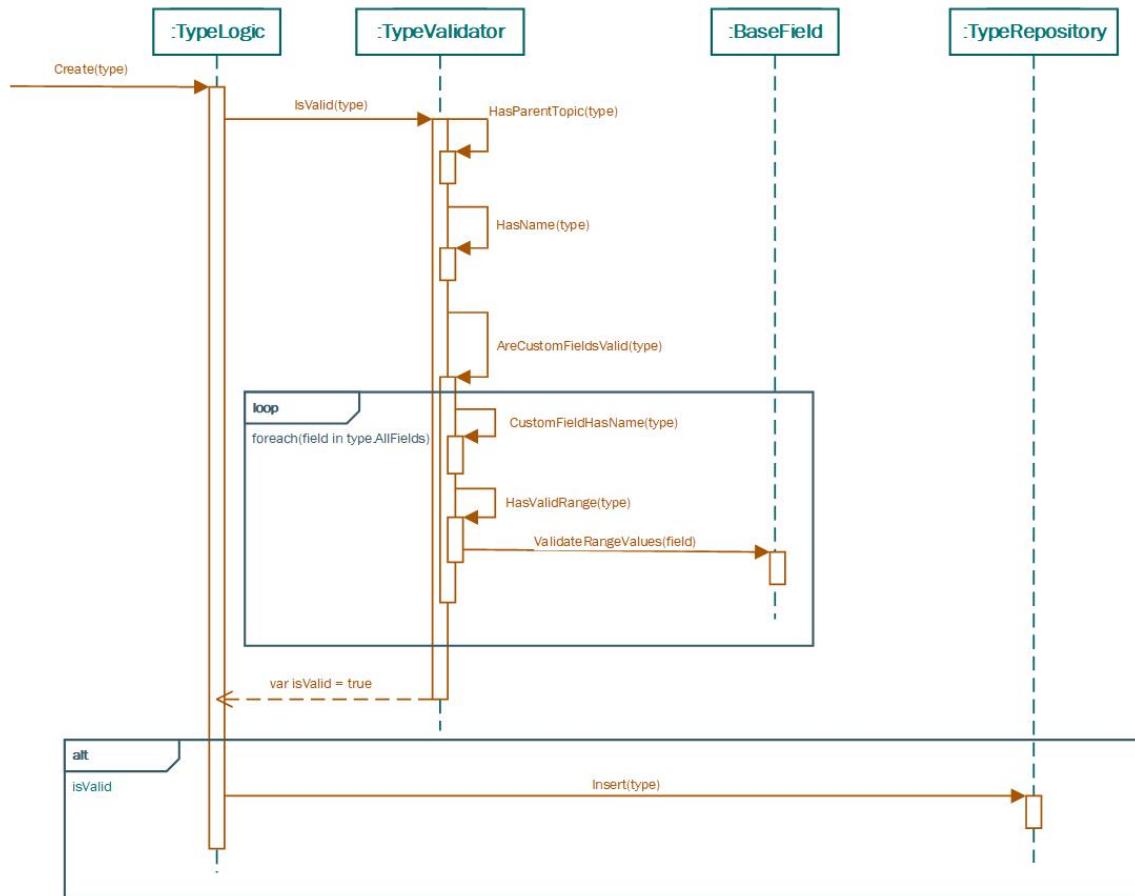


Figura 2.5: Crear Topic Type desde la lógica

En el siguiente diagrama se puede observar un caso concreto, asumiendo que el la lista de campos adicionales hay tres campos de distinto tipo y que la variable `field` cambia en cada iteración del loop.

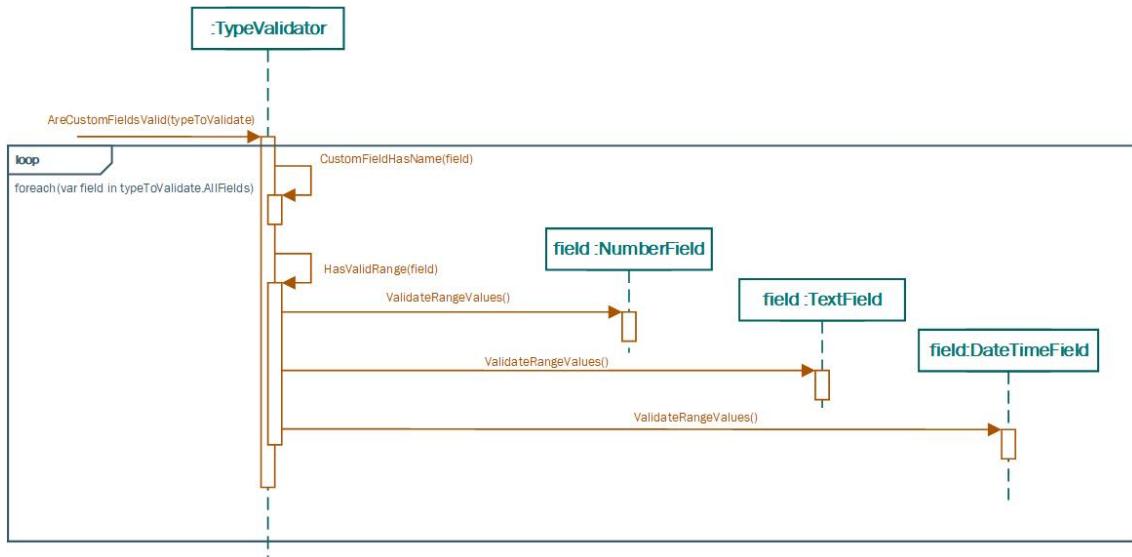


Figura 2.6: Validar rango de campos adicionales caso concreto

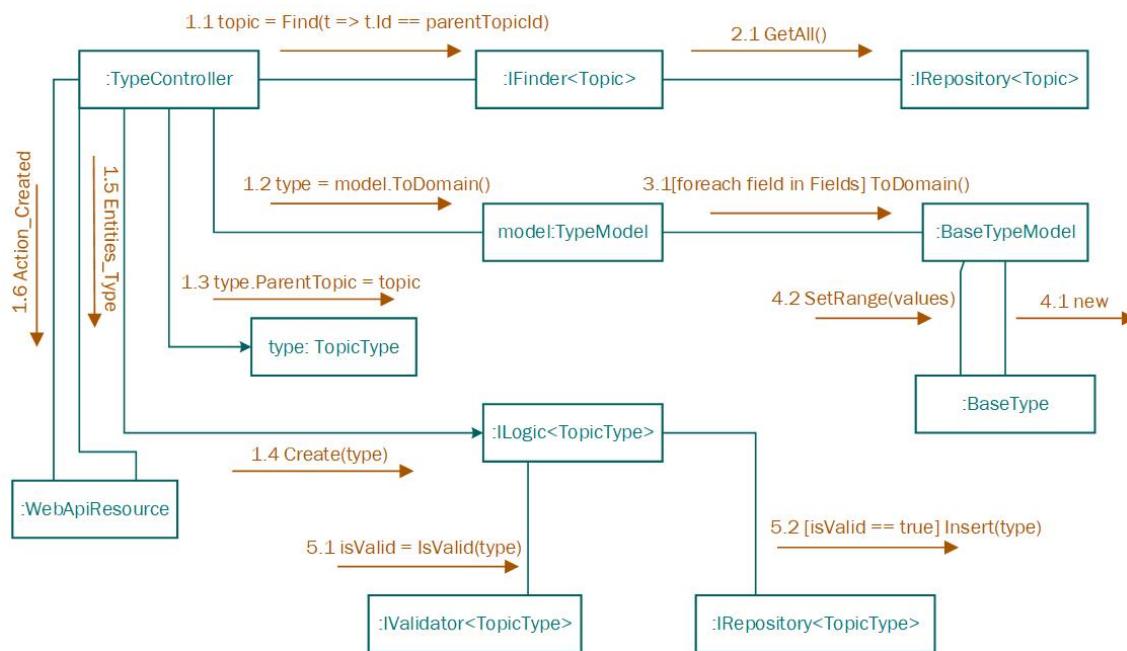


Figura 2.7: Crear un topic type

2.3. Paquete de Data Access

2.3.1. Diagrama de Clases

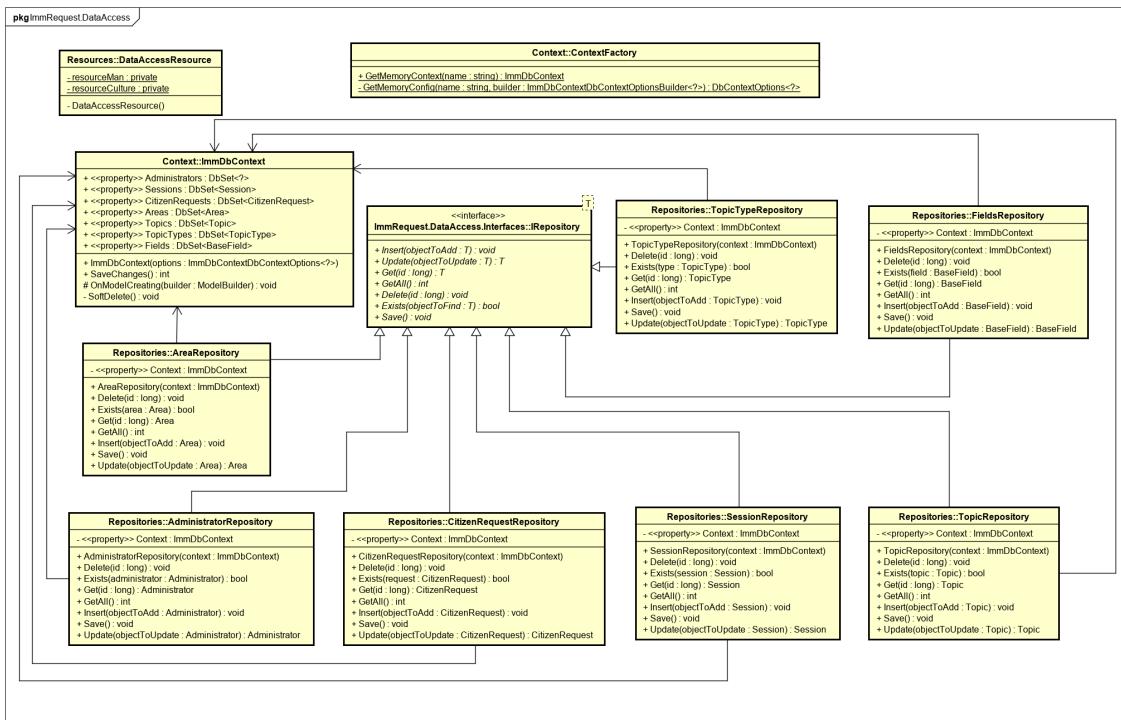


Figura 2.8: Diagrama de clases del paquete Data Access

Para el data access se determinó una interfaz que establece las funciones básicas para cualquier repositorio. De esta forma cada repositorio implementa sus comportamiento específico, lo que nos permite incluir las entidades de navegación requeridas para cada entidad.

Un detalle a destacar, se implementó un *Query Filter* para las entidades que proveen la interfaz *SoftDelete* que mencionamos en la sección anterior Subsección 2.2.1. La razón de esto, es para prevenir que cuando el repositorio retorne las entidades de la base, no retorne aquellos que fueron marcados como eliminados. El único caso en el que se ignora este filtro es en el *Get* y *GetAll* de la `CitizenRequest` y en el repositorio de `BaseFields`, por la forma en que Entity Framework resuelve los includes. De no ignorar este filtro, no traería aquellas solicitudes que cuyos tipos o campos adicionales fueron eliminados por algún administrador.

Otro dato interesante es que los datos de las Áreas y Temas, al igual que un administrador, son creados con migrations a la base. Esas migrations no modifican en ninguna forma el diseño del sistema, sino que aprovechan el Migration builder para ejecutar sentencias SQL e ingresar los datos necesarios para la correcta ejecución.

Nota importante: *Dichas migrations tienen como pre requisito que no hayan datos de prueba ingresados previamente a la base. Ya que la migration que ingresa los tipos depende de los IDs de las Areas creadas en la migration anterior. Si la base*

esta siendo creada por primera vez no va ver problema

2.3.2. Diagrama de Entidades

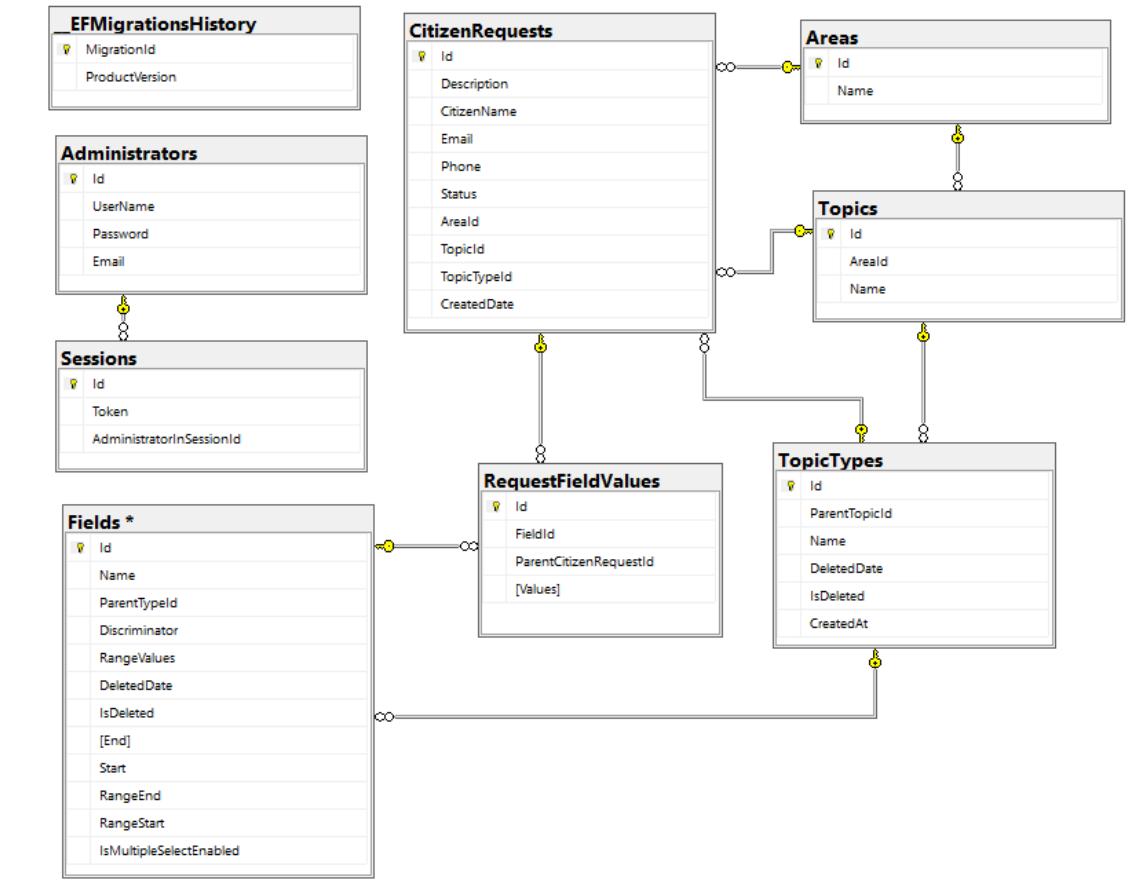


Figura 2.9: Tablas en la base de datos

Agregamos a este diagrama la tabla donde se guarda el registro de migrations aplicadas a la base.

2.4. Paquete de Lógica de Negocio

2.4.1. Diagrama de Clases

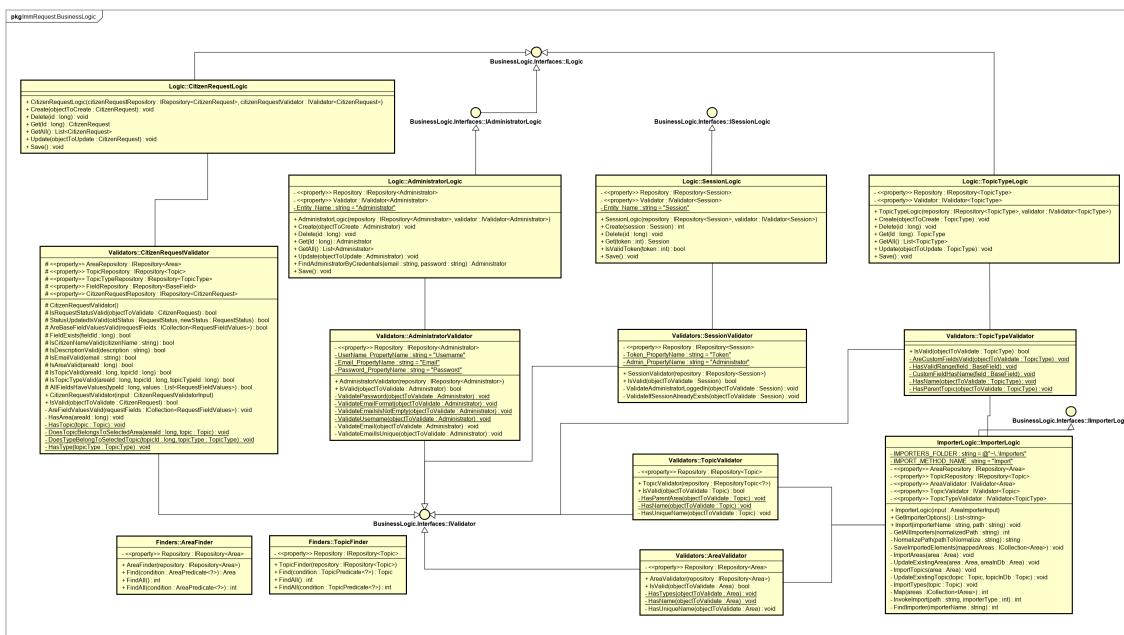


Figura 2.10: Diagrama de Clases de la lógica

Nuestra solución de paquete de lógica se divide en dos grandes pedazos. Las clases que implementan *ILogic*, cuya responsabilidad es manejar la lógica de negocio relacionada a alguna de las entidades, y las que implementan *IValidator*, cuya responsabilidad es validar las entidades que forman parte de las operaciones lógicas. Se tomó esta decisión para poder repartir la responsabilidad de realizar las operaciones del negocio de validar que cumplan las condiciones necesarias para guardar los datos a la base. Nos da la flexibilidad de poder crear nuevas clases de lógica, que no requieran de validadores y que éstos puedan ser usados en otras partes del sistema si se requiere. Estas clases fueron separadas en paquetes distintos para que sean reusables, en caso necesario. Para tanto la lógica como los validadores se usaron patrones strategy para tener la flexibilidad de crear distintas implementaciones de cada uno que cumplan con el comportamiento y sean intercambiables si existiera la necesidad.

La interfaz *IFinder* define un comportamiento de búsqueda para un tipo genérico. Su objetivo, es poder implementar una lógica reducida para entidades que no requieran operaciones crud, como la que propone *ILogic*. Su objetivo es evitar que se implementen funciones innecesarias que solo generarían ruido en el código.

2.4.2. Diagramas de Interacción

Un nuevo componente de este paquete es la clase *ImporterLogic*. Es la encargada de importar las áreas según el importador requerido por el usuario. La interfaz que implementa: *IImporterLogic*, nos permite crear nuevas clases que se dediquen a importar distintos objetos del sistema, ya que define el comportamiento común que se debe implementar.

A continuación se pueden como funciona un validador, en el caso de una *CitizenRequest*, y como colaboran con las clases de lógica, en el caso de la creación de un *Administrador*.

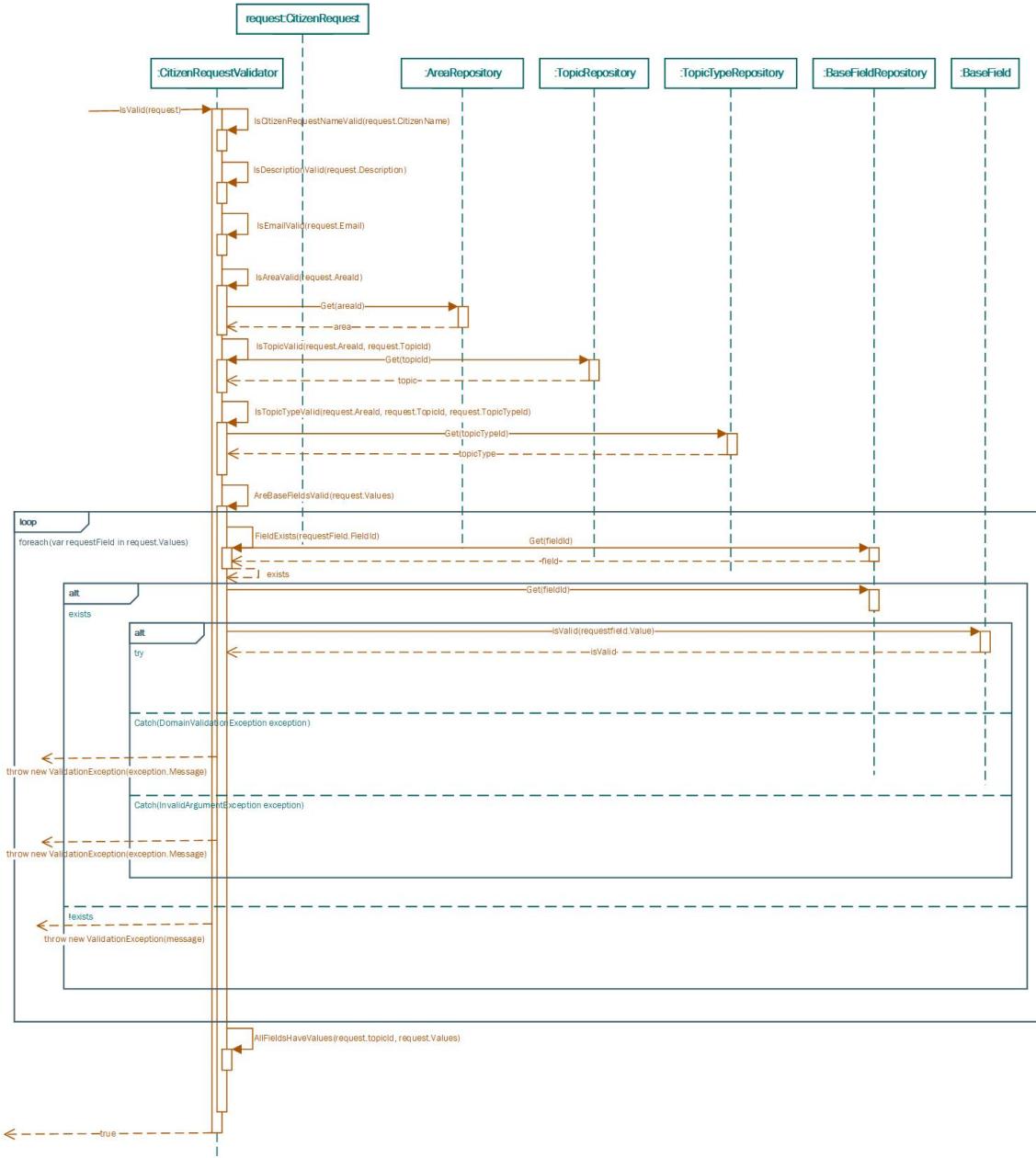


Figura 2.11: Validación de una citizen request

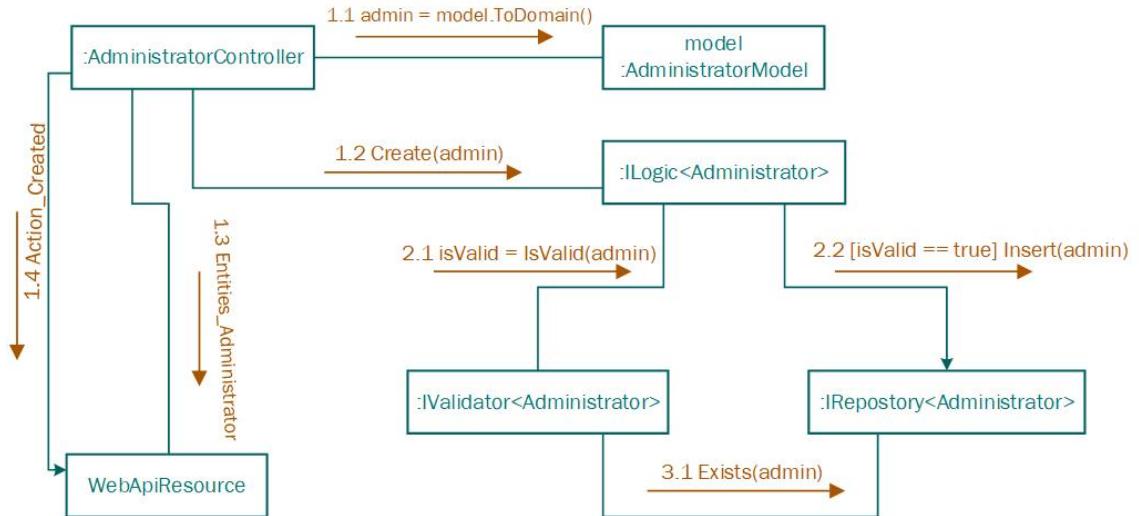


Figura 2.12: Creación de un administrador

2.5. Paquete Importer

2.5.1. Diagrama de Clases

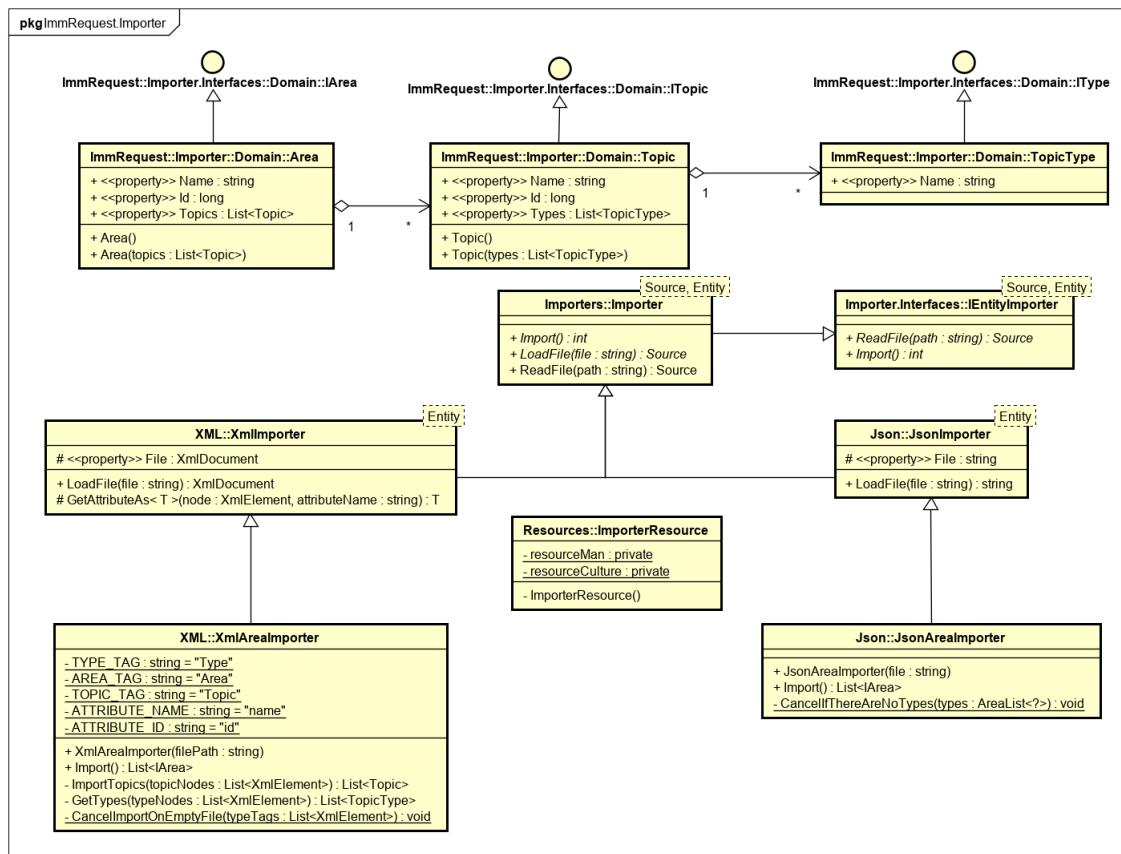


Figura 2.13: Creación de un administrador

Este paquete se divide principalmente en dos partes, el de dominio y el de los importadores. El primero implementa las interfaces ofrecidas por el paquete *.Interfaces* creando objetos similares a los originales de la solución. La idea detrás de este dominio en interfaz es abstraer la información importante sobre nuestros objetos de modo de ofrecer al desarrollador de importadores el comportamiento básico que espera nuestro sistema. Adicionalmente, nos permite esconder nuestra implementación real dando a conocer lo mínimo indispensable.

El segundo, es nuestra implementación de los importadores con un enfoque en reuso. La interfaz *"IEntityImporter<Source, Entity>"* contiene dos parámetros genéricos, que son el resultado de los métodos que define la misma. El método *"ReadFile"* tiene como objetivo levantar la fuente de los datos y retornar un *"Source"* que es el tipo de objeto que necesita este importador. Consecuentemente, cada importador puede definir como levantar los datos necesarios y como retornarlos para su utilización.

El método *"Import"* retorna un *"Entity"* el cual simboliza los datos que se quieren importar. Esto implica que se pueden definir importadores para cualquiera de los objetos del dominio anterior, o cualquier otro objeto si se utiliza en otro proyecto.

Una limitación de esta implementación fue no definir un método para los requerimientos de cada importador en el frontend. Actualmente el endpoint de importación requiere de un archivo, el cual los importadores pueden optar por ignorar pero no es ideal como solución de diseño. Un cambio a futuro, sería que cada importador implemente un método que retorne los parámetros requeridos para que el frontend pueda renderizarlos y el endpoint espere algún objeto dinámico en lugar de un archivo.

En el caso de nuestros importadores, aplicamos dos patrones en conjunto: el patrón *Strategy* y *Template Method*. El patrón *strategy* lo define la interfaz, debido a que impone el comportamiento que todo importador debe tener. El *template method*, se usa varias veces y el objetivo fue evitar repetir el mismo código para cada importador. La clase padre *Importer* implementa el método definido por la interfaz y agrega el manejo de errores común, pero establece un método hook para que sus hijos implementen su propio comportamiento. Además, no define el *Source* ni el *Entity*. Las clases *XmlImporter* y *JsonImporter* mantienen la misma intención pero definen el retorno *Source* para sus implementaciones hijas y agregan el manejo de errores propio de ellos. La idea detrás de estas clases es que futuras implementaciones de importadores que también utilicen json o xml, puedan heredar de estas y hacer uso de los métodos ofrecidos por ellas. Finalmente, se encuentran los importadores concretos que definen su entidad de retorno como *Área* y implementan los métodos de importación.

2.6. Paquete Web Api

2.6.1. Diagrama de Clases

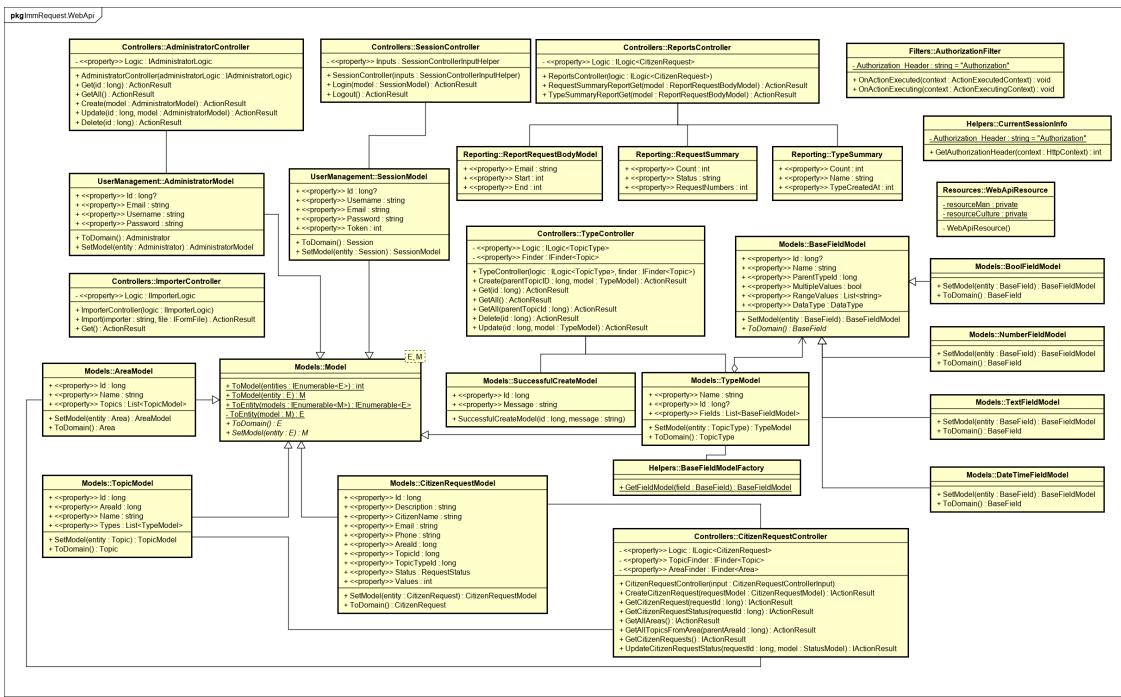


Figura 2.14: Diagrama de Clases de la Web Api

Para poder cumplir con los requerimientos necesarios se hicieron las seis controllers: *AdministratorController*, para el mantenimiento de Administradores, *CitizenController*, para la creación y mantenimiento de solicitudes, *TypeController*, para el mantenimiento de tipos y sus campos adicionales, *SessionController* para el control de sesión de un administrador, *ReportsController*, para el acceso a los reportes y *Importer*, para el manejo de las importaciones. Se buscó darle a cada controller la responsabilidad sobre alguna entidad y sus operaciones crud, de forma que fuera fácil ubicar que controller realiza que operación.

Los controllers reciben los datos en formato json, los cuales un binder los transforma en nuestras clases de modelos, que tienen la responsabilidad de convertir los datos recibidos a clases de dominio o viceversa.

Para el caso especial de los campos adicionales, aplicamos el patrón *Factory Method*. La clase *BaseFieldModel* define el método abstracto para crear los distintos campos a su versión de dominio. (Método *ToDomain()*). Luego cada *BaseFieldModel* crea su propia instancia de *field*. *TypeModel* utiliza este método para crear su lista de campos en dominio, sin necesidad de conocer que tipos de campos posee.

Sin embargo, en el caso de la creación de *Fields* a modelos tuvimos que utilizar una clase helper que nos permitiera crear la instancia del modelo según el *Data Type* de cada *field*, esto se debió a que la clase *BaseField* es abstracta. De igual forma el

modelo de *TypeModel* desconoce como se crean tanto los modelos como los objetos del dominio de los campos adicionales y utiliza la fábrica.

2.6.2. Diagramas de Interacción

El siguiente diagrama ejemplifica la creación de los modelos *TypeModel* y *BaseFieldModel*. Aunque en este ejemplo no se usa clases concretas para simplificar el diagrama y reflejar la interacción entre los modelos.

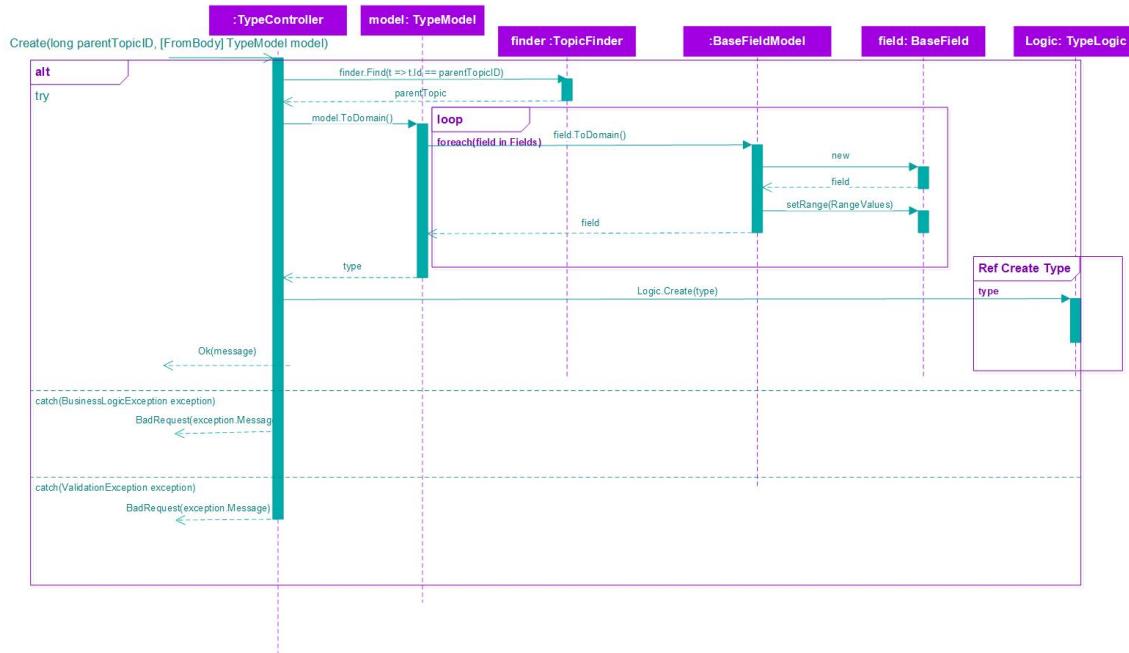


Figura 2.15: Crear un Type desde el Controller

2.7. Deploy de la Aplicacion

El deploy de la aplicación cuenta con tres nodos. El primer nodo contiene la base de datos de SQL Server, que se utilizará para persistir los datos. El segundo nodo contiene el servidor en que se encontrara el API desarrollada. El protocolo de comunicación para estos dos nodos es TCP/IP.

El tercer nodo es el servidor cliente, el cual se presenta en forma de SPA y expone la interfaz de usuario para el uso de los servicios ofrecidos por la API. La comunicación entre estos se da a través de HTTP v 1.0.

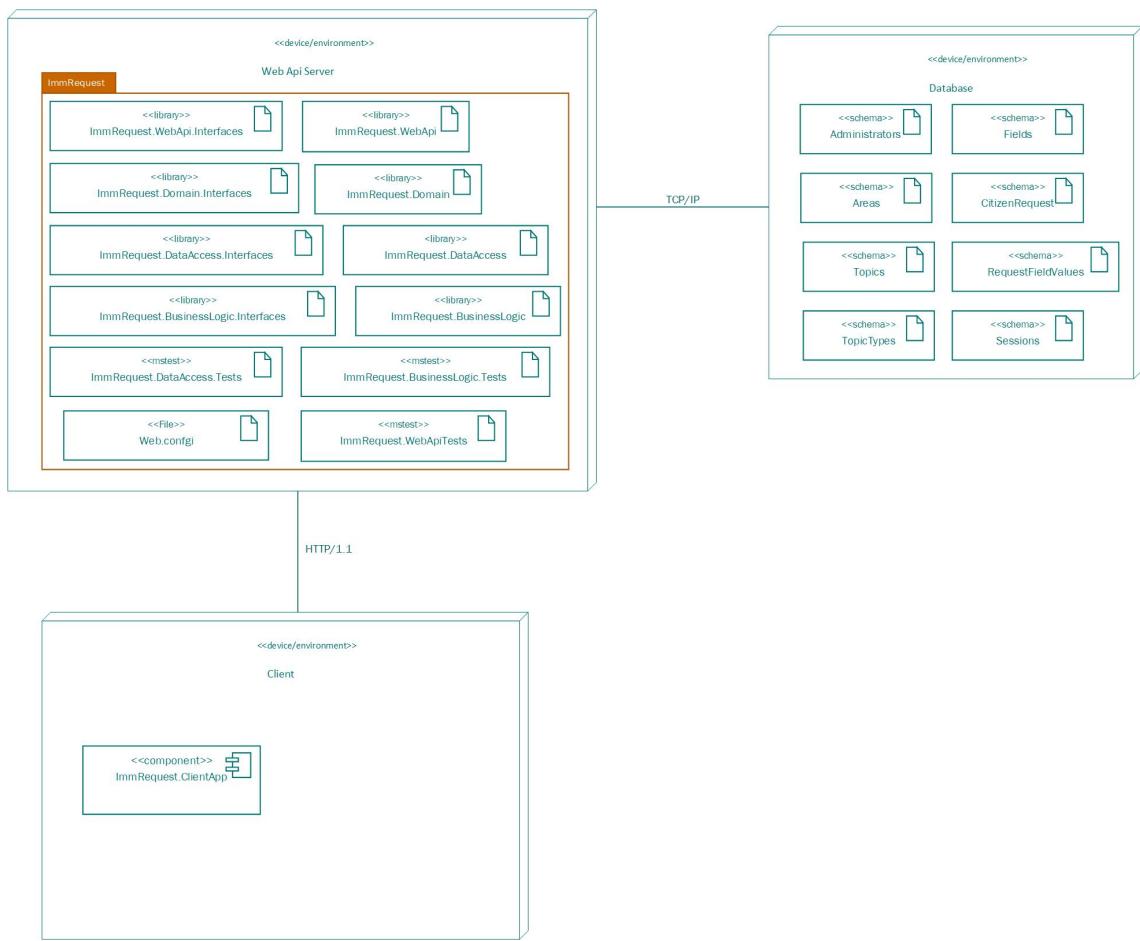


Figura 2.16: Diagrama de Deploy

3. Métricas

Los siguientes resultados fueron obtenidos a través de la herramienta Ndepend [2]. En la carpeta de documentación del repositorio podrán encontrar una carpeta con el nombre: *Ndepend Results*. Al abrir el html de esa carpeta se podrán observar todos los resultados obtenidos por el análisis. En este capítulo resumiremos los resultados y propondremos nuestra conclusión al respecto de estos.

3.1. Abstracción e Inestabilidad

Como se podrá observar en las siguientes figuras, los paquetes de mayor abstracción y estabilidad son aquellos que contienen las interfaces que nuestros componentes utilizan para comunicarse. El paquete de dominio se encuentra en una situación bastante crucial, al ser uno de los paquetes más estables respecto a dependencias, pero extremadamente concreto. Consecuentemente, un cambio en este paquete impacta directamente en el resto.

Por otro lado, recordando el diagrama de dependencias de paquetes Subsección 2.1.1, aquellos de mayor inestabilidad y menor abstracción dependen de lo más estables, lo que nos permite reducir el impacto de cambio. Esto es debido a que los paquetes de interfaces esconden la implementación de los concretos, provocando que sus empleadores no se percaten del cambio detrás.

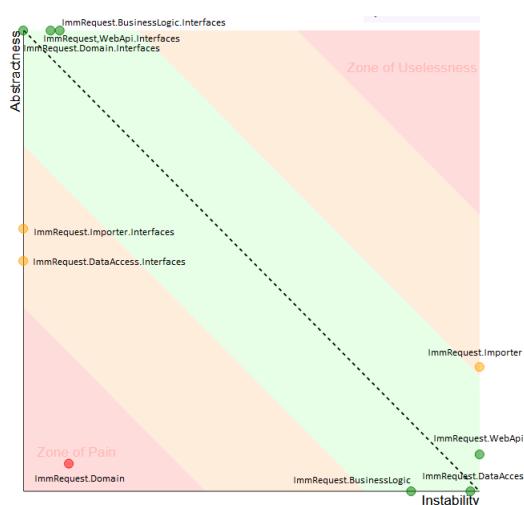


Figura 3.1: Gráfica de abstracción e inestabilidad por Ndepend

Paquete	Abstracción
ImmRequest.Domain.Interfaces	1
ImmRequest.Domain	0.06
ImmRequest.DataAccess.Interfaces	0.05
ImmRequest.DataAccess	0
ImmRequest.BusinessLogic.Interfaces	1
ImmRequest.BusinessLogic	0
ImmRequest.Importer.Interfaces	0.57
ImmRequest.Importer	0.27
ImmRequest.WebApi.Interfaces	1
ImmRequest.WebApi	0.08

Tabla 3.1: Abstracción de paquetes

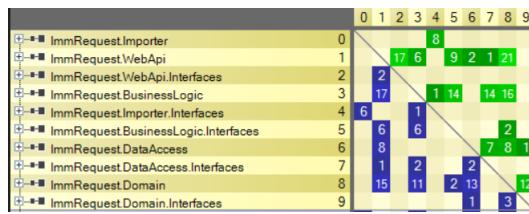


Figura 3.2: Matriz de dependencias de Ndepend

Paquete	Acomplamiento Aferente	Acomplamiento Entrante	Inestabilidad
ImmRequest.Domain.Interfaces	13	0	0
ImmRequest.Domain	47	5	0.1
ImmRequest.DataAccess.Interfaces	22	0	0
ImmRequest.DataAccess	2	79	0.98
ImmRequest.BusinessLogic.Interfaces	23	2	0.08
ImmRequest.BusinessLogic	6	33	0.85
ImmRequest.Importer.Interfaces	9	0	0
ImmRequest.Importer	0	11	1
ImmRequest.WebApi.Interfaces	17	1	0.06
ImmRequest.WebApi	0	184	1

Tabla 3.2: Inestabilidad de paquetes

3.2. Cohesión Relacional

La cohesión relacional expone que tan relacionadas se encuentran las clases dentro de un paquete. Otra forma de visualizarlo es: si las responsabilidades se encuentran bien ubicadas. Observando la siguiente tabla, notamos que aquellos con menor cohesión son los paquetes de interfaces. Esto es a que funcionan como contenedores

para las interfaces que comunican a componentes entre si. De los paquetes concretos, únicamente el Data Access se encuentra por debajo la cota inferior ideal (uno). Creemos que se debe a que los repositorios no se relacionan entre sí, sino que se responsabilizan de manejar el ingreso y edición de su objeto asignado a la base de datos. Ningún otro paquete indica mal repartición de responsabilidades.

Paquete	Cohesión Relacional
ImmRequest.Domain.Interfaces	0.33
ImmRequest.Domain	2.28
ImmRequest.DataAccess.Interfaces	0.5
ImmRequest.DataAccess	0.72
ImmRequest.BusinessLogic.Interfaces	0.33
ImmRequest.BusinessLogic	1.57
ImmRequest.Importer.Interfaces	0.43
ImmRequest.Importer	1.91
ImmRequest.WebApi.Interfaces	0.5
ImmRequest.WebApi	1.78

Tabla 3.3: Cohesión Relacional de cada paquete

3.3. Conclusión

Al analizar estos resultados, observamos que nuestra solución cumple con los principios de acoplamiento. Nuestros paquetes más inestables dependen de aquellos con mayor abstracción y estabilidad. Tampoco presentamos dependencias cíclicas. [Figuras, 3.1, 3.2, 3.2]

En cuanto a los principios de cohesión, el que se observa más afectado es el de *Reuso Común*, ya que si algún assembly de interfaces o el paquete de Data Access fuera de reusarse, es posible que no se utilicen todas las clases. Esto es debido a las pocas relaciones que tienen éstas entre sí. Creemos, sin embargo, que nuestra solución limita que los cambios en un paquete impacten solo a éste.

Asimismo, confiamos en nuestra solución de diseño debido a que las nuevas funcionalidades provocaron cambios en el único paquete que se encuentra en la "Zone Of Pain" pero no causaron mayor impacto en el resto de nuestro sistema. Nuestros paquetes menos estables dependen de abstracciones que esconden las implementaciones reales y nos ayuda a reducir el impacto de cambio. El siguiente enfoque para nuestro sistema debería ser poder mejorar la cohesión para así fomentar el reuso de los paquetes contenidos, alcanzando cumplir con los principios de cohesión.

A. Clean Code

A.1. Test Driven Development

Al igual que la entrega anterior, se utilizó *Test Driven Development* como metodología de desarrollo. Esto implica desarrollar pruebas unitarias previamente al desarrollo del código funcional, para obtener un sistema de mayor confiabilidad y robustez. Si bien TDD no garantiza que no existan bugs, ayuda a reducir su existencia y verificar que las funcionalidades requeridas sean cumplidas por el código desarrollado. [3]

Nuevamente, nuestro desarrollo se realizó de forma *Inside-Out*” lo que requiere que se comienza por las capas de menor nivel hasta alcanzar las de mayor nivel. En esta entrega, se arranco por funcionalidades que impactaran los niveles más bajos, como la base de datos. Por ejemplo, para los reportes se agregaron campos de fechas de creación a los tipos y solicitudes.

Continuación ejemplificación de la metodología con commits. Vale aclarar, que los mensajes de los commits no siempre fueron los más descriptivos, ya que muchas veces el equipo se limito a usar únicamente *[RED]* o *[GREEN]* para mostrar su uso.

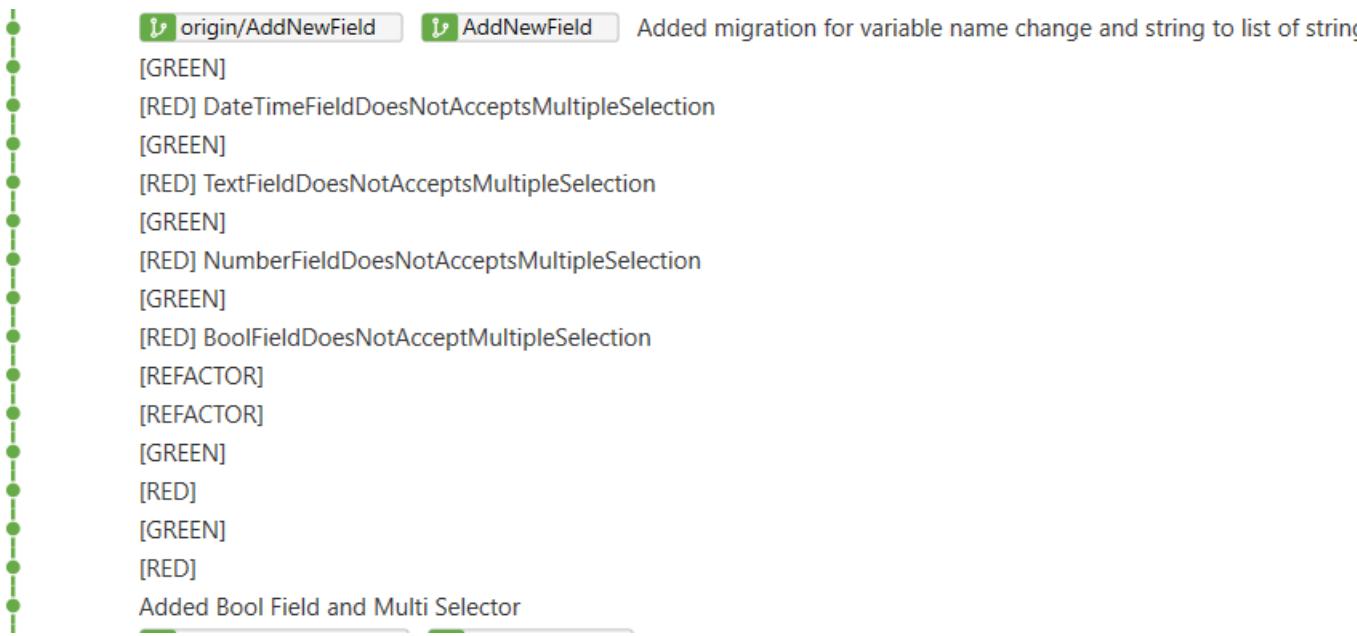


Figura A.1: Agregando el nuevo campo

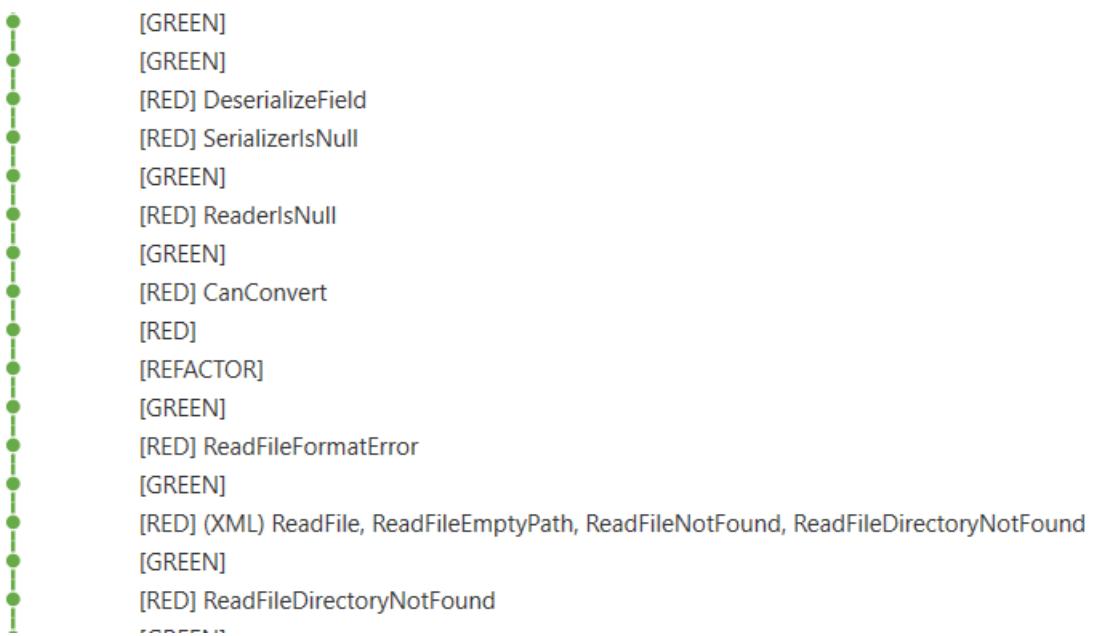


Figura A.2: Desarrollo del paquete Importer

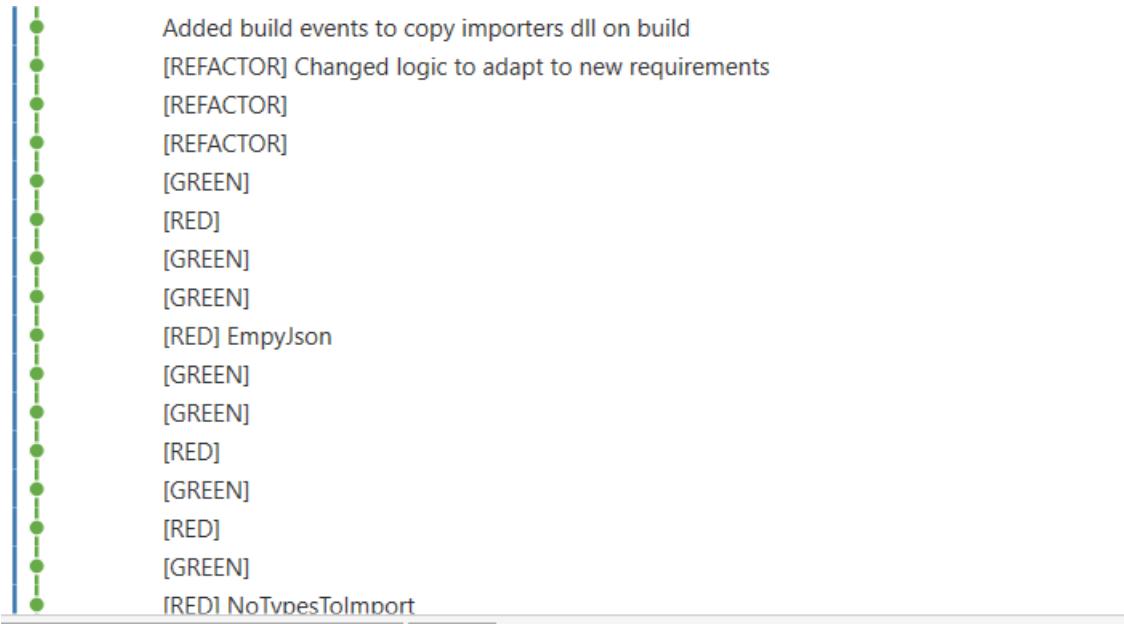


Figura A.3: Modificando la Business Logic para usar los importers

A.2. Cobertura de pruebas

Para obtener la cobertura del código se utilizó la funcionalidad de Visual Studio 2019 Enterprise, herramienta provista por la universidad. Se agregó un archivo de configuración para poder personalizar el análisis y evitar que se analicen los paquetes de Tests y excluimos los paquetes de Migrations y Resources. No se excluyeron otros paquetes en esta nueva instancia.

La justificación para excluir paquetes de nuestro análisis sigue siendo la misma, por la que la incluiremos nuevamente. El paquete de *Extentions en la Web Api* contiene un clase que crea una extensión para los servicios del startup. No se la excluyó, por que esperamos que en ese paquete se agreguen nuevas extensiones y esas si se prueben. La clase no se probó porque contiene métodos de terceros y dichos tests no aportarían valor a nuestra solución actualmente

La razón para excluir los paquetes de Migrations y Resources del análisis de cobertura, es que al ser código auto generado por herramientas desarrolladas por un tercero reconocido dentro de la industria y autor de las mismas -Microsoft-, nos sentimos en la libertad de confiar que es código confiable y funcional. Sin embargo, no significa que no se debería probar, pero creemos que el modo de prueba no deberían ser pruebas unitarias, sino otro tipo de pruebas como de integración que evalúen que nuestro sistema se adapte correctamente a la herramienta. En conclusión, al determinar que estos paquetes no serían probados, se decidió excluirlos del análisis, ya que no realizan un aporte significativo al resultado. [Documentación anterior]

A continuación el análisis general y por proyecto de cobertura.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.businesslogic.Exceptions	0	0.00%	922	100.00%
immrequest.businesslogic.Extentions	0	0.00%	2	100.00%
immrequest.businesslogic.Helpers	0	0.00%	350	93.83%
immrequest.businesslogic.Helpers.Automapper	9	5.00%	171	95.00%
immrequest.businesslogic.Helpers.Inputs	0	0.00%	4	100.00%
immrequest.businesslogic.Logic	7	3.35%	202	96.65%
immrequest.businesslogic.Logic.Finders	0	0.00%	60	100.00%
immrequest.businesslogic.Logic.ImporterLogic	3	1.49%	199	98.51%
immrequest.businesslogicValidators	3	0.72%	413	99.28%

Figura A.4: Cobertura general

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
ImmRequest.BusinessLogic.Exceptions	0	0.00%	12	100.00%
ImmRequest.BusinessLogic.Extentions	0	0.00%	34	100.00%
ImmRequest.BusinessLogic.Helpers	0	0.00%	48	100.00%
ImmRequest.BusinessLogic.Helpers.Automapper	0	0.00%	44	100.00%
ImmRequest.BusinessLogic.Helpers.Inputs	0	0.00%	17	100.00%
ImmRequest.BusinessLogic.Logic	7	3.35%	202	96.65%
ImmRequest.BusinessLogic.Logic.Finders	0	0.00%	60	100.00%
ImmRequest.BusinessLogic.Logic.ImporterLogic	3	1.49%	199	98.51%
ImmRequest.BusinessLogicValidators	3	0.72%	413	99.28%

Figura A.5: Cobertura Business Logic

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.dataaccess.dll	0	0.00%	922	100.00%
ImmRequest.DataAccess.Context	0	0.00%	170	100.00%
ImmRequest.DataAccess.Repositories	0	0.00%	752	100.00%

Figura A.6: Cobertura Data Access

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.dataaccess.dll	0	0.00%	922	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
ImmRequest.DataAccess.Interfaces.Exceptions	0	0.00%	2	100.00%

Figura A.7: Cobertura Cobertura Data Access Interfaces

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.dataaccess.dll	0	0.00%	922	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	23	6.17%	350	93.83%
ImmRequest.Domain	8	11.11%	64	88.89%
ImmRequest.Domain.Exceptions	0	0.00%	10	100.00%
ImmRequest.Domain.Fields	15	5.45%	260	94.55%
ImmRequest.Domain.UserManagement	0	0.00%	16	100.00%

Figura A.8: Cobertura Dominio

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.dataaccess.dll	0	0.00%	922	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	23	6.17%	350	93.83%
immrequest.importer.dll	9	5.00%	171	95.00%
{} ImmRequest.Importer.Domain	9	34.62%	17	65.38%
{} ImmRequest.Importer.Extentions	0	0.00%	8	100.00%
{} ImmRequest.Importer.Importers	0	0.00%	36	100.00%
{} ImmRequest.Importer.ImportersJson	0	0.00%	7	100.00%
{} ImmRequest.Importer.ImportersXML	0	0.00%	103	100.00%

Figura A.9: Cobertura Importer

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.dataaccess.dll	0	0.00%	922	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	23	6.17%	350	93.83%
immrequest.importer.dll	9	5.00%	171	95.00%
immrequest.importer.interfaces.dll	0	0.00%	4	100.00%
{} ImmRequest.Importer.Interfaces.Exceptions	0	0.00%	4	100.00%
immrequest.webapi.dll	186	15.00%	1054	85.00%

Figura A.10: Cobertura Importer Interfaces

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Usuario_DESKTOP-KGD1RJS 2020-06-20 18_31_51.coverage	231	6.14%	3532	93.86%
immrequest.businesslogic.dll	13	1.25%	1029	98.75%
immrequest.dataaccess.dll	0	0.00%	922	100.00%
immrequest.dataaccess.interfaces.dll	0	0.00%	2	100.00%
immrequest.domain.dll	23	6.17%	350	93.83%
immrequest.importer.dll	9	5.00%	171	95.00%
immrequest.importer.interfaces.dll	0	0.00%	4	100.00%
immrequest.webapi.dll	186	15.00%	1054	85.00%
{} ImmRequest.WebApi	97	100.00%	0	0.00%
{} ImmRequest.WebApi.Controllers	7	1.35%	513	98.65%
{} ImmRequest.WebApi.Exceptions	0	0.00%	2	100.00%
{} ImmRequest.WebApi.Extentions	43	100.00%	0	0.00%
{} ImmRequest.WebApi.Filters	1	2.56%	38	97.44%
{} ImmRequest.WebApi.Helpers	5	12.50%	35	87.50%
{} ImmRequest.WebApi.Helpers.Binders	10	21.74%	36	78.26%
{} ImmRequest.WebApi.Models	21	5.75%	344	94.25%
{} ImmRequest.WebApi.Models.Reporting	1	5.56%	17	94.44%
{} ImmRequest.WebApi.Models.UserManagement	1	1.43%	69	98.57%

Figura A.11: Cobertura Web api

A.3. Clean Code

Al igual que la ultima vez, se utilizó como guía el Libro de Clean Code escrito por Robert C. Martin para los estándares utilizados durante la etapa de desarrollo.
[4]

Se hizo énfasis en los siguientes puntos:

- Nombres de funciones y variables que dieran sentido y contexto al lector del código.
- Limitar la cantidad de parámetros que recibe un método o constructores de clases. En caso de necesitarlos todos, creamos DTOs para hacer que el código sea mas limpio.
- Se evito hardcodear valores de mensajes y usaron constantes y resources.
- Evitamos hacer RTTI
- Evitamos el uso de comentarios.
- Refactoreamos métodos a medida que crecían y buscamos que tuvieran una única responsabilidad. Procuramos mantener el mismo nivel de abstracción dentro de los métodos.
- Preferimos mayor cantidad de métodos por clase para mantener una sola responsabilidad y nivel de abstracción de los mismos.
- Aprovechamos las herramientas de polimorfismo del lenguaje para evitar y reducir el reuso de código utilizando herencias e interfaces.

Algunos ejemplos en el código:

Un ejemplo del ultimo punto, son las clases *Json Importer* y *Xml Importer*, son clases abstractas que redefinen los métodos de la interfaz para atrapar los errores comunes a estos tipos de importadores, pero le permiten a sus hijos implementar su propia lógica.

```
private const string TYPE_TAG = "Type";
private const string AREA_TAG = "Area";
private const string TOPIC_TAG = "Topic";

private const string ATTRIBUTE_NAME = "name";
private const string ATTRIBUTE_ID = "id";
```

Figura A.12: Uso de constantes para prevenir strings mágicos

```
var topics = ImportTopics(topicNodes);
var name = GetAttributeAs<string>(xmlArea, ATTRIBUTE_NAME);
var id = GetAttributeAs<long>(xmlArea, ATTRIBUTE_ID);
    . . .
```

Figura A.13: Nombre de métodos comprensibles y fáciles de leer

```
public static class TestConstants
{
    private const string ROOT_PATH = "~\\..\\..\\..\\..\\..\\..\\{0}\\Files\\";

    7 references | 7/7 passing | juliette Ruchel, 27 days ago | 1 author, 1 change
    public static string JsonPath
    {
        get
        {
            return string.Format(ROOT_PATH, "JsonTests");
        }
    }

    7 references | 7/7 passing | juliette Ruchel, 27 days ago | 1 author, 1 change
    public static string XMLPath
    {
        get
        {
            return string.Format(ROOT_PATH, "XMLTests");
        }
    }
}
```

Figura A.14: Uso de clases que contiene constantes para evitar repetir las mismas

Este ultimo ejemplo es de uno de los paquetes de tests, pero creemos que estos deben ser tan mantenidos como el código común.

B. Documentación API

B.1. Estándares

La aplicación fue creada siguiendo los estándares establecidos por REST, un estilo de arquitectura de aplicaciones propuesto en el año 2000 por Roy Fielding.[6] Esto consiste en:

- Usa una interfaz uniforme, que ayuda a desacoplar las implementaciones del cliente y el servidor.
- Permite organizar la Api según los recursos del sistema. Por ejemplo, en esta Api se crearon controllers basándose en recursos existentes del sistema: Session, CitizenRequest, Type y Administrator.
- Las urls se deben basar en sustantivos y no verbos.
Por ejemplo */api/CitizenRequest/2*
- Una colección es un recurso separado del item dentro de la colección, y debería tener su propia URI.
- Se utilizaron los métodos comunes a la mayoría de las Apis REST : GET, PUT, POST & DELETE
- Se utilizaron códigos de status HTTP para indicar el estado de la respuesta al usuario

B.2. Mecanismos de Autenticación

Además de un mecanismo de autenticación básico como es el login, se agregó un filtro para controlar que el administrador que intenta acceder a los endpoints del sistema tenga acceso.

B.2.1. Authorization Filter

Verifica que quienes quieren envían una solicitud a un endpoint cumplan ciertas condiciones. Las condiciones son :

- Enviar un token en el header indicando que el administrador está logueado en el sistema.

Este filtro se puede utilizar al tener un administrador logueado en el sistema. Se deberá usar al permitir a un conjunto de usuarios (ya sea administrador o otros roles que puedan ser agregados) acceder a ciertas funcionalidades del sistema.

B.3. Códigos Utilizados

- OK Status code : 200 Indica que la acción realizada se ejecutó correctamente
- Bad Request Status Code : 400 Indica que hubo un error al realizar la acción requerida.
- Forbidden Status Code : 403 Indica que el servidor se niega a permitir la acción requerida.
- Internal Server Error: 500 Indica un error en el servidor al ejecutar una cierta acción.

B.4. Especificación de Endpoints

Para realizar esta parte, se utilizó el paquete de Swagger [7] para .Net Core.

B.4.1. Administrator Controller

The screenshot shows the Swagger UI for a GET request to the endpoint `/api/Administrator/{id}`. The endpoint is described as "Permite a un administrador obtener información de cualquier administrador del sistema".

Parameters

Name	Description
id * required integer (path)	Este parámetro contiene el identificador del administrador id - Este parámetro contiene el identificador c

Authorization * required

Code	Description	Links
200	Se devuelve la información requerida.	No links
400	Administrador no existente.	No links
403	Token invalido o vacio	No links

PUT /api/Administrator/{id} Permite a un administrador actualizar información de otro administrador en el sistema

Parameters

Name	Description
id * required integer (path)	Este parámetro contiene el identificador del administrador id - Este parámetro contiene el identificador c
Authorization * required Guid (header)	Access token Authorization - Access token

Request body

Este modelo contiene la información a actualizar del administrador

Example Value | Schema

```
{
  "id": 0,
  "email": "string",
  "username": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	Se actualizó información del administrador	No links
400	Error. No se actualizó información del administrador	No links
403	Token invalido o vacio	No links

DELETE /api/Administrator/{id} Permite a un administrador borrar otro administrador del sistema

Parameters

Name	Description
id * required integer (path)	Este parámetro contiene el identificador del administrador id - Este parámetro contiene el identificador c
Authorization * required Guid (header)	Access token Authorization - Access token

Responses

Code	Description	Links
200	Se borró el administrador del sistema	No links
400	Error. No se pudo borrar al administrador	No links
403	Token invalido o vacio	No links

GET /api/Administrator Permite a un administrador obtener información de todos los administradores del sistema

Parameters

Name	Description
Authorization * required	Access token Guid (header)
	Authorization - Access token

Responses

Code	Description	Links
200	Se devuelve la información requerida.	No links
403	Token invalido o vacio	No links

POST /api/Administrator Permite a un administrador crear otro administrador en el sistema

Parameters

Name	Description
Authorization * required	Access token Guid (header)
	Authorization - Access token

Request body

Este modelo contiene la información del administrador

Example Value | Schema

```
{
  "id": 0,
  "email": "string",
  "username": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	Se creó el administrador	No links
400	Error. No se creó el administrador	No links
403	Token invalido o vacio	No links

B.4.2. Citizen Request Controller

POST /api/CitizenRequest Permite al usuario ingresar una solicitud al sistema

Parameters

No parameters

Request body

Este modelo contiene la información acerca de la solicitud

Example Value | Schema

```
{
  "id": 0,
  "description": "string",
  "citizenName": "string",
  "email": "string",
  "phone": "string",
  "areaId": 0,
  "topicId": 0,
  "topicTypePrid": 0,
  "status": "string",
  "values": [
    {
      "id": 0,
      "parentCitizenRequestId": 0,
      "fieldId": 0,
      "value": [
        "string"
      ]
    }
  ]
}
```

Responses

Code	Description	Links
200	Se creó la solicitud con éxito	No links
400	Error. No se creó la solicitud	No links

GET /api/CitizenRequest Permite al administrador obtener todas las solicitudes del sistema.

Parameters

Name Description

Authorization * required
Guid
(header)
Access token
Authorization - Access token

Responses

Code	Description	Links
200	Se devuelve la solicitud requerida.	No links
403	Token invalido o vacio	No links

GET /api/CitizenRequest/{requestId} Permite al administrador obtener una solicitud existente en el sistema

Parameters

Name	Description
requestId * required integer (path)	Este parámetro contiene el número de la solicitud existente requestId - Este parámetro contiene el número de la solicitud existente
Authorization * required Guid (header)	Access token Authorization - Access token

Responses

Code	Description	Links
200	Se devuelve la solicitud requerida.	No links
400	La solicitud no ha sido encontrada.	No links
403	Token invalido o vacio	No links

PUT /api/CitizenRequest/{requestId} Permite al administrador actualizar el status de una solicitud existente en el sistema

Parameters

Name	Description
requestId * required integer (path)	Este parámetro contiene el número de la solicitud existente requestId - Este parámetro contiene el número de la solicitud existente
Authorization * required Guid (header)	Access token Authorization - Access token

Request body

Este parámetro contiene el nuevo status que debe tener la solicitud

Example Value | Schema

```
{
  "status": 0
}
```

Responses

Code	Description	Links
200	La solicitud requerida fue actualizada.	No links
400	Status no existe o error en el sistema.	No links
403	Token invalido o vacio	No links

GET /api/CitizenRequest>Status/{requestId} Permite a un usuario obtener el status de su solicitud

Parameters

Name	Description
requestId * required integer (path)	Este parámetro contiene el número del solicitud existente requestId - Este parámetro contiene el número

Responses

Code	Description	Links
200	Se devuelve el status de la solicitud.	No links
400	La solicitud no ha sido encontrada.	No links

GET /api/CitizenRequest/Areas Permite a un usuario obtener todas las áreas del sistema

Parameters

No parameters	
---------------	--

Responses

Code	Description	Links
200	Se devuelven las áreas existentes en el sistema.	No links

GET /api/CitizenRequest/Topics/{parentAreaId} Permite a un usuario obtener todos los temas de un área del sistema.

Parameters

Name	Description
parentAreaId * required integer (path)	Este parámetro contiene el número de área al cual los temas pertenecen parentAreaId - Este parámetro contiene el númer

Responses

Code	Description	Links
200	Se devuelven los temas para el área en el sistema.	No links

B.4.3. Importer Controller

POST /api/Importer Permite a un administrador realizar ingresar nuevas areas usando un importador dado.

Parameters

Name	Description
Authorization * required	Access token
Guid (header)	Authorization - Access token

Request body

multipart/form-data

importer
string

ContentType
string

ContentDisposition
string

Headers
object

Length
integer(\$int64)

Name
string

FileName
string

Responses

Code	Description	Links
200	Success	No links
403	Token invalido o vacio	No links

GET /api/Importer Retorna la lista de importadores en el sistema.

Parameters

Name	Description
Authorization * required	Access token
Guid (header)	Authorization - Access token

Responses

Code	Description	Links
200	Success	No links
403	Token invalido o vacio	No links

B.4.4. Reports Controller

POST /api/Reports/RequestSummary Retorna la cantidad de solicitudes por estado realizadas por un mismo usuario en un rango de fechas.

Parameters

Name **Description**

Authorization * required
Access token
(header)
Authorization - Access token

Request body

application/json-patch+json

Example Value | Schema

```
{ "email": "string", "start": "2020-06-21T00:09:01.766Z", "end": "2020-06-21T00:09:01.766Z" }
```

Responses

Code	Description	Links
200	Success	No links
403	Token invalido o vacio	No links

POST /api/Reports/TypesSummary Retorna los tipos más usados en las solicitudes en un rango de fechas.

Parameters

Name **Description**

Authorization * required
Access token
(header)
Authorization - Access token

Request body

application/json-patch+json

Example Value | Schema

```
{ "email": "string", "start": "2020-06-21T00:09:01.779Z", "end": "2020-06-21T00:09:01.779Z" }
```

Responses

Code	Description	Links
200	SUCCESS	No links
403	Token invalido o vacio	No links

B.4.5. Session Controller

POST /api/Session Permite a un administrador loguearse al sistema.

Parameters

No parameters

Request body

Este modelo contiene la información para iniciar sesión
[Example Value](#) | [Schema](#)

```
{
  "id": 0,
  "username": "string",
  "email": "string",
  "password": "string",
  "token": "3fb89f64-5717-4562-b3fc-2c963f66afa6"
}
```

Responses

Code	Description	Links
200	Se inició sesión con éxito	No links
400	Error. No se pudo iniciar sesión.	No links

DELETE /api/Session Permite a un administrador cerrar sesión en el sistema.

Parameters

No parameters

Responses

Code	Description	Links
200	Se cerró sesión con éxito	No links
400	Error. No se pudo cerrar la sesión.	No links

B.4.6. Type Controller

POST /api/Type/{parentTopicID} Permite a un usuario crear un tipo de solicitud especificando su tema.

Parameters

Name	Description
parentTopicID * required integer (path)	Este parámetro contiene el identificador del tema parentTopicID - Este parámetro contiene el ic
Authorization * required Guid (header)	Access token Authorization - Access token

Request body

Este modelo contiene la información del nuevo tipo.
[Example Value](#) | [Schema](#)

```
{ "name": "string", "id": 0, "fields": [ { "id": 0, "name": "string", "parentTypeID": 0, "multipleValues": true, "rangeValues": [ "string" ], "dataType": 0 } ] }
```

Responses

Code	Description	Links
200	Se creó el tipo con éxito	No links
400	Error. No se pudo crear el tipo.	No links
403	Token invalido o vacio	No links

GET /api/Type/{id} Permite a un usuario obtener un tipo.

Parameters

Name	Description
id * required integer (path)	Este parámetro contiene el identificador del tipo en el sistema id - Este parámetro contiene el identificador c

Responses

Code	Description	Links
200	Se obtuvo el tipo con éxito	No links
400	Error. No se pudo obtener el tipo.	No links

DELETE /api/Type/{id} Permite a un usuario borrar un tipo del sistema.

Parameters

Name	Description
id <small>* required</small> integer (path)	Este parámetro contiene el identificador del tipo en el sistema id - Este parámetro contiene el identificador c

Authorization * required
Guid
(header)

Access token
Authorization - Access token

Responses

Code	Description	Links
200	Se borró el tipo del sistema	No links
400	Error. No se pudo borrar al tipo	No links
403	Token invalido o vacio	No links

PUT /api/Type/{id} Permite a un usuario actualizar la información de un tipo.

Parameters

Name	Description
id <small>* required</small> integer (path)	Este parámetro contiene el identificador del tipo en el sistema id - Este parámetro contiene el identificador c

Authorization * required
Guid
(header)

Access token
Authorization - Access token

Request body

application/json-patch+json

Example Value | Schema

```
{
  "name": "string",
  "id": 0,
  "fields": [
    [
      {
        "id": 0,
        "name": "string",
        "parentTypeId": 0,
        "multipleValues": true,
        "rangeValues": [
          "string"
        ],
        "dataType": 0
      }
    ]
  ]
}
```

Responses

Code	Description	Links
200	Se actualizó el tipo con éxito	No links
400	Error. No se pudo actualizar el tipo.	No links
403	Token invalido o vacio	No links

The image shows two screenshots of the Swagger UI interface, each representing a different API endpoint.

Endpoint 1: /api/type

- Method:** GET
- Description:** Permite a un usuario obtener todos los tipos del sistema.
- Parameters:** No parameters
- Responses:**
 - Code:** 200
Description: Se borró el administrador del sistema
 - Links:** No links

Endpoint 2: /api/type/all/{parentTopicId}

- Method:** GET
- Parameters:**

Name	Description
parentTopicId <small>required</small>	parentTopicId <small>integer (path)</small>
- Responses:**
 - Code:** 200
Description: Success
 - Links:** No links

B.5. Visualizar documentación de la API online

En la carpeta *Documentación* se adjunta el archivo *swagger.json*. Este permite visualizar la documentación de la Api desde la página <https://editor.swagger.io/> de la siguiente forma:

- En el menú, ir a *File* -> *ImportFile*.
- Importar el archivo *swagger.json* encontrado en la carpeta *Documentación*.

C. Tutorial Importador

C.1. Aclaraciones Generales

La herramienta del importador permite ingresar nuevas áreas con sus respectivos temas y sub tipos. Adicionalmente, permite ingresar nuevos temas a áreas existentes o nuevos tipos a temas existentes.

Para que el sistema puede diferenciar entre nuevas áreas y las ya existentes, a la hora de crear el archivo, las áreas existentes deberán tener su Id. Esto aplica de igual forma para los temas. No se podrá asignar un tema existente, a un área nueva.

Es importante destacar que si algún objeto a ingresar del archivo falla, el resto no serán ingresados, de forma de preservar la integridad de los datos. En las siguientes dos secciones, se encontraran ejemplos para las implementaciones incluidas.

C.2. Json Area Importer

```
[  
  {  
    "name": "Tramites", //new Area  
    "topics": [  
      {  
        "name": "Horario de atención",  
        "types": []  
      },  
      {  
        "name": "Docuemntacion",  
        "types": [  
          {  
            "name": "Falta informacion sobre documentos requeridos"  
          },  
          {  
            "name": "Falta de colaboracion"  
          }  
        ]  
      }  
    ],  
    {  
      "name": "Transporte", //Existing Area  
      "id": "4",  
      "topics": [  
        {  
          "name": "Paradas", //Existing Sub Topic  
          "id": "14",  
          "types": [  
            {  
              "name": "Sin Lista de Omnibus"  
            },  
            {  
              "name": "Sin bancos o techo"  
            }  
          ]  
        },  
        {  
          "name": "Paros", //New Topic for Existing Area  
          "types": [  
            {  
              "name": "Paros publicos"  
            },  
            {  
              "name": "Paros Privados"  
            }  
          ]  
        }  
      ]  
    }  
  ]
```

Figura C.1: Ejemplo de archivo json

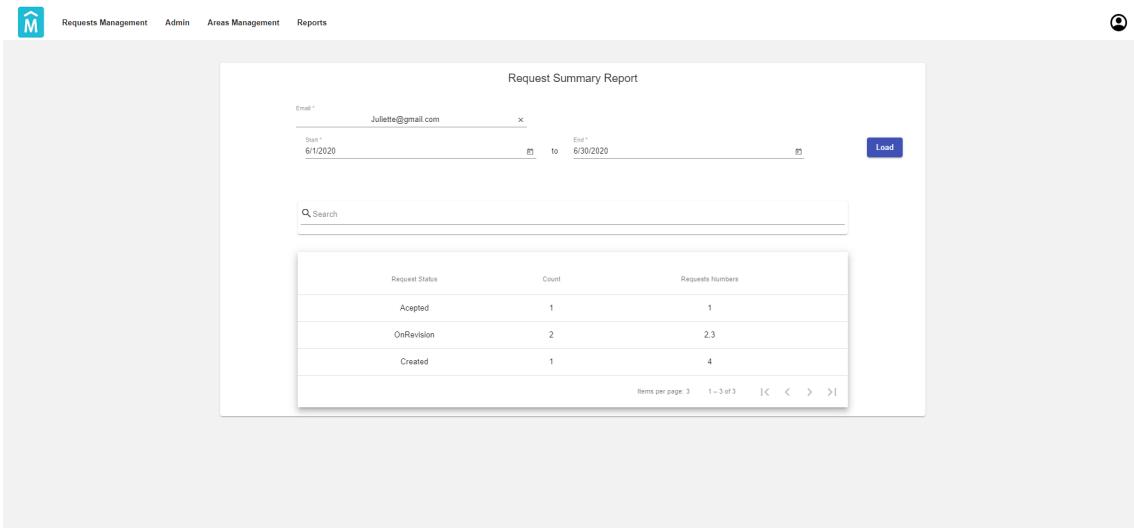
C.3. Xml Area Importer

```
<?xml version="1.0" encoding="utf-8"?>
<List>
  <Area name="Eventos" > <!--New Area-->
    <Topic name="Gubernamentales">
      <Type name="Dias Festivos"></Type>
    </Topic>
    <Topic name="Internacionales">
      <Type name="Deportivos"></Type>
      <Type name="Musicales"></Type>
    </Topic>
  </Area>
  <Area name="Saneamiento" id="3">
    <Topic name="Obstrucciones o Perdidas" id="11" > <!--New Types For Existing Topic-->
      <Type name="Perdidas de Gas"></Type>
      <Type name="Inundacion en edificio"></Type>
    </Topic>
    <Topic name="Mantenimiento" > <!--New Topic For Existing Area-->
      <Type name="Mantenimiento de cañerias"></Type>
      <Type name="Otros"></Type>
    </Topic>
  </Area>
</List>
```

Figura C.2: Ejemplo de archivo xml

D. Páginas Front End

D.1. Reporte A: Request Summary



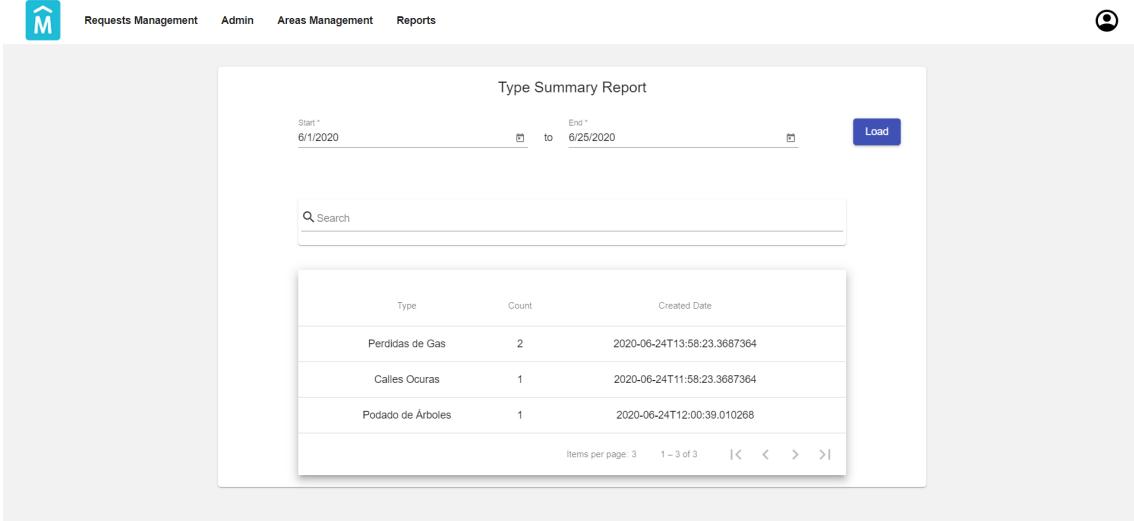
The screenshot shows a web-based reporting application. At the top, there is a navigation bar with icons for Requests Management, Admin, Areas Management, and Reports. Below the navigation, a search bar displays the email address "Juliette@gmail.com". Underneath the search bar, there are date filters: "Start" set to "6/1/2020" and "End" set to "6/30/2020". To the right of these filters is a blue "Load" button. Below the search bar is a search input field with the placeholder "Search". The main content area is titled "Request Summary Report" and contains a table with three columns: "Request Status", "Count", and "Requests Numbers". The table data is as follows:

Request Status	Count	Requests Numbers
Accepted	1	1
OnRevision	2	2,3
Created	1	4

At the bottom of the table, there is a footer with the text "Items per page: 3 1 - 3 of 3 | < > >>|".

Figura D.1: Reporte A

D.2. Reporte B: Type Summary



The screenshot shows a web-based application interface for a 'Type Summary Report'. At the top, there is a navigation bar with icons for 'Requests Management', 'Admin', 'Areas Management', and 'Reports'. On the right side of the header is a user profile icon.

The main content area is titled 'Type Summary Report'. It features a search bar with the placeholder 'Search' and a date range selector with 'Start' set to '6/1/2020' and 'End' set to '6/25/2020'. A blue 'Load' button is positioned to the right of the date fields.

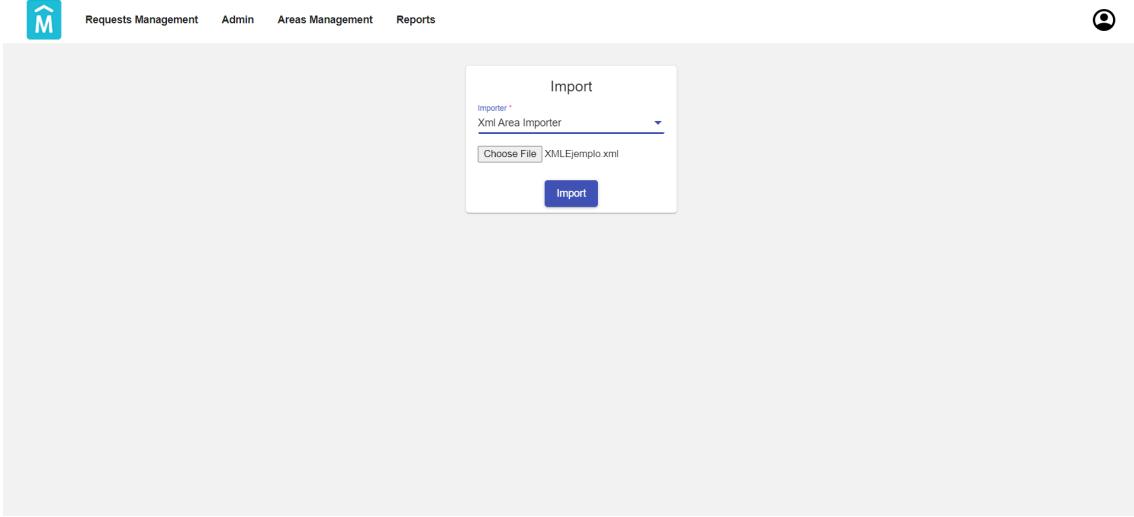
Below the search and date fields is a table with three rows of data:

Type	Count	Created Date
Perdidas de Gas	2	2020-06-24T13:58:23.3687364
Calles Ocuras	1	2020-06-24T11:58:23.3687364
Podado de Árboles	1	2020-06-24T12:00:39.010268

At the bottom of the table, there is a pagination control with the text 'Items per page: 3' followed by '1 – 3 of 3' and a set of navigation arrows.

Figura D.2: Reporte B

D.3. Importar XML



The screenshot shows a modal dialog box titled 'Import'. Inside the dialog, there is a dropdown menu labeled 'Importer' with the option 'Xml Area Importer' selected. Below the dropdown is a file input field with the placeholder 'Choose File' and the path 'XML/Ejemplo.xml' displayed. At the bottom of the dialog is a blue 'Import' button.

Figura D.3: Importar archivo XML

D.4. Solicitudes

The screenshot shows a user interface for creating a request. At the top right is a profile icon. The main area has a title 'Make a request!' and a numbered list of steps:

- 1 Select an area:
 - Espacios Publicos y Calles
 - Limpieza
 - Saneamiento
 - Transporte
- 2 Select a topic.
- 3 Select topic's type.
- 4 Describe your issue.
- 5 Tell us about yourself.
- 6 Add any additional fields you need.

Figura D.4: Creación de una solicitud

Request Number	Request Description	Citizen Name	Email	Phone	Area Name	Topic Name	Topic Type Name	Status	Actions
1	No hay luz en mi calle	Juliette	Juliette@gmail.com	0990999099	Espacios Publicos y Calles	Alumbrado	Calles Ocuras	Created	
2	Poder árboles por favor, gracias	Juliette	Juliette@gmail.com	0990999099	Espacios Publicos y Calles	Arbolado y Plantacion	Podado de Árboles	Created	
3	Hay un perdida de gas en el edificio y demoraron mucho en resolverla	Juliette	Juliette@gmail.com	0990999099	Saneamiento	Obstrucciones o Perdidas	Perdidas de Gas	Created	
4	Hubo un perdida en mi edificio. Help	Juliette	Juliette@gmail.com	0990999099	Saneamiento	Obstrucciones o Perdidas	Perdidas de Gas	Created	

Items per page: 5 0 of 0 |< < > >|

Figura D.5: Listado de solicitudes

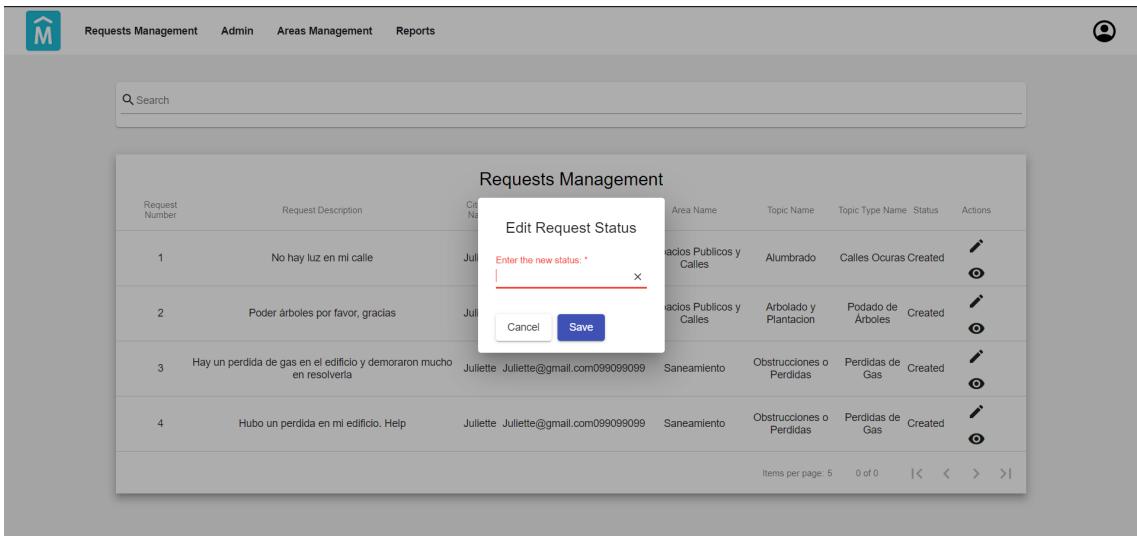


Figura D.6: Cambio de estado de solicitud

D.5. Administradores

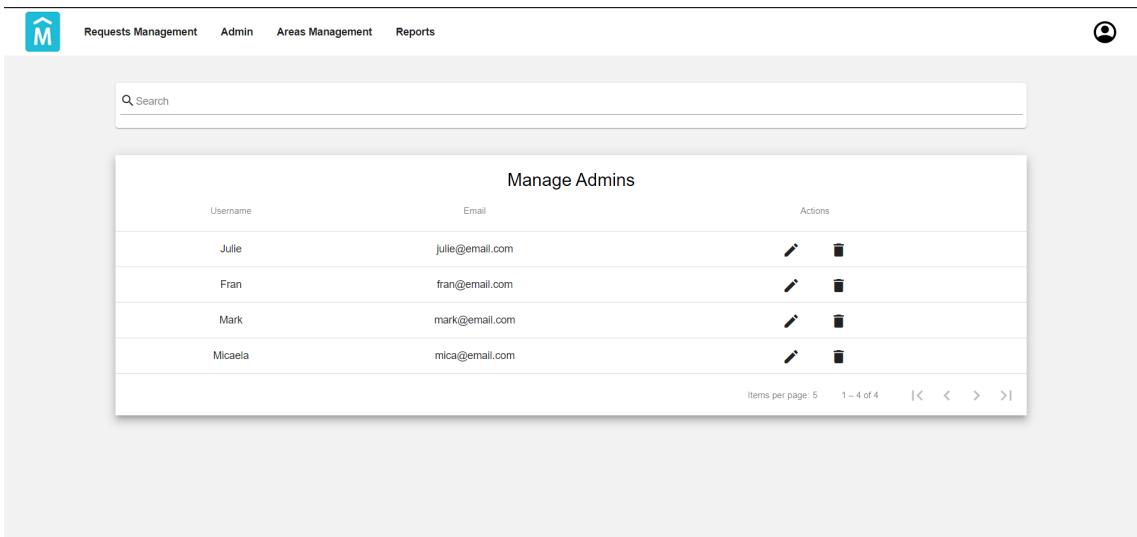


Figura D.7: Lista de Administradores

The screenshot shows a web-based application interface for managing users. At the top, there is a navigation bar with icons for Requests Management, Admin, Areas Management, and Reports. On the right side of the header is a user profile icon. The main content area is titled "Edit Admin". It contains three input fields: "Username *", which has "Julie" typed into it; "Email *", which has "julie@email.com" typed into it; and "Password *". Below these fields is a small "Edit" button. The background of the page is light gray.

Figura D.8: Edición de Administrador

D.6. Áreas, Temas y Tipos

The screenshot shows a web-based application interface for managing areas. At the top, there is a navigation bar with icons for Requests Management, Admin, Areas Management, and Reports. On the right side of the header is a user profile icon. Below the header is a search bar with the placeholder text "Search". The main content area is titled "Areas" and displays a table with four rows of data. The columns are labeled "#", "Areas", and "Actions". The data is as follows:

#	Areas	Actions
1	Espacios Publicos y Calles	
2	Limpieza	
3	Saneamiento	
4	Transporte	

At the bottom of the table, there is a pagination control with the text "Items per page: 5 1 – 4 of 4 < < > > |".

Figura D.9: Todas las áreas

The screenshot shows a web application interface for managing areas. At the top, there is a navigation bar with links for Requests Management, Admin, Areas Management, and Reports. On the far right of the header is a user profile icon. Below the header, a breadcrumb navigation shows 'All Areas'. A search bar is present. The main content area is titled 'Espacios Publicos y Calles'. It displays a table with five rows, each representing a topic. The columns are labeled '#', 'Topics', and 'Actions'. The topics listed are: 1. Alumbrado, 2. Arbolado y Plantacion, 3. Equipamiento Urbano, 4. Fuentes, Graffitis e Instalaciones, and 5. Otros. Each row has a edit icon in the 'Actions' column. At the bottom of the table, there is a pagination control showing 'Items per page: 5', '1 – 5 of 5', and navigation arrows.

Figura D.10: Área con sus temas

The screenshot shows a web application interface for managing topics. At the top, there is a navigation bar with links for Requests Management, Admin, Areas Management, and Reports. On the far right of the header is a user profile icon. Below the header, a breadcrumb navigation shows 'Area'. A search bar is present. The main content area is titled 'Alumbrado'. It displays a table with one row, representing a type. The columns are labeled 'Types' and 'Actions'. The type listed is 'Calles Ocuras'. There is an edit icon in the 'Actions' column. At the bottom of the table, there is a pagination control showing 'Items per page: 5', '1 – 1 of 1', and navigation arrows. A blue plus sign icon is located above the search bar.

Figura D.11: Tema con sus tipos

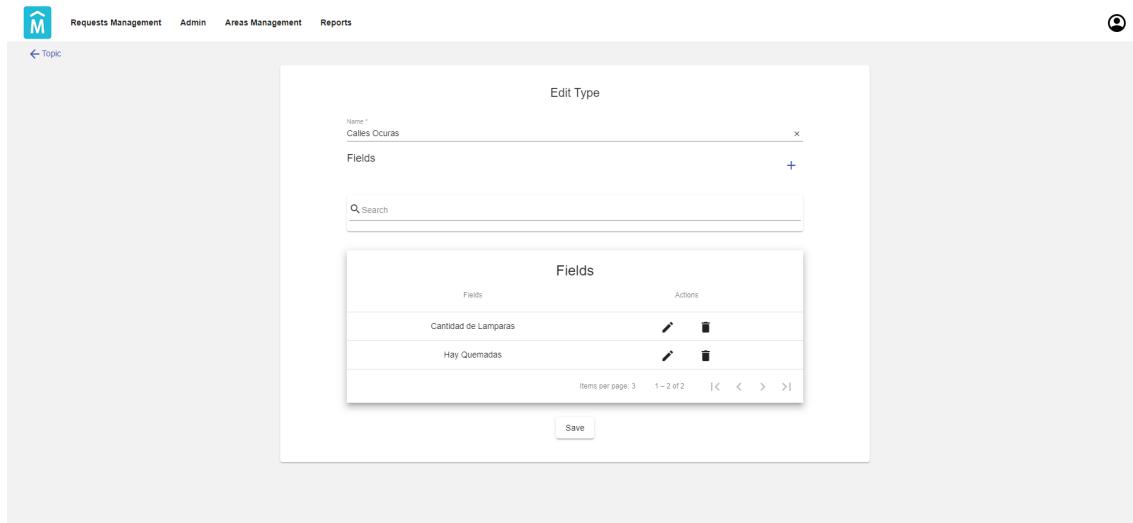


Figura D.12: Editor de Tipo

E. Manual Deploy

A continuación se detallarán los pasos para realizar el deploy de la aplicación.

E.1. Deploy Lado Servidor

1. En la carpeta de la solución, ejecutamos el siguiente comando:
dotnet build -c Release

```
C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126>dotnet build -c Release
Microsoft (R) Build Engine version 16.6.0+5ff70bc9e for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
InnRequest.Domain.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.Domain.Interfaces\bin\Release\netstandard2.0\
  1 file(s) copied.
  1 file(s) copied.
InnRequest.DataAccess.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.DataAccess.Interfaces\bin\Release\netstandard2.0\
InnRequest.Importer.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.Importer.Interfaces\bin\Release\netstandard2.0\
InnRequest.WebApi.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.WebApi.Interfaces\bin\Release\netstandard2.0\
InnRequest.Importer -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.Importer\bin\Release\netstandard2.0\InnRequest.Importer.dll
InnRequest.Domain -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.Domain\bin\Release\netstandard2.0\InnRequest.Domain.dll
InnRequest.DataAccess -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.DataAccess\bin\Release\netstandard2.0\InnRequest.DataAccess.dll
InnRequest.DataAccess.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.DataAccess.Tests\bin\Release\netcoreapp3.1\InnRequest.DataAccess.Tests.dll
InnRequest.BusinessLogic.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.BusinessLogic.Interfaces\bin\Release\netstandard2.0\
InnRequest.Importer.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.Importer.Tests\bin\Release\netcoreapp3.1\InnRequest.Importer.Tests.dll
InnRequest.BusinessLogic -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.BusinessLogic\bin\Release\netstandard2.0\InnRequest.BusinessLogic.dll
InnRequest.BusinessLogic.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.BusinessLogic.Tests\bin\Release\netcoreapp3.1\InnRequest.BusinessLogic.Tests.dll
CSC : warning CS8034: Unable to load Analyzer assembly C:\Program Files\dotnet\sdk\NuGetFallbackFolder\microsoft.aspnetcore.mvc.analyzers\2.2.0\analyzers\dotnet
dy loaded [C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.WebApi\InnRequest.WebApi.csproj]
  InnRequest.WebApi -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.WebApi\bin\Release\netcoreapp3.1\InnRequest.WebApi.dll
ControllerTests\ReportControllerTests.cs(42,25): warning CS0160: The field 'ReportControllerTests.AllTypes' is never used [C:\Users\image1\Desktop\Francisco\Fac
ts.csproj]
  InnRequest.WebApi.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.WebApi.Tests\bin\Release\netcoreapp3.1\InnRequest.WebApi.Tests.dll
Build succeeded.

CSC : warning CS8034: Unable to load Analyzer assembly C:\Program Files\dotnet\sdk\NuGetFallbackFolder\microsoft.aspnetcore.mvc.analyzers\2.2.0\analyzers\dotnet
dy loaded [C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203942-233126\InnRequest.WebApi\InnRequest.WebApi.csproj]
  ControllerTests\ReportControllerTests.cs(42,25): warning CS0160: The field 'ReportControllerTests.AllTypes' is never used [C:\Users\image1\Desktop\Francisco\Fac
ts.csproj]
    2 Warning(s)
    0 Error(s)

Time Elapsed: 00:00:06.78
```

Figura E.1: Output de la ejecución en línea de comandos

2. Luego, también en la carpeta de la solución, ejecutamos el siguiente comando:
`dotnet publish -c Release`

```
C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126>dotnet publish -c Release
Microsoft (R) Build Engine version 16.6.0+5FF70BC09 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
ImmRequest.DataAccess.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.DataAccess.Interfaces\bin\Release\netstandard2.0\ImmRequest.DataAccess.Interfaces.dll
ImmRequest.DataAccess.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.DataAccess.Interfaces\bin\Release\netstandard2.0\ImmRequest.DataAccess.Interfaces.dll
1 file(s) copied.
ImmRequest.Domain.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Domain.Interfaces\bin\Release\netstandard2.0\ImmRequest.Domain.Interfaces.dll
ImmRequest.Domain.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Domain.Interfaces\bin\Release\netstandard2.0\ImmRequest.Domain.Interfaces.dll
1 file(s) copied.
ImmRequest.Importer.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer.Interfaces\bin\Release\netstandard2.0\ImmRequest.Importer.Interfaces.dll
ImmRequest.Importer.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer.Interfaces\bin\Release\netstandard2.0\ImmRequest.Importer.Interfaces.dll
1 file(s) copied.
ImmRequest.WebApi.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.WebApi.Interfaces\bin\Release\netstandard2.0\ImmRequest.WebApi.Interfaces.dll
ImmRequest.WebApi.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.WebApi.Interfaces\bin\Release\netstandard2.0\ImmRequest.WebApi.Interfaces.dll
1 file(s) copied.
ImmRequest.Importer -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer\bin\Release\netstandard2.0\ImmRequest.Importer.dll
ImmRequest.Importer -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer\bin\Release\netstandard2.0\ImmRequest.Importer.dll
1 file(s) copied.
ImmRequest.Domain -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Domain\bin\Release\netstandard2.0\ImmRequest.Domain.dll
ImmRequest.Domain -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Domain\bin\Release\netstandard2.0\ImmRequest.Domain.dll
1 file(s) copied.
ImmRequest.DataAccess -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.DataAccess\bin\Release\netstandard2.0\ImmRequest.DataAccess.dll
ImmRequest.BusinessLogic.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.BusinessLogic.Interfaces\bin\Release\netstandard2.0\ImmRequest.BusinessLogic.Interfaces.dll
ImmRequest.BusinessLogic.Interfaces -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.BusinessLogic.Interfaces\bin\Release\netstandard2.0\ImmRequest.BusinessLogic.Interfaces.dll
1 file(s) copied.
ImmRequest.Importer.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer.Tests\bin\Release\netcoreapp3.1\ImmRequest.Importer.Tests.dll
ImmRequest.DataAccess -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.DataAccess\bin\Release\netstandard2.0\ImmRequest.DataAccess.dll
ImmRequest.Importer.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer.Tests\bin\Release\netcoreapp3.1\ImmRequest.Importer.Tests.dll
ImmRequest.Importer -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer\bin\Release\netstandard2.0\ImmRequest.Importer.dll
ImmRequest.Domain -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Domain\bin\Release\netstandard2.0\ImmRequest.Domain.dll
ImmRequest.DataAccess -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.DataAccess\bin\Release\netstandard2.0\ImmRequest.DataAccess.dll
ImmRequest.BusinessLogic -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.BusinessLogic\bin\Release\netstandard2.0\ImmRequest.BusinessLogic.dll
ImmRequest.BusinessLogic.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.BusinessLogic.Tests\bin\Release\netcoreapp3.1\ImmRequest.BusinessLogic.Tests.dll
ImmRequest.DataAccess -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.DataAccess\bin\Release\netstandard2.0\ImmRequest.DataAccess.dll
ImmRequest.Importer.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Importer.Tests\bin\Release\netcoreapp3.1\ImmRequest.Importer.Tests.dll
ImmRequest.Domain -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.Domain\bin\Release\netstandard2.0\ImmRequest.Domain.dll
ImmRequest.WebApi -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.WebApi\bin\Release\netcoreapp3.1\ImmRequest.WebApi.dll
ImmRequest.WebApi.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.WebApi.Tests\bin\Release\netcoreapp3.1\ImmRequest.WebApi.Tests.dll
ImmRequest.WebApi -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.WebApi\bin\Release\netcoreapp3.1\ImmRequest.WebApi.dll
ImmRequest.WebApi.Tests -> C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126\ImmRequest.WebApi.Tests\bin\Release\netcoreapp3.1\ImmRequest.WebApi.Tests.dll
C:\Users\image1\Desktop\Francisco\Facultad\Diseno2\2020\203042-233126>
```

Figura E.2: Output de la ejecución en línea de comandos

3. Copiamos la carpeta publish generada (localizada en `{raiz del proyecto}/ImmRequest.WebApi/bin/Release/netcoreapp3.1`) en el path `C:/inetpub/wwwroot`

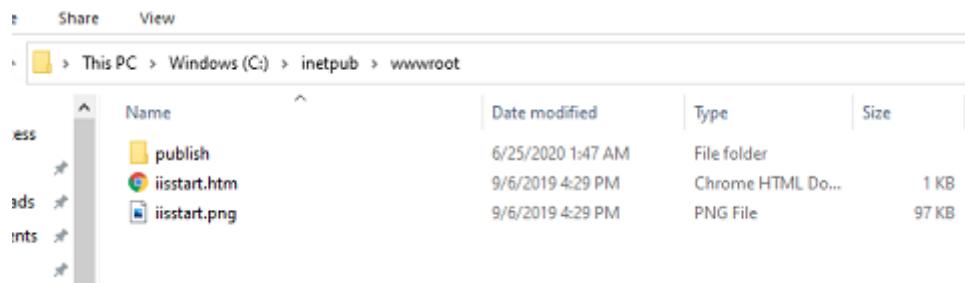


Figura E.3: Path de carpeta publish.

- Copiamos a la carpeta publish, copiada en el paso anterior, la carpeta *Importers* localizada en la carpeta de la solución.
- Cambiamos el connection string (definido en appsettings.json) de la base de datos al correspondiente a nuestra máquina en la carpeta publish.

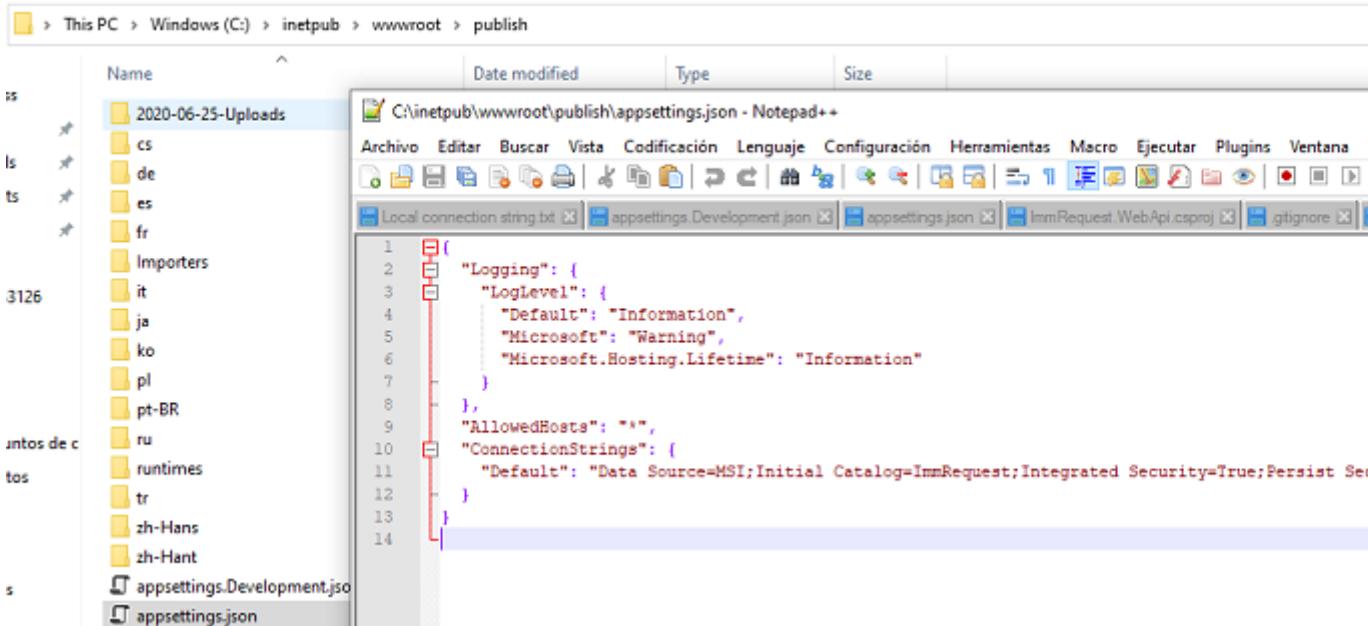


Figura E.4: Connection strings en appsettings.json.

- En IIS creamos un sitio nuevo:

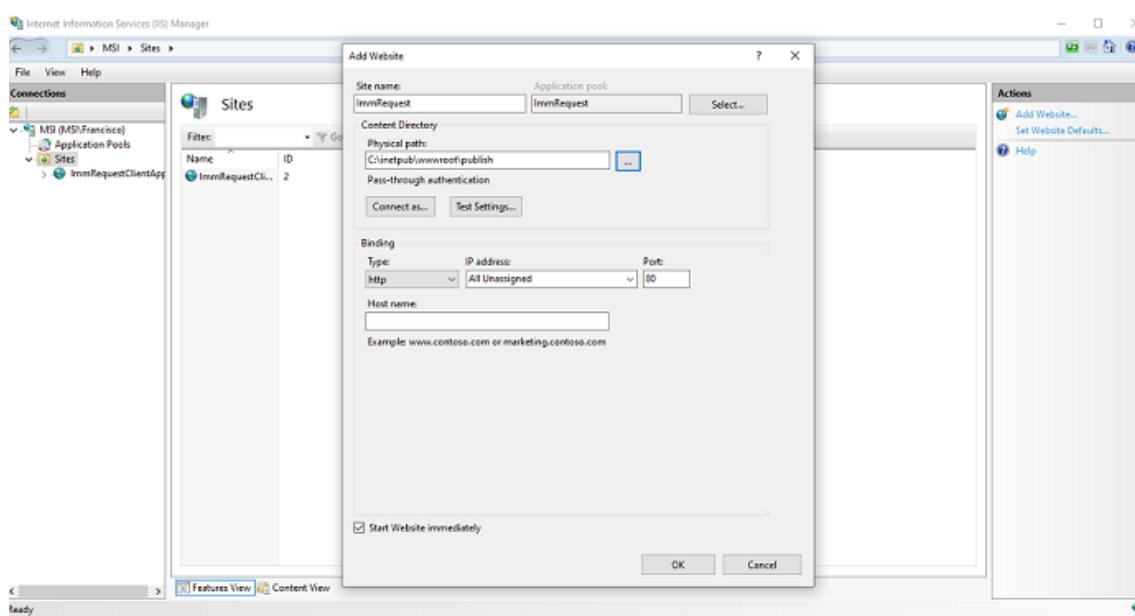


Figura E.5: Creando sitio en IIS.

7. Restauramos nuestro backup para cargar la base (pegando el archivo .bak)

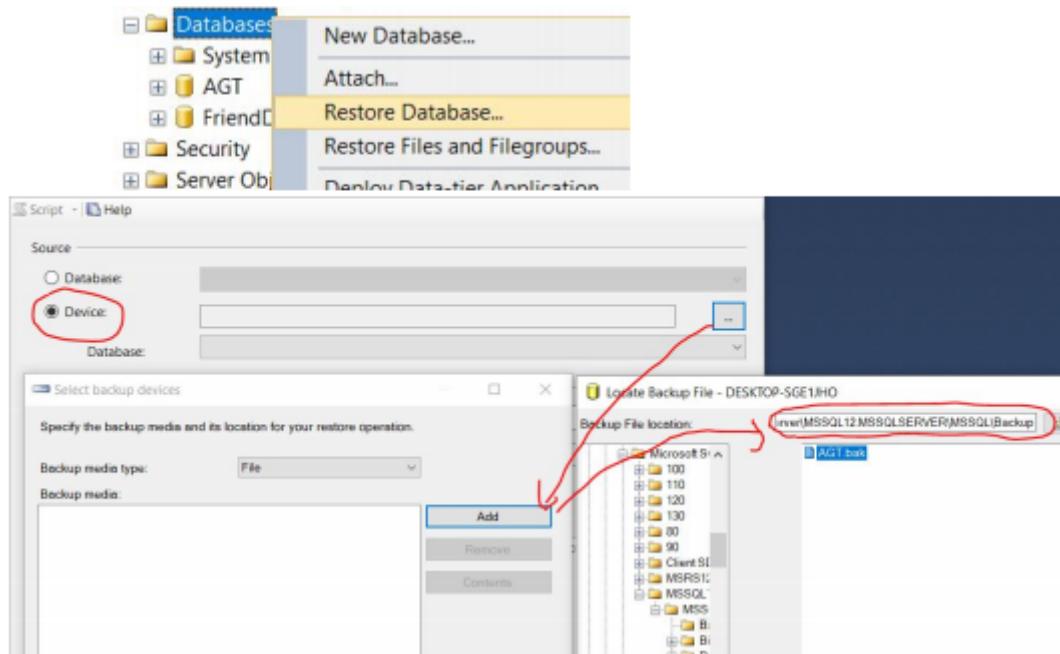


Figura E.6: Restauración backup.

8. Por último, creamos y otorgamos permisos al usuario de IIS sobre la base de datos. El Login Name corresponde a “IIS APPPOOL{NOMBRE DEL GRUPO DE APLICACIONES DE IIS}”

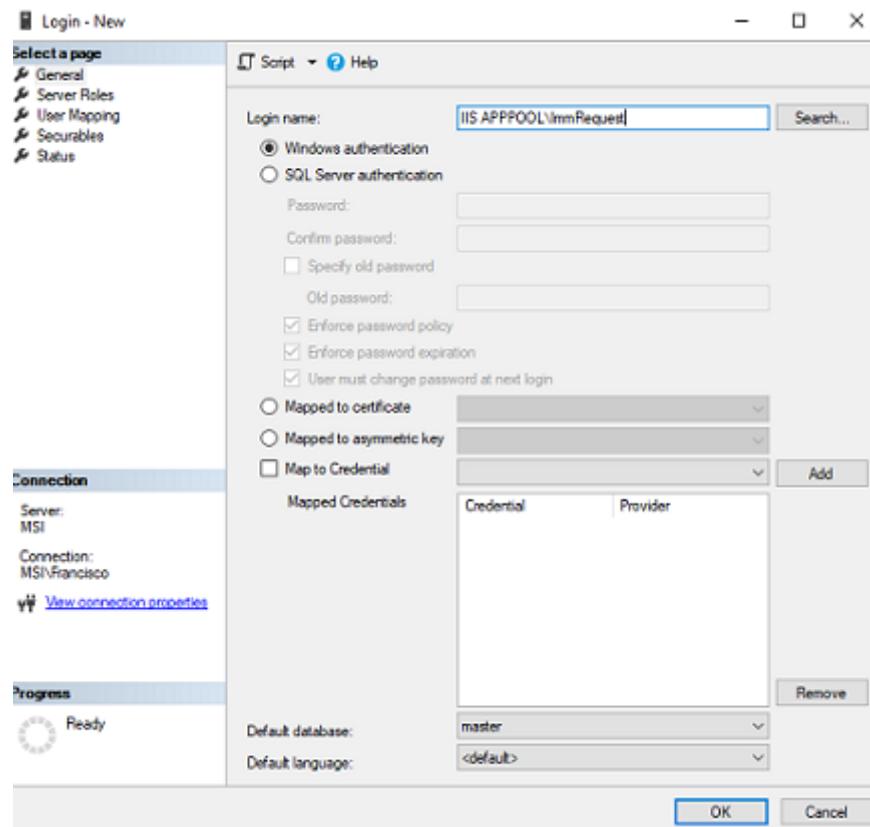


Figura E.7: Creando login.

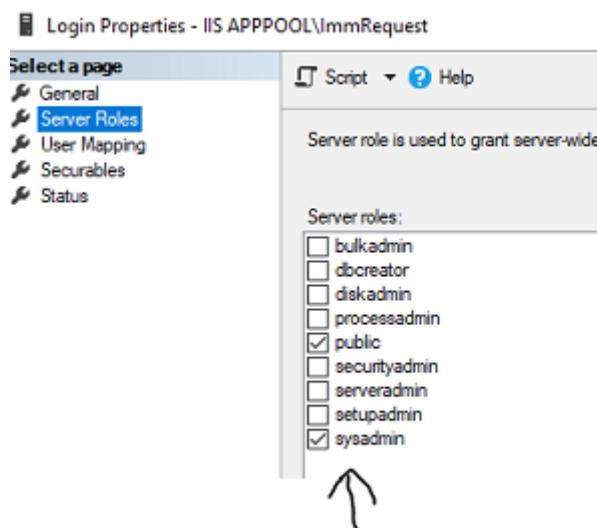
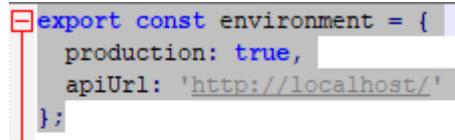


Figura E.8: Otorgando permisos sysadmin.

E.2. Deploy Lado Cliente

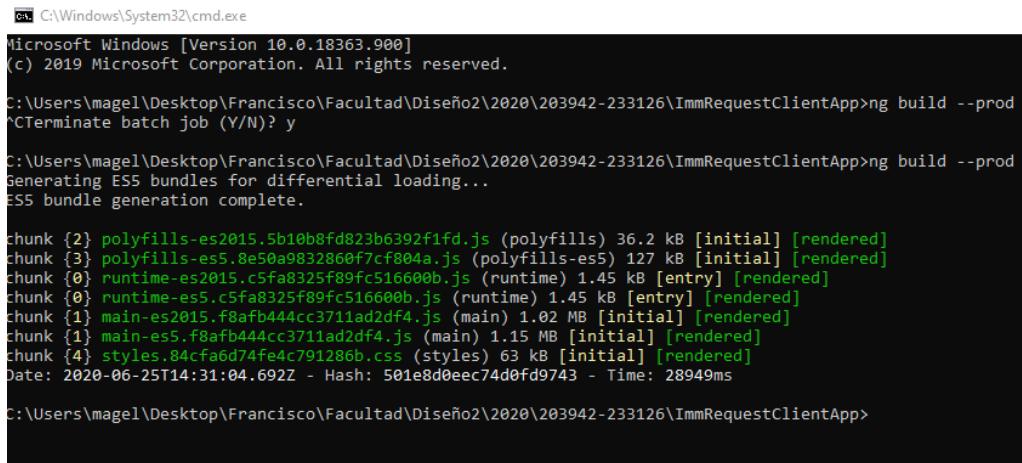
1. En la carpeta donde se encuentra el archivo `environment.prod.ts` (en nuestro caso `{raiz del proyecto}/ImmRequestClientApp/src/environments`), colocamos en el campo `apiUrl` la dirección del servidor deployeado en la sección anterior:



```
export const environment = {
  production: true,
  apiUrl: 'http://localhost/'
};
```

Figura E.9: environment.prod.ts

2. En la carpeta donde esta el proyecto del cliente (en nuestro caso `{raiz del proyecto}/ImmRequestClientApp`), ejecutamos el siguiente comando:
`ng build --prod`



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\magel\Desktop\Francisco\Facultad\Diseño2\2020\203942-233126\ImmRequestClientApp>ng build --prod
^CTerminate batch job (Y/N)? y

C:\Users\magel\Desktop\Francisco\Facultad\Diseño2\2020\203942-233126\ImmRequestClientApp>ng build --prod
Generating ES5 bundles for differential loading...
ES5 bundle generation complete.

chunk {2} polyfills-es2015.5b10b8fd823b6392f1fd.js (polyfills) 36.2 kB [initial] [rendered]
chunk {3} polyfills-es5.8e50a9832860f7cf804a.js (polyfills-es5) 127 kB [initial] [rendered]
chunk {0} runtime-es2015.c5fa8325f89fc516600b.js (runtime) 1.45 kB [entry] [rendered]
chunk {0} runtime-es5.c5fa8325f89fc516600b.js (runtime) 1.45 kB [entry] [rendered]
chunk {1} main-es2015.f8afb444cc3711ad2df4.js (main) 1.02 MB [initial] [rendered]
chunk {1} main-es5.f8afb444cc3711ad2df4.js (main) 1.15 MB [initial] [rendered]
chunk {4} styles.84cfa6d74fe4c791286b.css (styles) 63 kB [initial] [rendered]
Date: 2020-06-25T14:31:04.692Z - Hash: 501e8d0eec74d0fd9743 - Time: 28949ms

C:\Users\magel\Desktop\Francisco\Facultad\Diseño2\2020\203942-233126\ImmRequestClientApp>
```

Figura E.10: Output de la ejecución en línea de comandos

3. Luego de ejecutado el comando, copiamos la carpeta generada por el build de angular (en nuestro caso, de nombre `ImmRequestClientApp` localizada en `{raiz del proyecto}/ImmRequestClientApp/dist`) a la carpeta `C:/inetpub/wwwroot`

4. Para finalizar, creamos el sitio en IIS (tiene que ser con puerto distinto al puerto elegido para el servidor). En nuestro caso elegimos el puerto 8888:

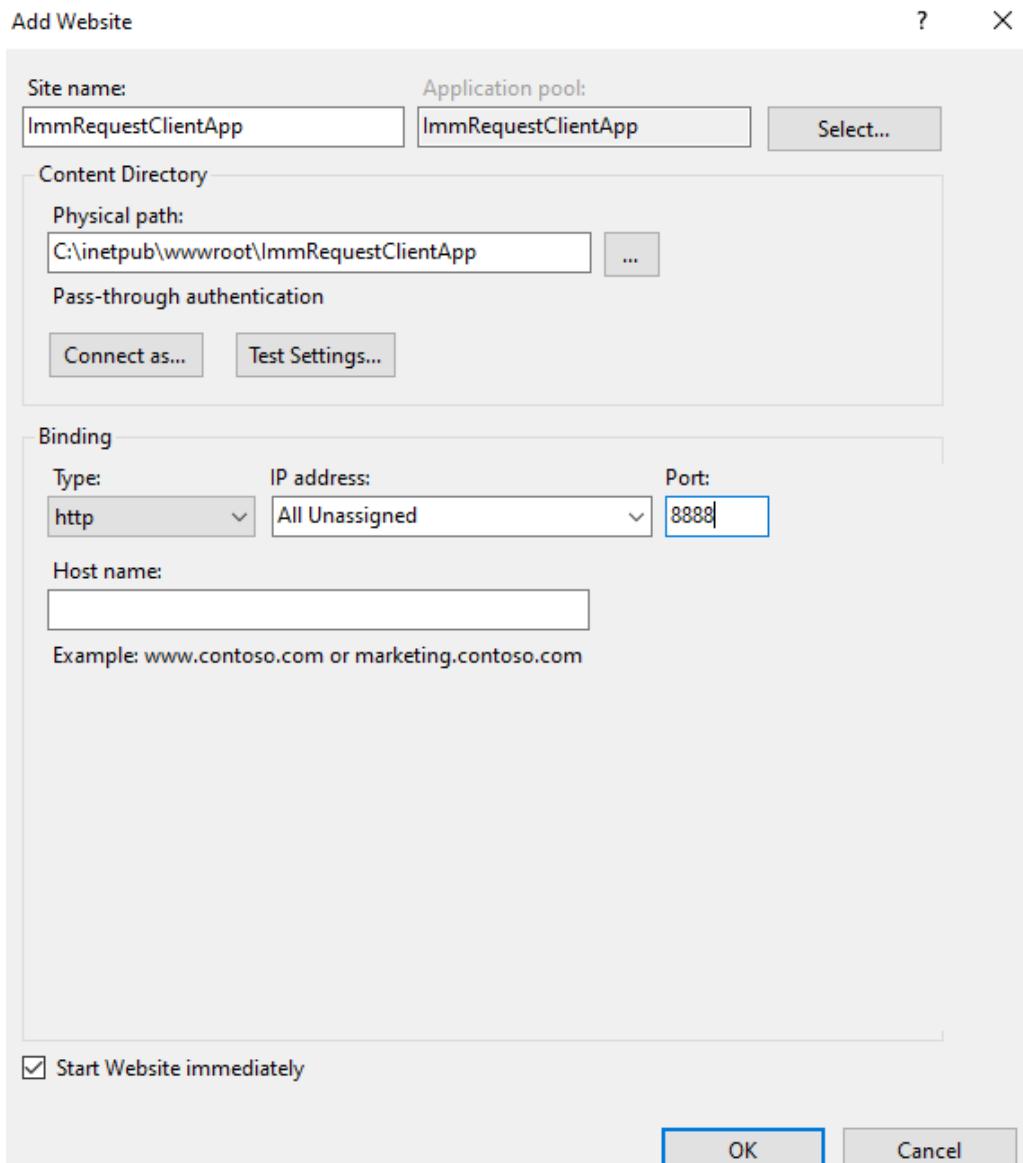


Figura E.11: Output de la ejecución en línea de comandos

Bibliografía

- [1] T. Dykstra. (2013, jul) Implementing the repository and unit of work patterns in an asp.net mvc application (9 of 10). [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>
- [2] Z. P. LTD. Ndepend. [Online]. Available: <https://www.ndepend.com/docs/getting-started-with-ndepend>
- [3] M. Fowler. (2005, mar) Test driven development. [Online]. Available: <https://martinfowler.com/bliki/TestDrivenDevelopment.html>
- [4] R. C. Martin, *Clean Code*, 1st ed. Prentice Hall, aug 2008.
- [5] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: <http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf>
- [6] Microsoft Azure. (2018) API design. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design>
- [7] Richard Morris. (2020) Swagger tools for documenting API's built on ASP.NET Core. [Online]. Available: <https://github.com/domaindrivendev/Swashbuckle.AspNetCore>
- [8] T. GIbb. (2017, jul) How to install iis on windows 8 or windows 10. [Online]. Available: <https://www.howtogeek.com/112455/how-to-install-iis-8-on-windows-8/>