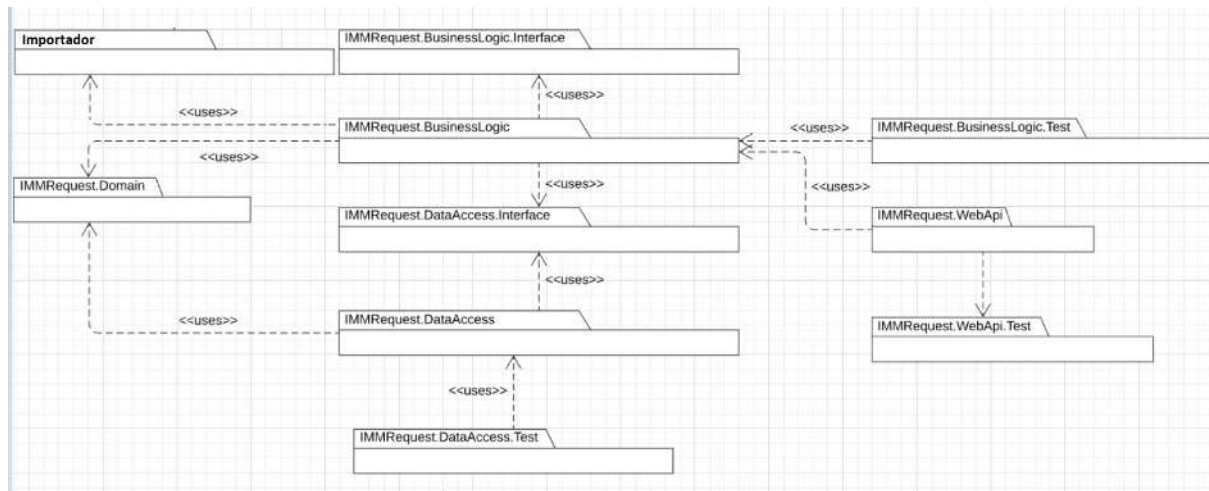


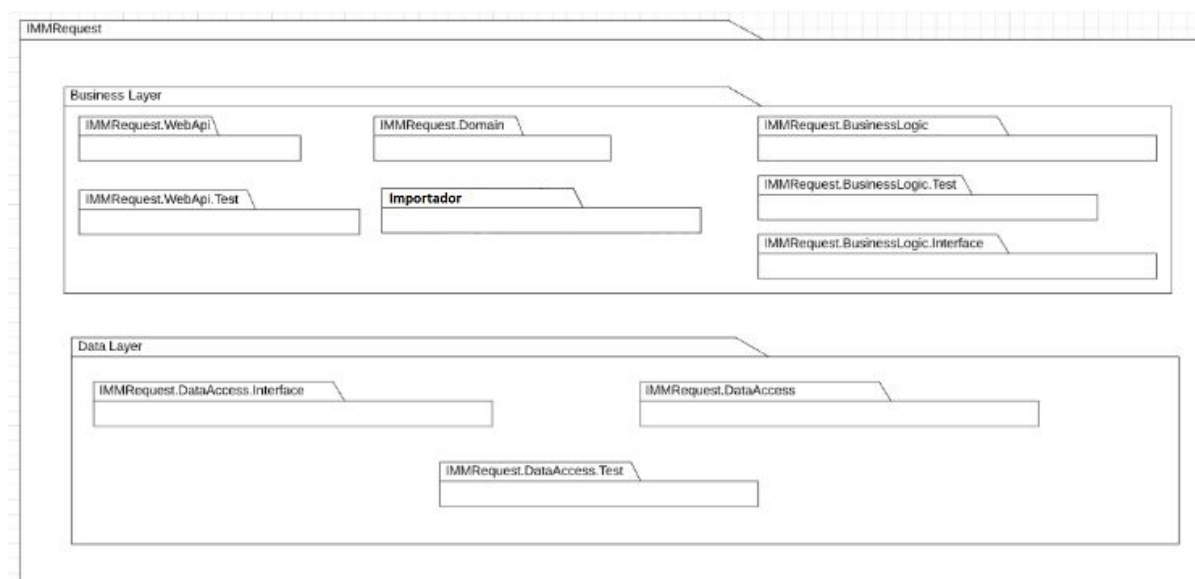
Resumen

Se desarrolló una aplicación Web Angular la cual permite a los ciudadanos de Montevideo poder hacer denuncias o informar diferentes situaciones que ocurren en la ciudad para que la intendencia pueda reaccionar más rápidamente. La misma funciona permitiendo ingresar solicitudes con un formato pre-definido, pudiendo agregar nueva información, actualizar datos, eliminarlos o obtener datos ingresados previamente. También para los Administradores del sistema, permite modificar estados de las solicitudes, administrar los tipos, administrar otros administradores, generar reportes, importar datos. El presente documento detalla todo el proceso para la creación de la aplicación, así como las herramientas y técnicas usadas para asegurar la calidad del proceso de desarrollo. El sistema tiene un error conocido que es con los campos adicionales, si bien al sistema se le pueden agregar Tipos, al hacer la asignación con los campos adicionales, los mismos no se cargan en el sistema.

1. Diagrama de descomposición de los namespaces del proyecto



2. Diagrama general de paquetes mostrando los paquetes organizados por capas y sus dependencias.

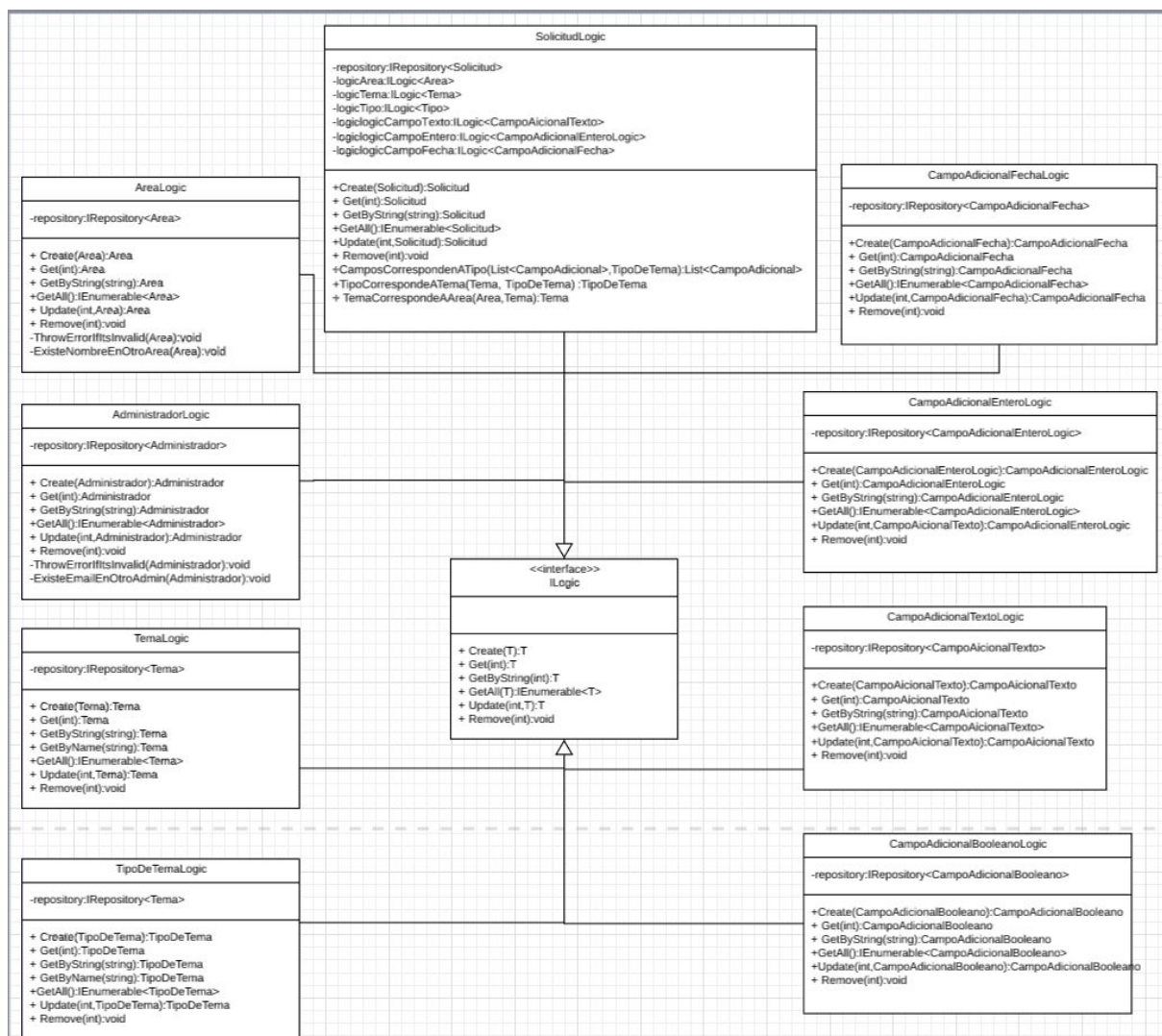


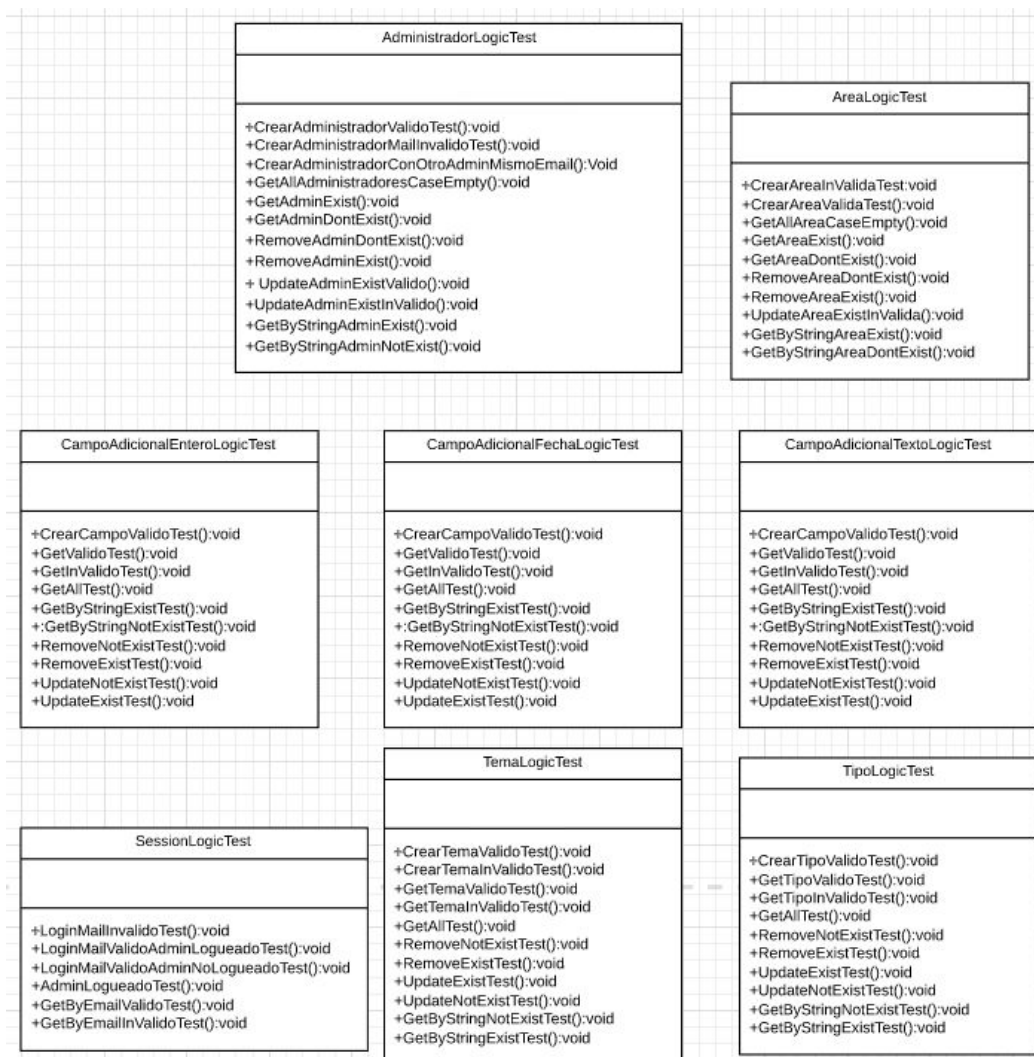
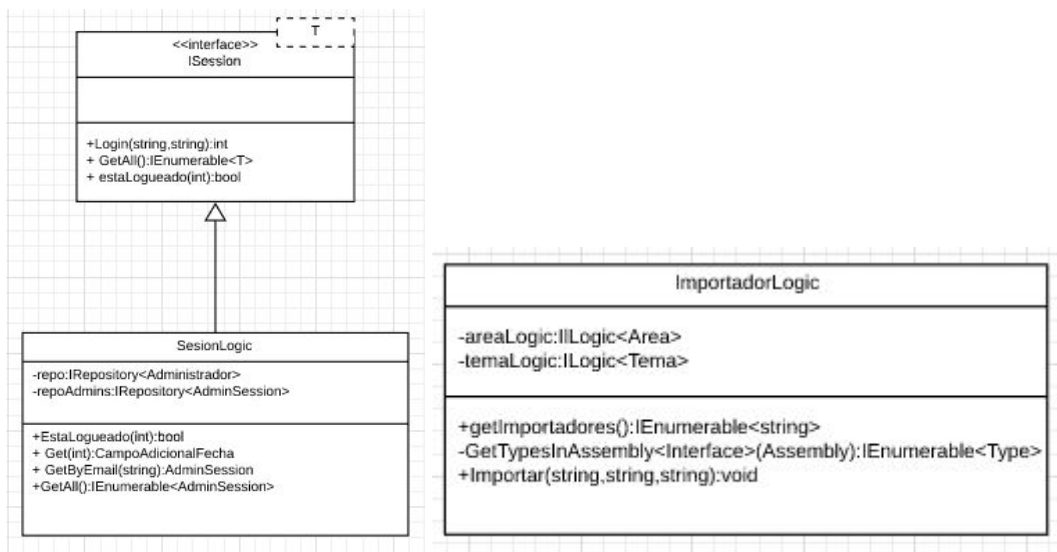
Descripción de Paquetes

- **Business Logic:**

El paquete de Business Logic contiene toda la lógica de la aplicación, esta misma se encarga de interactuar con la capa de Acceso a datos, y es utilizada por la web api para tratar la información que se le pasa y tiene que ser tratada. El paquete de Business Logic.Interface contiene todas las interfaces que se van a usar en business Logic, estas son ILogic y ISession. El paquete Business Logic.Test es donde se van a poner todos los Test automáticos para las diferentes clases de BusinessLogic.

Diagrama de Clases:

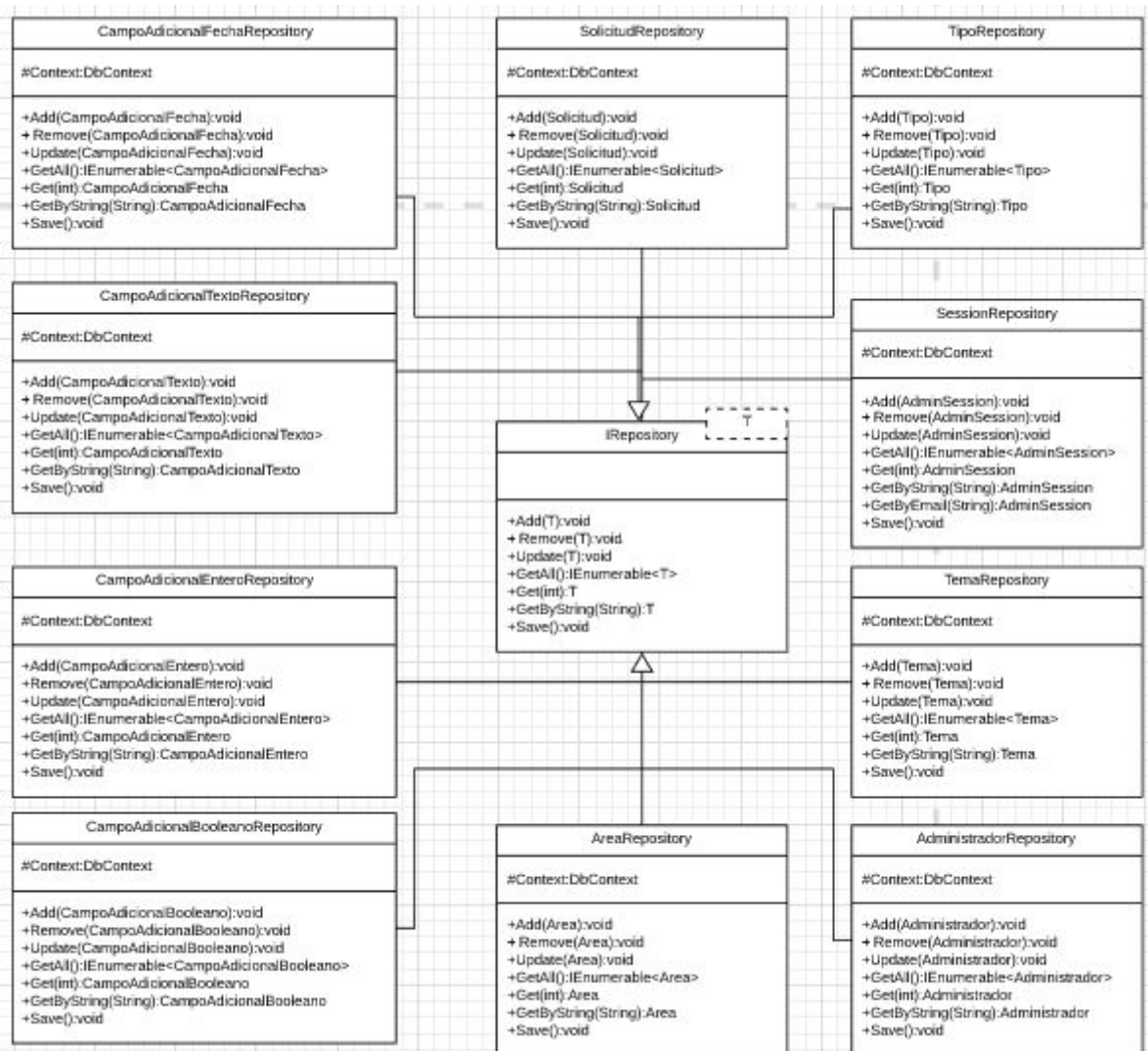




- **DataAccess:**

El paquete DataAccess contiene la lógica para interactuar con la base de Datos de la aplicación, contiene los diferentes Repositorios para las diferentes clases, Contiene los contextos de los diferentes DBSet Necesarios y el contextFactory. El paquete DataAccess.Interface contiene la interfaz para implementar los diferentes repositorios(IRepository). El paquete DataAccess.Test contiene los test automáticos para los diferentes Repositorios.

Diagrama de Clases:



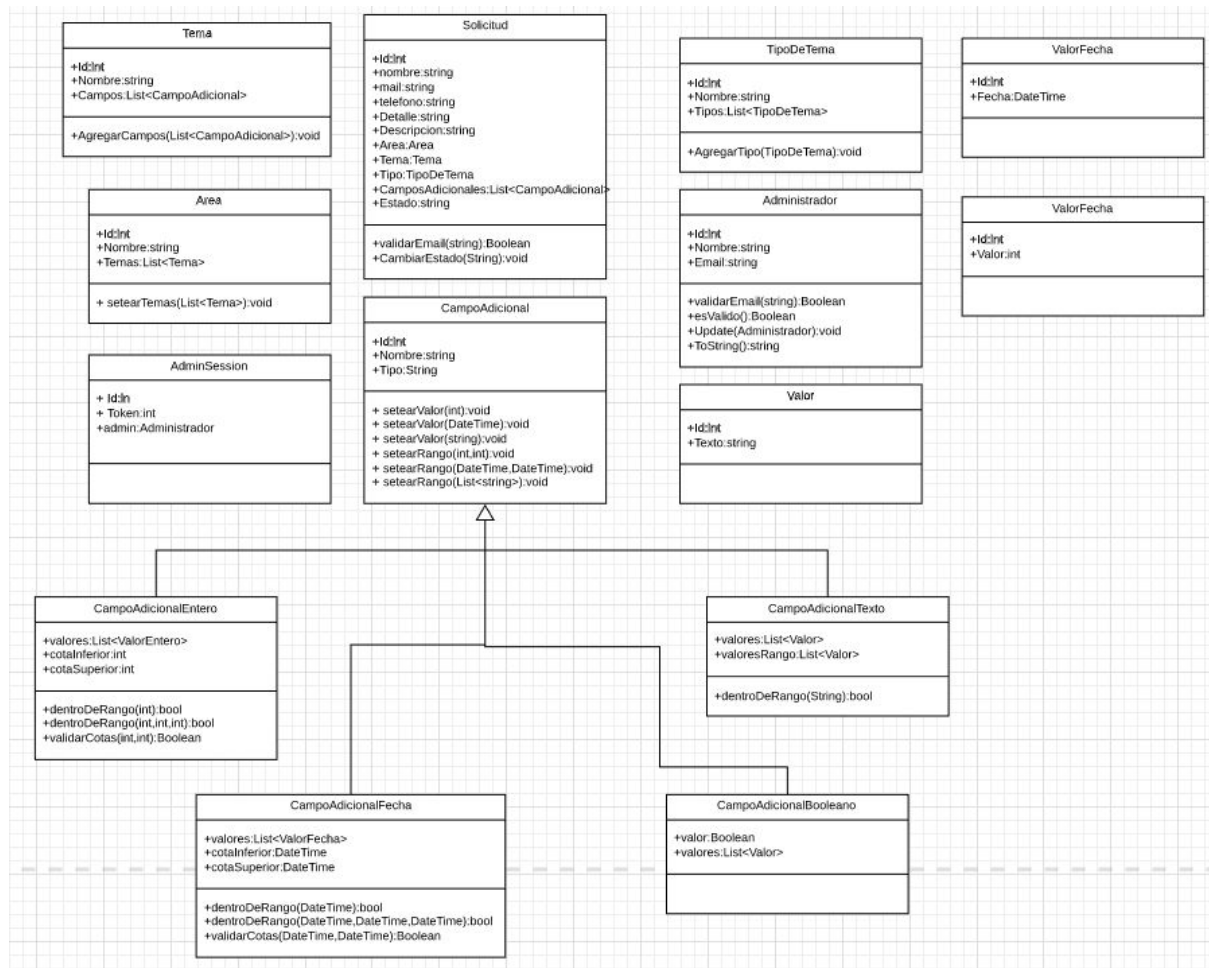
ContextFactory
<u>+ GetMemoryContext(string) :IMMRequestContext</u> <u>+GetMemoryConfig(DbContextOptionsBuilder, string) :DbContextOptions</u> <u>+GetSqlContext():IMMRequestContext</u> <u>+GetSqlConfig(DbContextOptionsBuilder):DbContextOptions</u>

AdministradorRepositoryTest	AreaRepositoryTest	SessionRepositoryTest
+AddAdmin:Void +RemoveAdminExist:Void +RemoveAdminNotExist:Void +UpdateAdminExist:Void +UpdateAdminNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void	+AddArea:Void +RemoveAreaExist:Void +RemoveAreaNotExist:Void +UpdateAreaExist:Void +UpdateAreaNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void	+AddSession:Void +RemoveSessionExist:Void +RemoveSessionNotExist:Void +UpdateSessionExist:Void +UpdateSessionNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void
CampoAdicionalEnteroRepositoryTest	CampoAdicionalFechaRepositoryTest	CampoAdicionalTextoRepositoryTest
+AddCampo:Void +RemoveCampoExist:Void +RemoveCampoNotExist:Void +UpdateCampoExist:Void +UpdateCampoNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void	+AddCampo:Void +RemoveCampoExist:Void +RemoveCampoNotExist:Void +UpdateCampoExist:Void +UpdateCampoNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void	+AddCampo:Void +RemoveCampoExist:Void +RemoveCampoNotExist:Void +UpdateCampoExist:Void +UpdateCampoNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void
SolicitudRepositoryTest	TemaRepositoryTest	TipoRepositoryTest
+AddSolicitud:Void +RemoveSolicitudExist:Void +RemoveSolicitudNotExist:Void +UpdateSolicitudExist:Void +UpdateSolicitudNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void	+AddTema:Void +RemoveTemaExist:Void +RemoveTemaNotExist:Void +UpdateTemaExist:Void +UpdateTemaNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void	+AddTipo:Void +RemoveTipoExist:Void +RemoveTipoNotExist:Void +UpdateTipoExist:Void +UpdateTipoNotExist:Void +GetAll():void +GetByIdExist:Void +GetByIdNotExist:Void +GetByStringExist:Void +GetByStringNotExist:Void

- **Domain:**

El paquete Domain contiene las clases del dominio del sistema. Estas son Administrador, AdminSession, Area, Tema, TipoDeTema, CamposAdicionales y Solicitud.

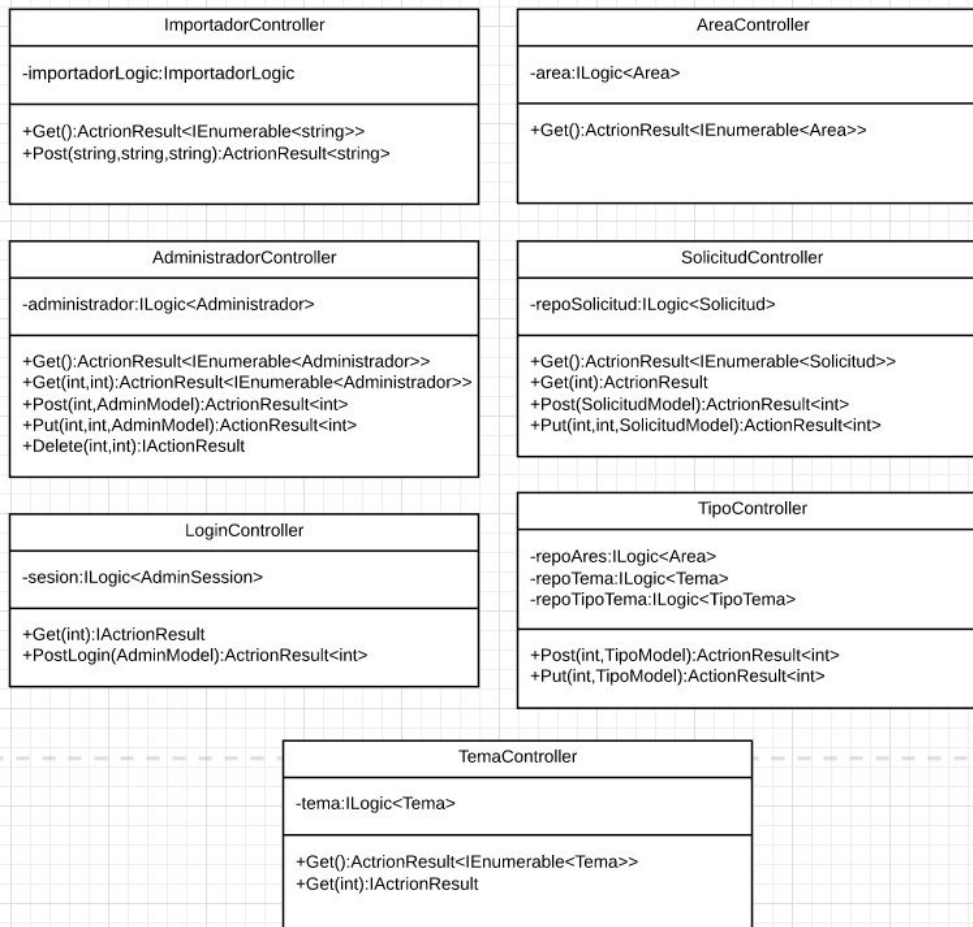
Diagrama de Clases:

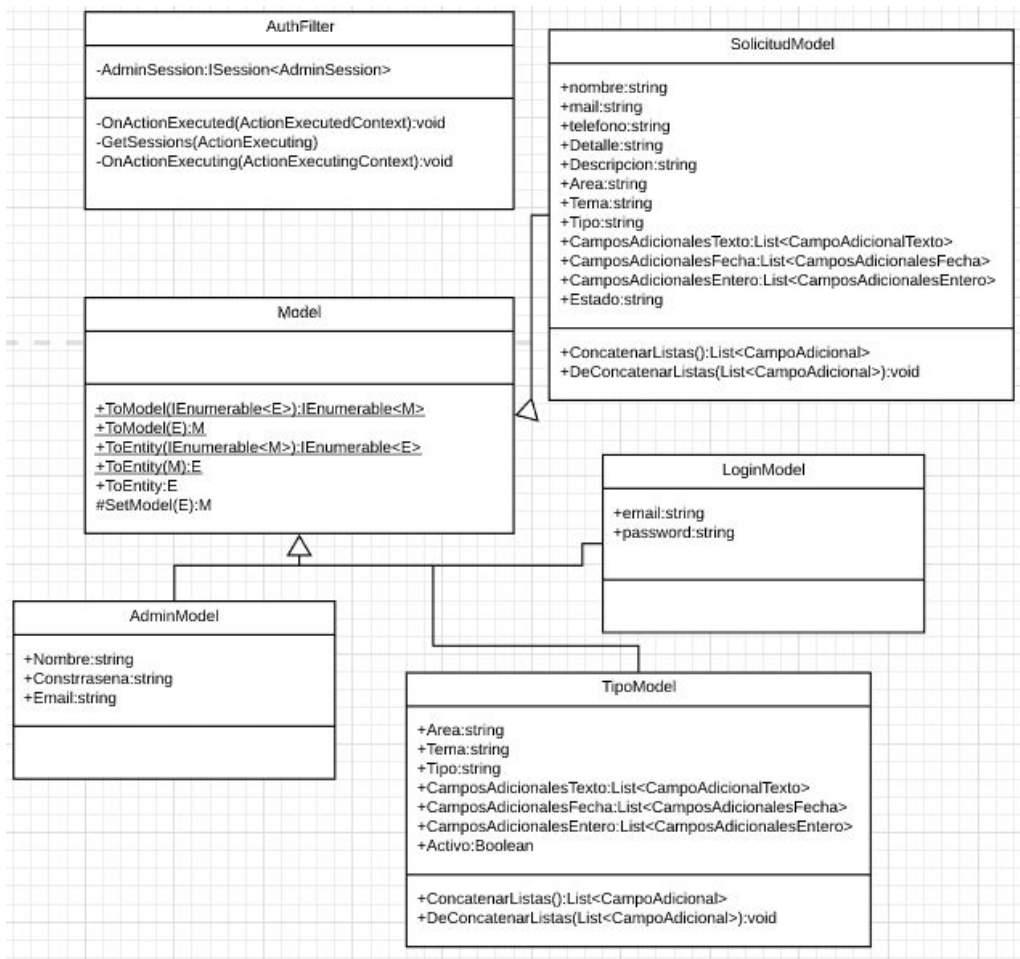


- **WebApi:**

El Paquete WebApi contiene todo para implementar la Api Rest, contiene los controllers, los Models, el filtro de Autenticacion, y propiedades de la Api. El paquete WebApi.Test contiene los Test Automáticos que se corren sobre la WebApi.

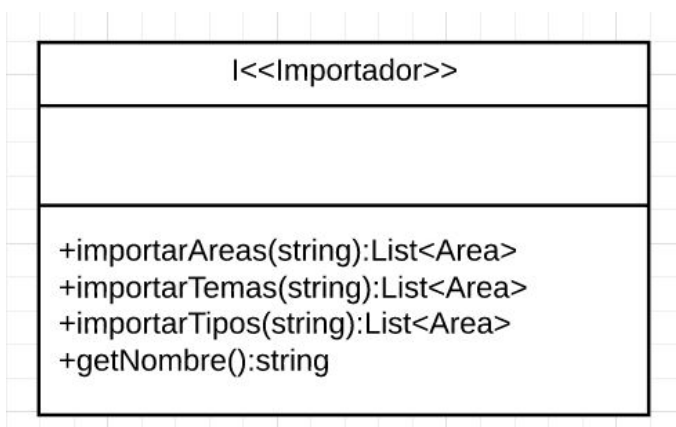
Diagrama de Clases:





- Importador

Este Paquete se encarga de brindar la interfaz de importador, para así poder ser utilizada por otros desarrolladores y mediante Reflection, poder ser utilizada por la aplicación.



Las jerarquías utilizadas dependen del tipo, para las lógicas, se utilizó una interfaz llamada ILogic, esta interfaz tiene un tipo genérico T, con elementos de ese tipo y al momento de hacer una implementación de la interfaz se debe poner de qué tipo será. Para la lógica de sesión, se utilizó una interfaz llamada ISession con un tipo genérico T de la misma forma que con ILogic. Para el acceso a datos, se utilizó una interfaz llamada IRepository, esta tiene un tipo genérico T y tiene todas las funciones que se van a ejecutar con el acceso a datos en base de datos.

Todas las clases que implementen las interfaces van a tener que definir de qué tipo son, por ejemplo la lógica de sesión de administrador va a tener que definir como una ISession de Administrador y así para todas las implementaciones.

Modelo de tablas de estructura de la base de datos

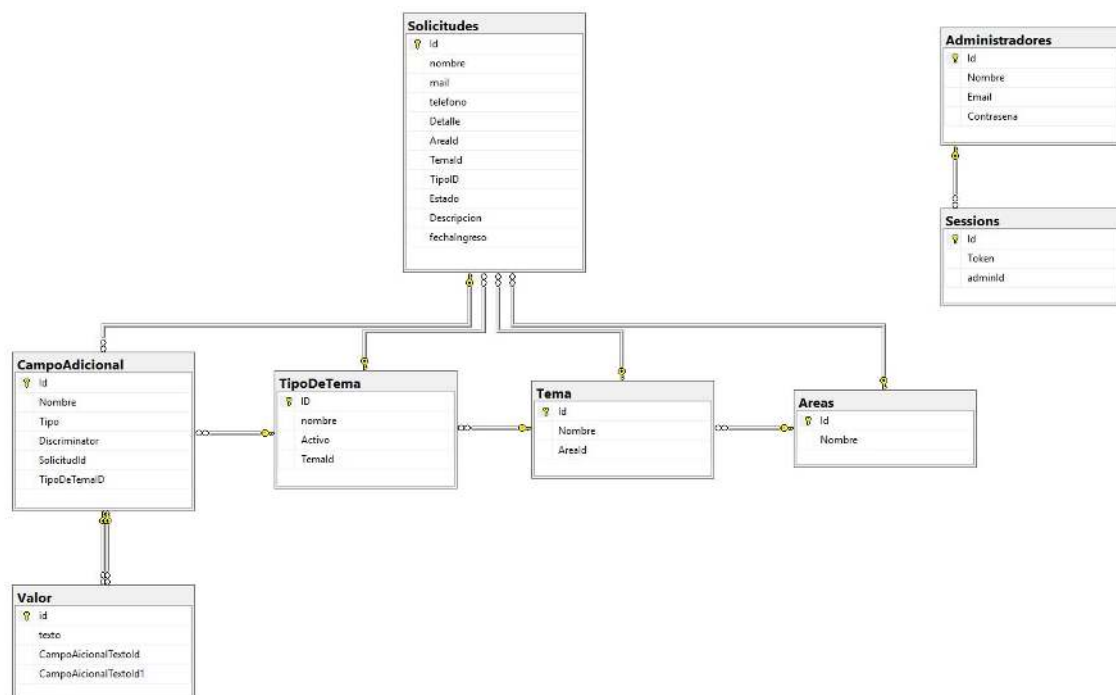
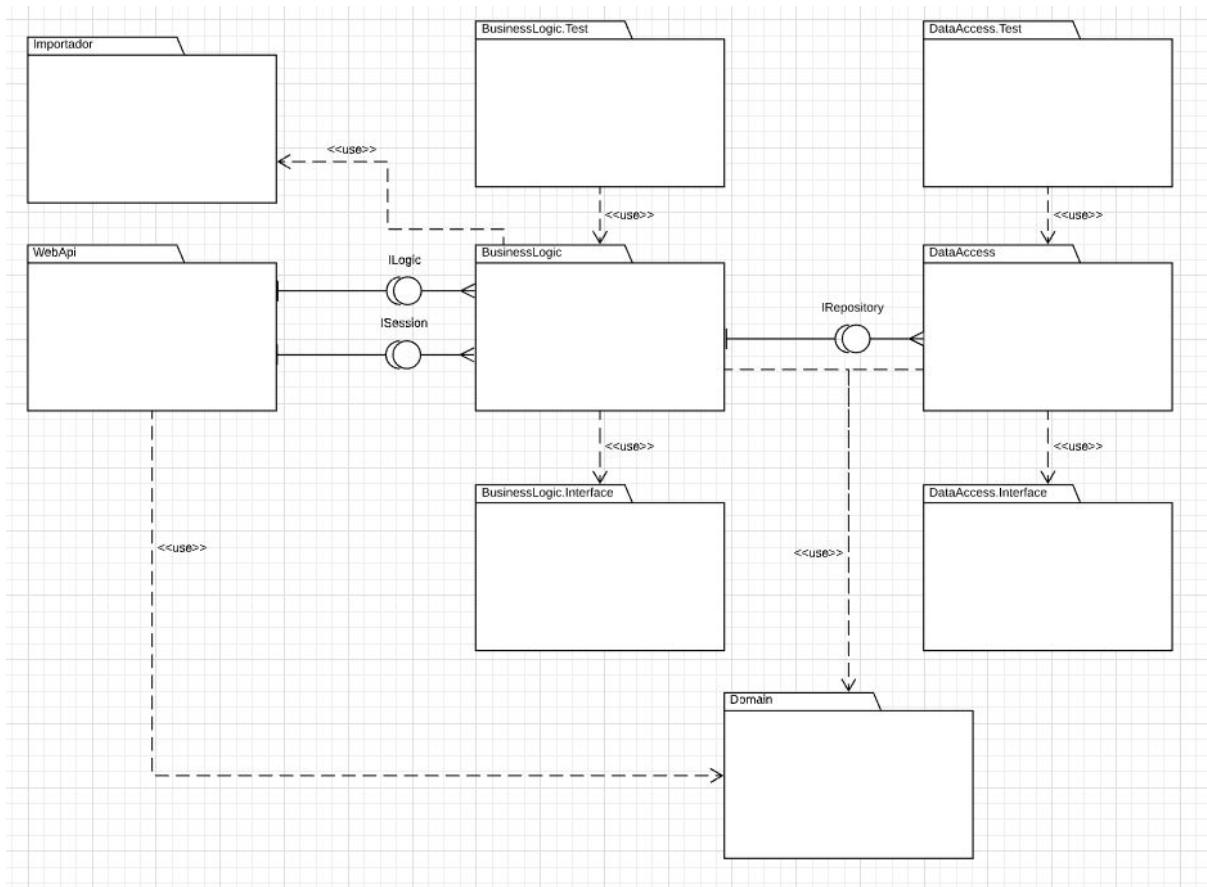


Diagrama de implementación



La solución se dividió en los componentes mostrados arriba para lograr una separación en capas y una mejor mantenibilidad, ya que si en el futuro queremos cambiar la lógica de negocio, con crear las implementaciones `ILogic` y `ISession` y cambiar el scoped estaría pronto.

Justificación del diseño

Se utilizó inyección de dependencias para así poder el acople entre capas y que no sean tan dependientes entre sí, haciendo que una clase que utiliza a otra, en vez de estar instanciando esa clase, la recibe por parámetros. Esto se logra en el Startup de la API, agregando servicios de Scope para así, cuando una clase precisa de otra por parámetros, el sistema automáticamente le pasa la implementación que se desea. Esto logra que en el caso de querer cambiar una lógica de negocio, solamente tenemos que implementar esta misma, que en el caso concreto de este proyecto es implementar una interfaz, y mientras esta nueva implementación satisfaga la interfaz se va a poder sustituir e inyectarla. Los beneficios de inyección de dependencia son que el código queda más limpio, la mantenibilidad del sistema es mejor y más fácil.

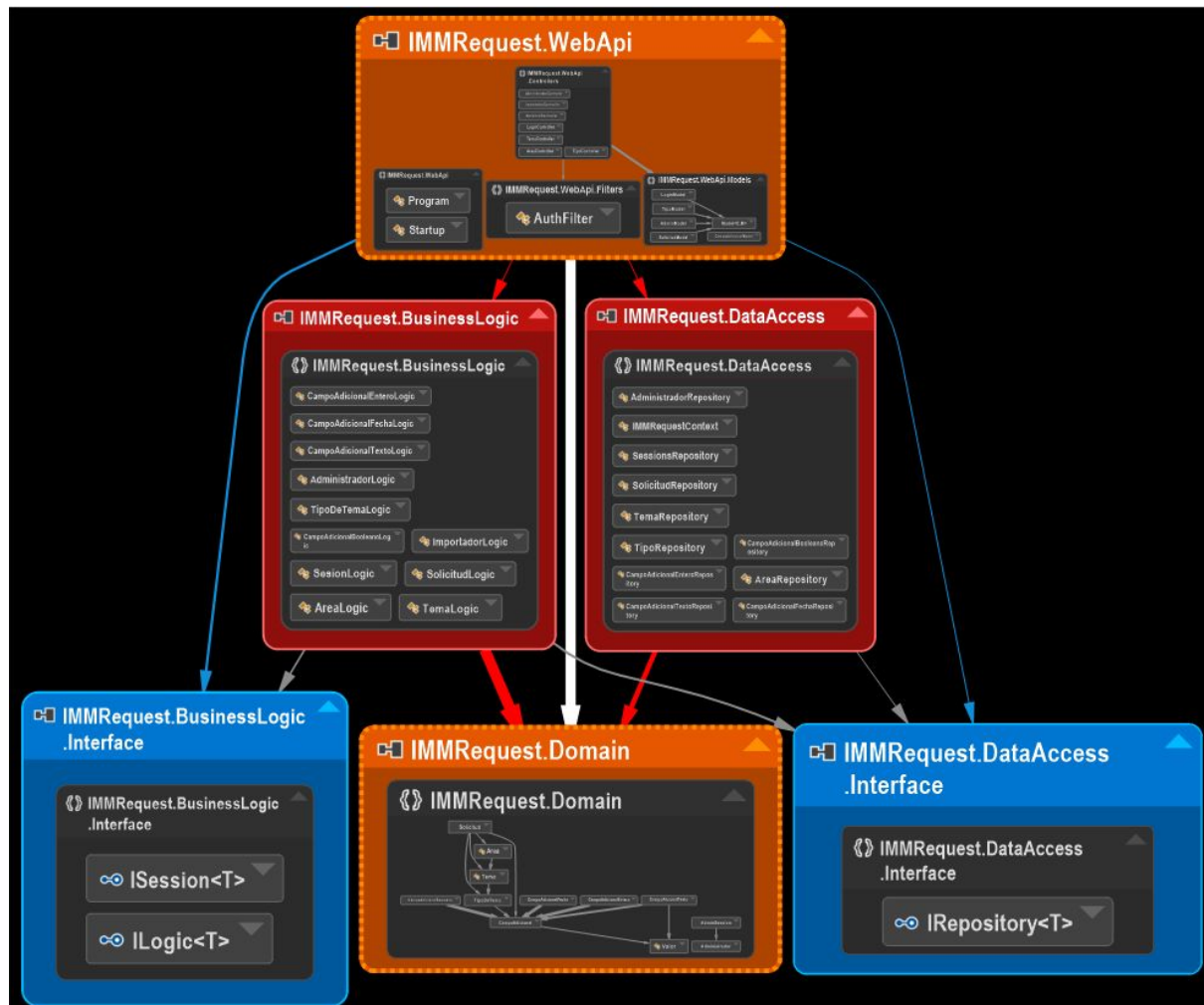
Intentamos seguir los principios SOLID, en este caso, con la inyección de dependencia logramos no violar SRP y OCP, ya que los módulos usan sus dependencias pero no se encargan de inicializarlas y el código es abierto a la extensión y cerrado a la modificación.

Los patrones de diseño y principios de diseño utilizados fueron:

1. Factory Method: este patrón se utilizó para los campos adicionales, al tener una interfaz campoAdicional, y cada implementación, también se utilizó para la lógica de negocio con ILogic e ISession.
2. REP: este principio se utiliza en todo el código, así facilitamos el reuso del código de la aplicación.
3. ADP: como podemos ver en el grafo de dependencia ingresado más adelante, no se encuentra ninguna dependencia cíclica, por lo que se cumple con este principio a nivel de acoplamiento.

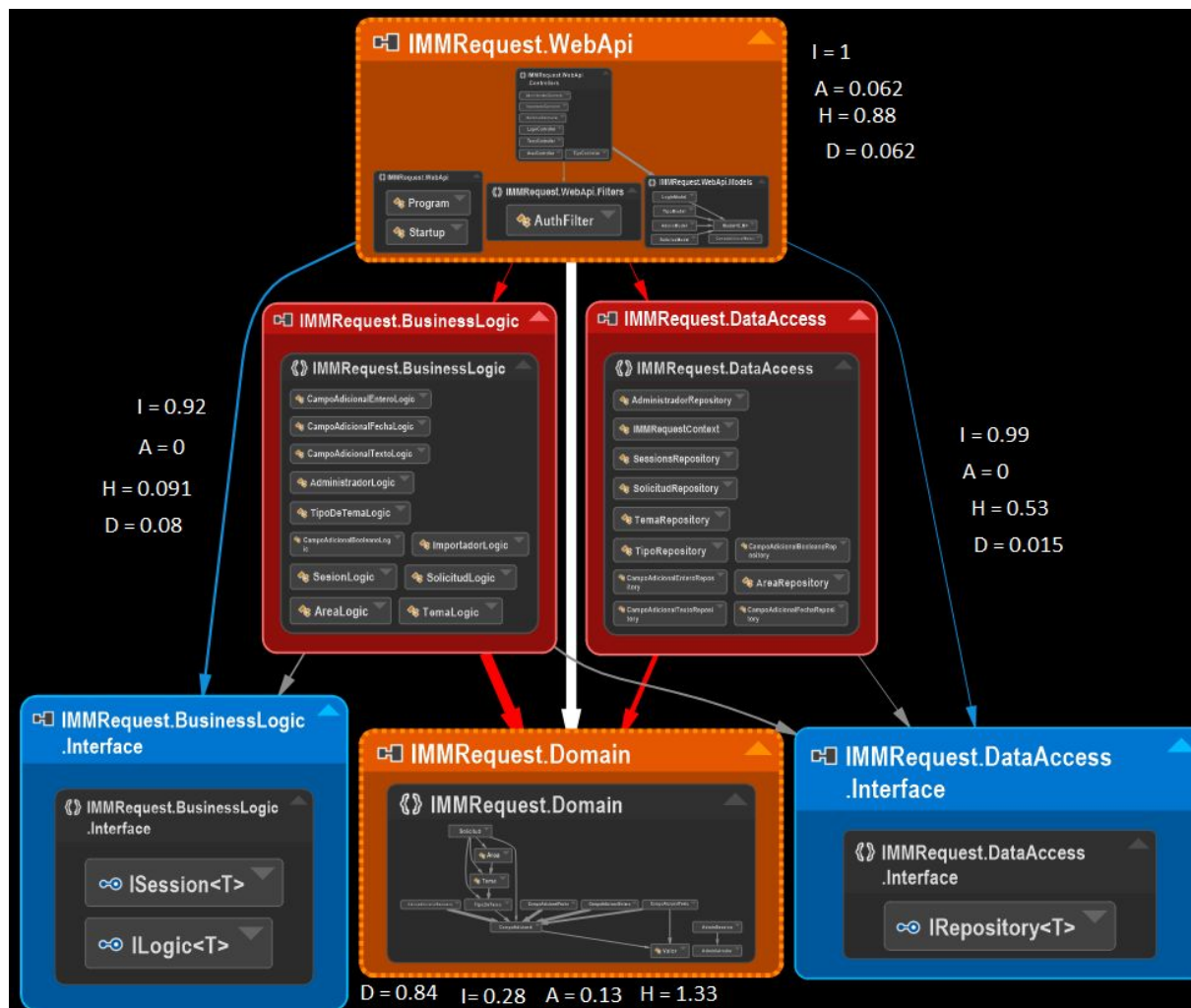
Grafo de Dependencias.

En el mismo notamos que no hay ninguna dependencia cíclica.



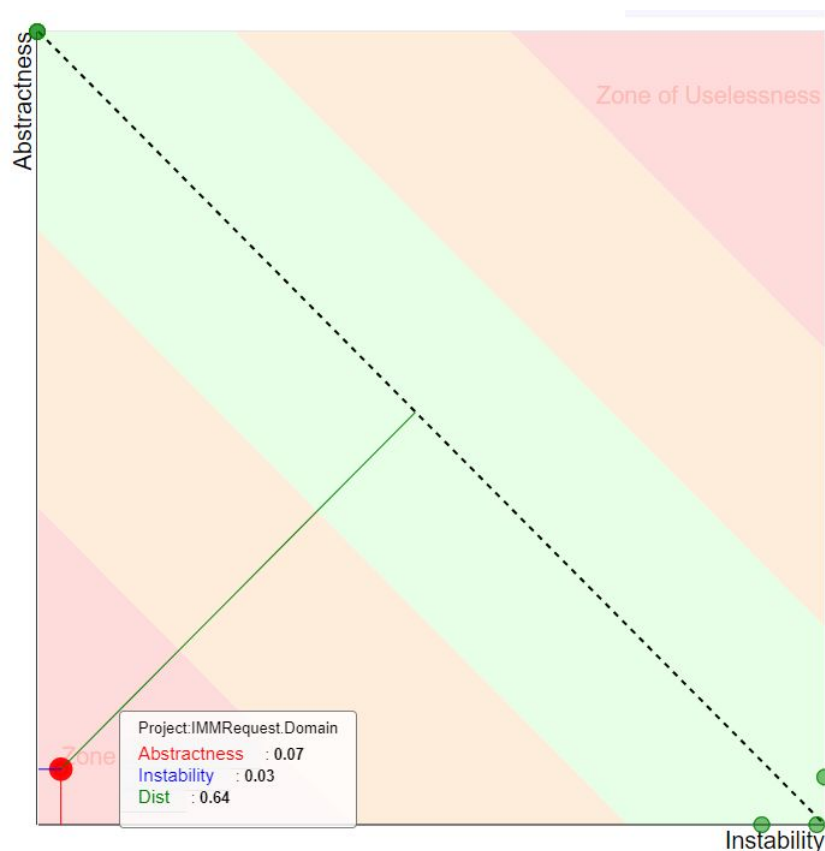
(Para poder verlo mejor, se agrega en el anexo la imagen con calidad completa para poder hacer zoom)

Calidad De Diseño:



Vemos que WebApi, BusinessLogic y DataAccess cumple con el principio de Abstracciones estables, ya que, todos tienden a una Inestabilidad de 1 e una Abstracción de 0, sin embargo Domain, Su inestabilidad es Bastante baja, pero su abstracción es bastante baja, para cumplir con el principio, tendria que tener una Abstracción más alta.

Las dependencias vemos que son todas de paquetes Inestables a paquetes más estables, por lo que se cumple con el Principio de Dependencias estables.



Vemos que, Domain es el único que está en una zona problemática, está en la zona de dolor, este paquete es concreto y estable, pero tiene el problema que no es extensible y en el caso de hacer algún cambio, estos cambios impactan en otros paquetes que dependen de él.

Las distancias del resto de los paquetes a la secuencia principal son muy bajas, entre 0.08 y 0.015, por lo que los paquetes o son abstractos y Estables, o son Inestables y concretos.

El principio de reuso común es el que se utiliza en este momento de la implementación del proyecto, ya que, al ser de las últimas etapas de la implementación, y la arquitectura está bastante estable, queremos favorecer el reuso.

Instrucciones para crear Importadores:

Documentacion para crear Nuevos Importadores en el sistema.

1. En la carpeta `\IMMRequest.WebApi\bin\Release\netcoreapp3.1` situada en la raíz del proyecto se debe copiar el archivo `Importador.dll`, en la nueva solución se debe referencia a dicho dll, e implementar todos los métodos que la interfaz importador tiene definido.
Estos son: `importarAreas()`, `importarTemas()`, `importarTipos()`, `getNombre()`
Importante, el método `getNombre()` tiene que retornar el nombre del importador que se quiere que aparezca en la app, y la aplicación, al ser llamada va a buscar la implementación que tiene ese nombre ingresado.
2. Una vez implementado todo, se debe generar el .dll de nuestra implementación, y el mismo se debe ingresar en la carpeta `\IMMRequest.WebApi\Importadores` para que la aplicación pueda encontrarlo.
3. al importar Areas, se debe retornar una lista de Areas nuevas, si se incluye una que ya existe en el sistema, la importación falla.
4. Al importar Temas, se retorna una lista de Area, estas areas tienen que existir en el sistema previamente, si no existen el sistema falla.
4. Al importar Tipos, se retorna una Lista de Area, que cada una tiene que contener Áreas y Temas Ya existentes en el sistema, si no existen las areas o los Tipos, e la importación falla

ANEXO

Anexo Imagenes Front End

1. Reporte A

Email Cliente

diego@diego.com

Filtrado por el Cliente : diego@diego.com

☒ OFF/ON

Encendido

Desde

6/1/2020

Hasta

6/30/2020

Crear Reporte

- Creada(2) = [2022,2024]

- En revisión(1) = [1022]

2. Reporte B

Email Cliente

Filtrado por el Cliente :

☐ OFF/ON

Apagado

Desde

6/1/2020



Hasta

6/30/2020



Crear Reporte

- (6)

- test(3)

- pedro5(1)

- pedro6(1)

- pedro4(1)

3. Importación

IMMRequest application

MenuLogin

Ruta de archivo a exportar

Que Importo?

Areas

Lista de Importadores?

Importar

4. Ingreso Nueva Solicitud

IMMRequest application

MenuLogin

Buscar Solicitud

Buscar solicitud por Numero de Solicitus

Agregar Solicitud

Agregar Nueva Solicitud

Nombre

Mail

Telefono

Area

Tema

Tipo

Detalle

5. Modificación de un Administrador

IMMRequest application Menu Login

Id	Nombre	Email	Password
1028	diego	diego@diego.com	diego
2029	admin	admin@admin.com	adminn

Nombre: admin

Mail: admin@admin.com

Password: adminn

Nombre

admin

Mail

admin@admin.com

Password

adminn

Modificar

Eliminar

6. Alta de Tipo de solicitud con campos Adicionales

Agregar Tipo

Agregar Tipo

Area

Tema

Nombre

Activo

Campos Adicionales

Campo Adicional Texto

Nombre

Valor campo

Valores

Agregar

Campo Adicional Fecha

Campo Adicional Entero

Campo Adicional Booleano

Cantidad de Campos Adicionales: 0

Agregar Tipo

7. Listado de solicitudes

IMMRequest application				Menu	Login
Id	Email	Nombre	Estado	Descripcion	
1013	dsa@dsa.com	rwerew	Creada		
1014	dsa@d.sa	dsada	Creada		
1015	diego@diego.com	diego	Aceptada	jhfgj	
1016	diego@diego.com	diegodsada	Creada		
1017	diego@diego.com	diegodsada	En revisión	nghfg	
1018	fsd@s.c	fsdfs	Creada		
1019	gfs@d.gdf	fsdfsd	Creada		
1020	test@Test.com	test	En revisión	diego@diego.com	
1021	aaaaaaaa@aaaaaa.aaaaaa	aaaaaaaaaaaaaaaaaa	Denegada	por bobo	
1022	diego@diego.com	fsdfds	En revisión		
2022	diego@diego.com	diego	Creada		
2023	fdsa@dsa.c	dfsa	Creada		
2024	diego@diego.com	gfsd	Creada	test	

8. Cambio de Estado de una Solicitud

2027	dssadsa@dsad.com	dsada	Creada
2028	dsada@dsa.com	dsda	Creada

Id: 2028

Nombre: dsda

Nuevo Estado: ▼

Descripcion:

Modificar

Universidad ORT Uruguay
Facultad de Ingeniería

Diseño de Aplicaciones 2

Obligatorio 2

Evidencia del diseño y especificación de la API

Diego Suero - 215193

<https://github.com/ORT-DA2/Obligatorio-1-215193>

Criterios REST

Para asegurar que la Api cumple con los criterios Rest, se utilizaron los HTTP verbs GET,PUT,POST,DELETE. Cada uno de estos verbos tiene la funcionalidad definida por CRUD, GET para obtener datos, Put para actualizar datos, Post para crear un nuevo recurso, Delete para borrar un recurso.

Las URI se hicieron lo más cortas posibles manteniendo la sencilla comprensión de parte del cliente, no se utilizaron verbos ni guiones y tienen una estructura jerárquica.

Mecanismo de autenticación

Para la autenticación se utilizó un Filter de Autenticación, el cual solicita un token que debe estar en el Header. El mismo es corroborado en la base de datos de sesiones, si no se encuentra se informa que no esta logueado o que es incorrecto y no se toma en cuenta la request.

Códigos de error

Los códigos de error que se utilizaron son:

- **BadRequest** : Indica que uno de los datos ingresados son incorrectos, se despliega con el mensaje informando el dato incorrecto y la razón.
- **OK** : Indica que la consulta fue satisfactoria, en el caso de un Post devuelve el ID del objeto creado, en el caso de Un get devuelve el objeto de Id ingresado, en el caso de un Put devuelve el Id del objeto actualizado.
- **NotFound** : Indica que el objeto que se está buscando mediante el ID no fue encontrado, también para cumplir los criterios REST, en el caso del Delete se devuelve un NotFound indicando que el objeto no existe más.
- **403**: Indica que el cliente no está logueado al sistema.

Resources de la API

La API esta dividida en 7 controllers diferentes, cada uno tiene una funcionalidad distinta y diferentes endpoints que se pueden utilizar

Administrador

El controller Administrador contiene 5 endpoints diferentes.

Administrador		▼
GET	/Administrador	
POST	/Administrador	
GET	/Administrador/{id}	
PUT	/Administrador/{id}	
DELETE	/Administrador/{id}	

1. GET

GET /Administrador

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ { "id": 0, "nombre": "string", "email": "string", "contrasena": "string" } }</pre></div>	No links

- Este endpoint GET devuelve una lista de todos los administradores que tiene el sistema.

2. POST

POST /Administrador

Parameters Try it out

Name	Description
auth	
integer	
(header)	

Request body application/json

Example Value | Schema

```
{  "nombre": "string",  "contrasena": "string",  "email": "string"}
```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div>	No links

- Este endpoint permite agregar un nuevo administrador al sistema, precisa un token de autentificación en el Header llamado “auth” para funcionar.

3. GET /id

GET /Administrador/{id}

Parameters Try it out

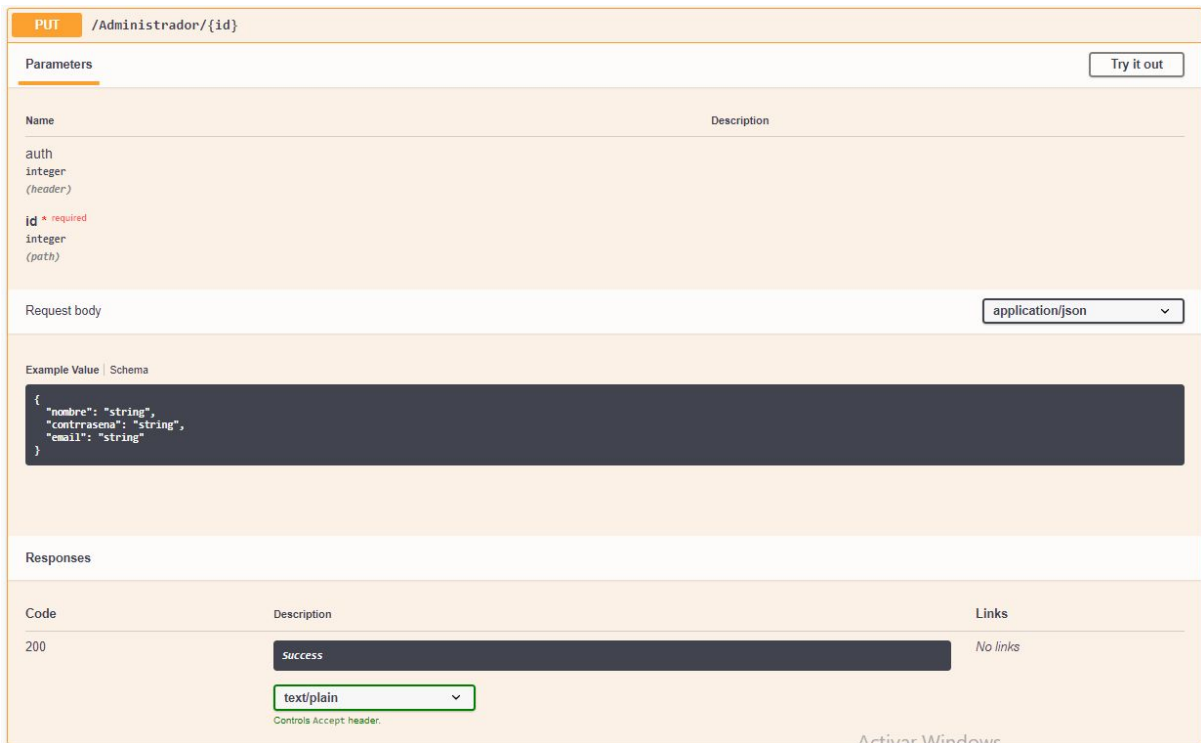
Name	Description
auth	
integer	
(header)	
id * required	
integer	
(path)	

Responses

Code	Description	Links
200	<div>Success</div>	No links

- Este endpoint permite obtener el Administrador con el Id que se pasa por el path del endpoint, precisa un token de autentificación en el Header llamado “auth” para funcionar.

4.PUT



The image shows the Swagger UI for the PUT endpoint `/Administrador/{id}`. The interface is divided into several sections:

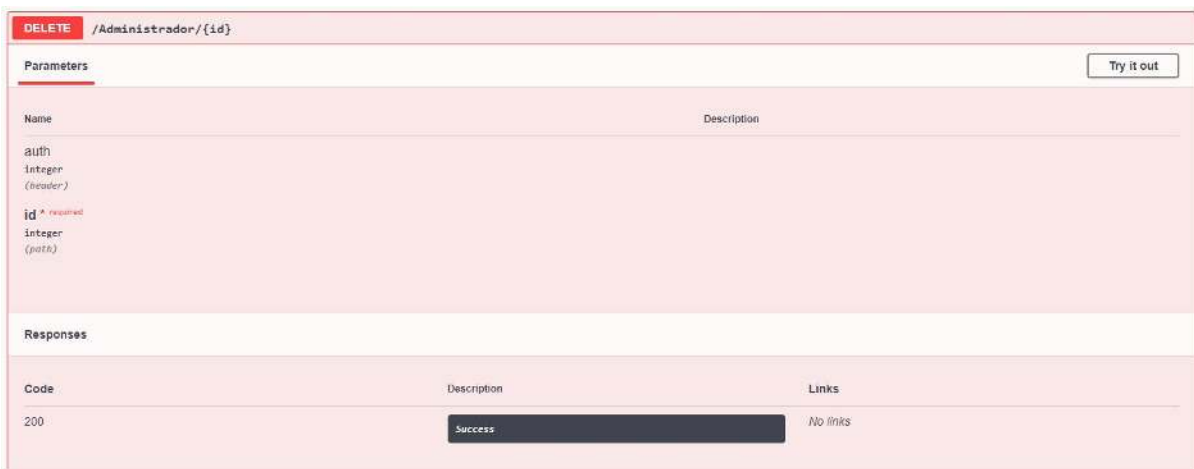
- Parameters:** A table with columns 'Name' and 'Description'. It lists two parameters:
 - `auth`: integer (header)
 - `id`: integer (path), marked as required with a red asterisk.
- Request body:** A dropdown menu set to `application/json`.
- Example Value / Schema:** A dark box containing a JSON schema:

```
{  "nombre": "string",  "contrasena": "string",  "email": "string"}
```
- Responses:** A table with columns 'Code', 'Description', and 'Links'. It shows a single response:
 - `200`: Success (with a 'text/plain' dropdown), No links.

An 'Activar Windows' watermark is visible at the bottom right of the interface.

- Este endpoint permite modificar el Administrador con Id que se le pasa por el path, precisa un token de autentificacion en el Header llamado "auth" para funcionar.
- En el body se debe pasar un modelo de Administrador.

5. Delete



The image shows the Swagger UI for the DELETE endpoint `/Administrador/{id}`. The interface is divided into several sections:

- Parameters:** A table with columns 'Name' and 'Description'. It lists two parameters:
 - `auth`: integer (header)
 - `id`: integer (path), marked as required with a red asterisk.
- Responses:** A table with columns 'Code', 'Description', and 'Links'. It shows a single response:
 - `200`: Success, No links.

- Este endpoint permite eliminar el Administrador que se le pasa por el path, precisa un token de autentificacion en el Header llamado "auth" para funcionar.

Login

Login ▼

POST /Login

GET /Login

1. Post

POST /Login

Parameters Try it out

No parameters

Request body application/json ▼

Example Value | Schema

```
{  "email": "string",  "password": "string"}
```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain ▼</div> <div>Controls Accept header</div>	No links

- Este endpoint permite hacer el Login de un Administrador al sistema, en caso de haber ingresado los datos bien, retorna el token de autentificación que debe ser ingresado en los endpoint que requieren autentificación.

2. GET

GET /Login

Parameters Try it out

Name	Description
token integer (query)	

Responses

Code	Description	Links
200	<div>Success</div>	No links

- Este endpoint permite obtener el Id de un Administrador con el token de autentificación.

Solicitud

Solicitud		▼
GET	/Solicitud	
POST	/Solicitud	
GET	/Solicitud/{id}	
PUT	/Solicitud/{id}	

1- Get

GET

/Solicitud

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <pre>{ "id": 0, "nombre": "string", "mail": "string", "telefono": "string", "detalle": "string", "descripcion": "string", "area": { "id": 0, "nombre": "string", "temas": [{ "id": 0, "nombre": "string", "campos": [{ "id": 0, "nombre": "string", "tipo": "string" }] }], "activo": true } }</pre>	

No links

- Este endpoint permite obtener todas las Solicitudes Del sistema.

2- POST

POST

/Solicitud

Parameters

Try it out

No parameters

Request body

application/json

Example Value

Schema

```
{  "nombre": "string",  "mail": "string",  "telefono": "string",  "detalle": "string",  "area": "string",  "tema": "string",  "tipo": "string",  "estado": "string",  "descripcion": "string",  "camposAdicionalesTexto": [    {      "valor": "string",      "valores": [        {          "id": 0,          "texto": "string"        }      ],      "id": 0,      "nombre": "string",      "tipo": "string"    }  ],  "camposAdicionalesFecha": [    {      "valor": "2020-05-13T03:22:20.330Z",
```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div>	No links

- Este endpoint permite agregar una nueva solicitud al sistema, se debe agregar en el body un modelo de Solicitud. el mismo esta descrito en la documentacion del sistema

3- GET by Id

GET

/Solicitud/{id}

Parameters

Try it out

Name	Description
id * required integer (path)	

Responses

Code	Description	Links
200	<div>Success</div>	No links

- Este endpoint permite obtener una solicitud con el id que se pasa por el path.

4- PUT

PUT

/Solicitud/{id}

Try it out

Parameters

Name	Description
id * required integer (path)	

Request body

application/json

Example Value | Schema

```
{
  "nombre": "string",
  "mail": "string",
  "telefono": "string",
  "detalle": "string",
  "area": "string",
  "tema": "string",
  "tipo": "string",
  "estado": "string",
  "descripcion": "string",
  "composAdicionalesTexto": [
    {
      "valor": "string",
      "valores": [
        {
          "id": 0,
          "texto": "string"
        }
      ],
      "id": 0,
      "nombre": "string",
      "tipo": "string"
    }
  ],
  "composAdicionalesFecha": [
    {
      "valor": "2020-05-13T03:23:06.750Z",
      "cotaInferior": "2020-05-13T03:23:06.750Z",

```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div>	No links

- Este endpoint permite modificar la solicitud que se pasa por el path de la request, en el body se debe ingresar un modelo de la solicitud. De la misma, solo se puede modificar el valor “estado” y “descripcion”, todos los otros cambios no se tienen en cuenta.

Tipo

Tipo

POST

/Tipo

PUT

/Tipo

GET

/Tipo

1-POST

POST

/Tipo

Try it out

Parameters

Name	Description
auth	

integer
(header)

Request body

application/json

Example Value | Schema

```
{  "area": "string",  "loma": "string",  "tipo": "string",  "activo": true,  "composAdicionales": [    {      "id": 0,      "nombre": "string",      "tipo": "string"    }  ]}
```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header</div>	No links

- Este endpoint permite agregar un tipo al sistema, se debe ingresar en el body un modelo de Tipo. requiere que se ingrese en el Header un token de autentificacion.

2-PUT

PUT

/Tipo

Try it out

Parameters

Name	Description
auth	
integer	
(header)	

Request body

application/json

Example Value | Schema

```
{  "area": "string",  "tema": "string",  "tipo": "string",  "activo": true,  "camposAdicionales": [    {      "id": 0,      "nombre": "string",      "tipo": "string"    }  ]}
```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls accept header.</div>	No links

-Este endpoint permite modificar un tipo del sistema, se debe ingresar en el body un modelo de Tipo. requiere que se ingrese en el Header un token de autentificacion.

- Solamente se puede modificar si el tipo esta Activo o agregar campos adicionales al tipo.

3- GET

GET

/Tipo

Try it out

Parameters

No parameters

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls accept header.</div> <div>Example Value Schema</div> <div>string</div>	No links

- Este endpoint permite tener todos los tipos del sistema.

PUT

/Tipo

Parameters

Try it out

Name	Description
auth	
integer	
(header)	

Request body

application/json

Example Value | Schema

```
{  "area": "string",  "tema": "string",  "tipo": "string",  "activo": true,  "camposAdicionalesEnteros": [    {      "valor": 0,      "cotaInferior": 0,      "cotaSuperior": 0,      "id": 0,      "nombre": "string",      "tipo": "string"    }  ],  "camposAdicionalesTextos": [    {      "valor": "string",      "valores": [        {          "id": 0,          "texto": "string"        }      ]    },    {      "id": 0,      "nombre": "string",      "tipo": "string"    }  ]}
```

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div>	No links

AREA

Area

GET /Area

1-GET

GET /Area

Parameters

No parameters

Try it out

Responses

Code	Description	Links
200	<div>Success</div> <div>text/plain</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <pre>{ "id": 0, "nombre": "string", "temas": [{ "id": 0, "nombre": "string", "tipos": [{ "id": 0, "nombre": "string", "campos": [{ "id": 0, "nombre": "string", "tipo": "string" }], "activo": true }] }] }</pre>	

No links

-Este endpoint permite tener todas las Areas del sistema.

TEMA

Tema

GET /Tema

GET /Tema/{id}

1-GET

Tema

GET /Tema

Parameters

No parameters

Try it out

Responses

Code	Description	Links
200	<p>Success</p> <p>text/plain</p> <p>Controls Accept header.</p> <p>Example Value Schema</p> <pre>[{ "id": 0, "nombre": "string", "tipos": [{ "id": 0, "nombre": "string", "campos": [{ "id": 0, "nombre": "string", "tipo": "string" }], "activo": true }] }]</pre>	No links

- Este endpoint permite tener todos los Temas del sistema.

2-GET/id

GET /Tema/{id}

Parameters

Try it out

Name	Description
id * required integer (path)	

Responses

Code	Description	Links
200	<p>Success</p>	No links

- Este endpoint permite obtener el Tema del Id que se pasa por el Path

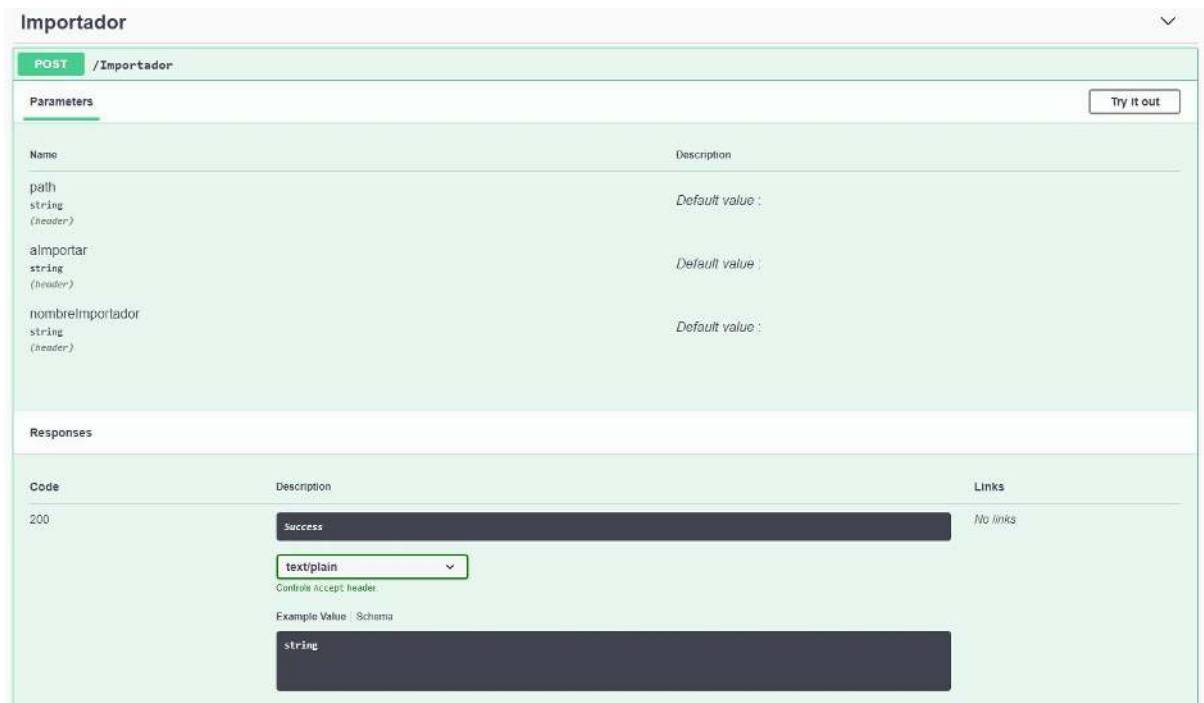
Importador

Importador

POST /Importador

GET /Importador

1- POST



The image shows the Swagger UI for the POST endpoint `/Importador`. The interface is divided into two main sections: **Parameters** and **Responses**.

Parameters: This section lists three headers:

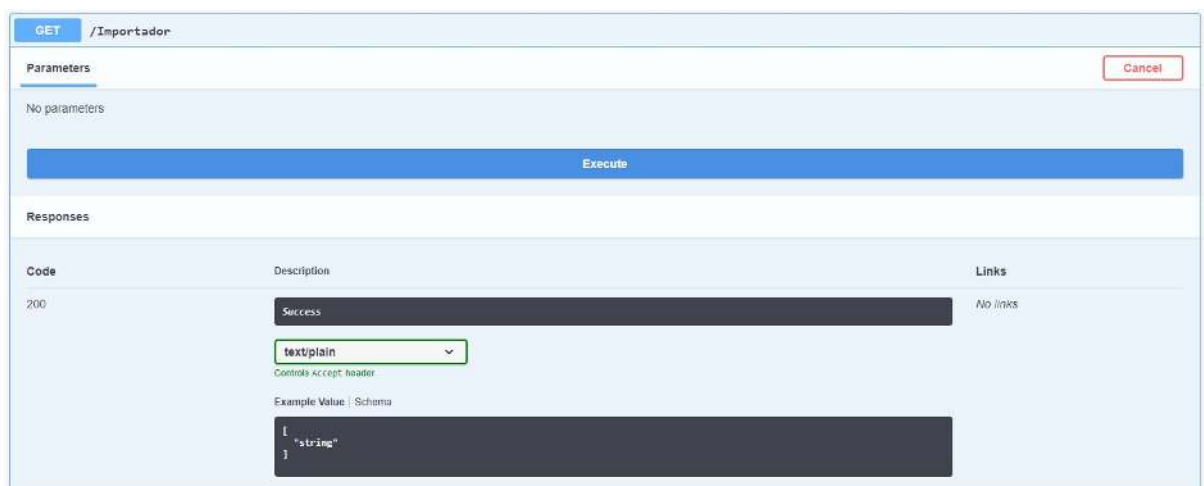
Name	Description
<code>path</code> string (header)	Default value :
<code>alimportar</code> string (header)	Default value :
<code>nombreimportador</code> string (header)	Default value :

Responses: This section shows the response for status code 200:

Code	Description	Links
200	<p>Success</p> <p>text/plain</p> <p>Controls accept header</p> <p>Example Value Schema</p> <p>string</p>	No links

- Este endpoint permite importar archivos desde el path que se ingresa por el header.
- `alimportar` debe tener los valores ["Áreas", "Temas", "Tipos"], el mismo le dice al sistema que se quiere importar desde el path ingresado.
- `nombreImportador`: se debe ingresar el nombre del Importador que se quiere utilizar.

2-GET



The image shows the Swagger UI for the GET endpoint `/Importador`. The interface is divided into two main sections: **Parameters** and **Responses**.

Parameters: This section indicates "No parameters" and includes an **Execute** button.

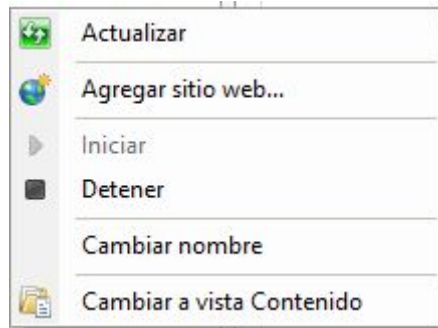
Responses: This section shows the response for status code 200:

Code	Description	Links
200	<p>Success</p> <p>text/plain</p> <p>Controls accept header</p> <p>Example Value Schema</p> <pre>{ "string" }</pre>	No links

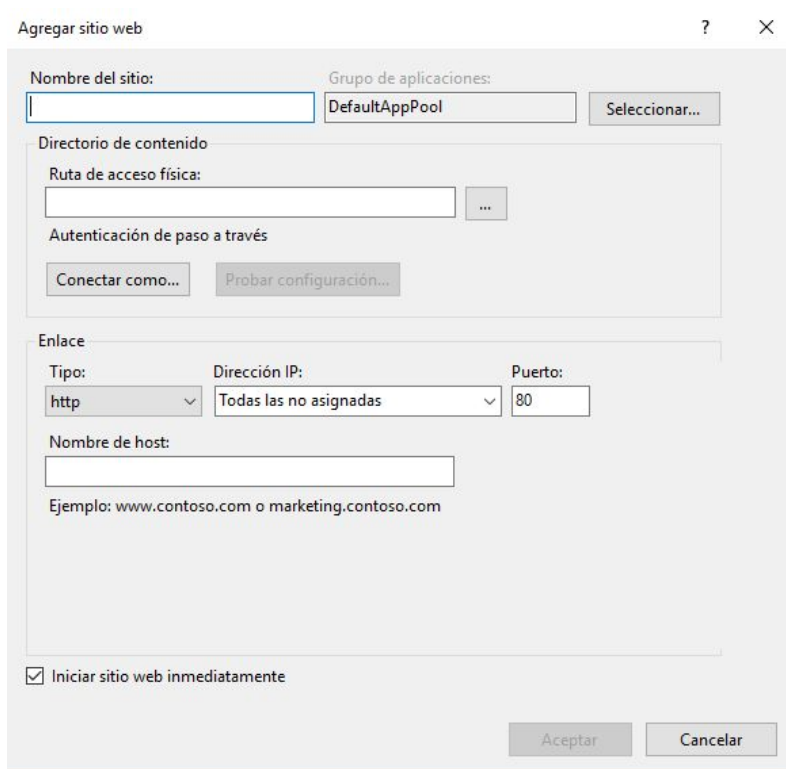
- Este endpoint permite obtener todos los nombres de los importadores que se ingresaron al sistema.

Deployment de la Aplicación Web:

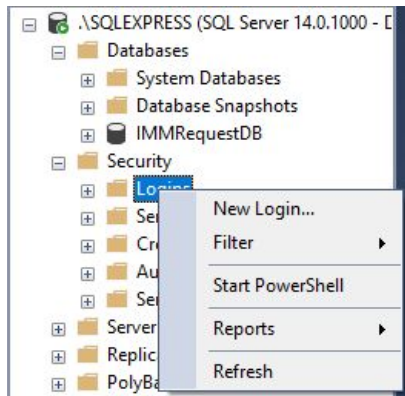
1. Copiar la carpeta de la aplicacion Angular y pegarla en "C:\inetpub\wwwroot". Esta carpeta es la carpeta por defecto del IIS, los usuarios de IIS tiene permisos necesarios sobre dicho directorio. Si ya existe una carpeta con el mismo nombre en dicho directorio, pueden renombrar su carpeta antes de pegarla.
2. Abrir IIS (CTRL + R y inetmgr)
3. Una vez en IIS, agregar un nuevo sitio web.(Click derecho en Sitios)



4. En el Nombre del Sitio, se pone el nombre que se desee.
En ruta de acceso física, seleccionar la carpeta donde esta guardada la aplicacion web "C:\inetpub\wwwroot\NOMBREDELAAPP".
Se puede elegir un puerto cualquiera, si se deja el 80 osea el por defecto, se debe dar de baja a la aplicacion por defecto de IIS.
Al momento de crear el sitio web se crea un usuario "IIS APPPOOL\NOMBREDELAAPP", esto nos va a servir en los pasos mas adelante.

A screenshot of the 'Agregar sitio web' (Add website) dialog box in IIS Manager. The dialog has several sections: 'Nombre del sitio:' (Site name) with a text box; 'Grupo de aplicaciones:' (Application pool) with a dropdown set to 'DefaultAppPool' and a 'Seleccionar...' button; 'Directorio de contenido:' (Content directory) with a 'Ruta de acceso física:' (Physical path) text box and a browse button (...); 'Autenticación de paso a través' (Authentication) with 'Conectar como...' and 'Probar configuración...' buttons; 'Enlace' (Binding) section with 'Tipo:' (Type) set to 'http', 'Dirección IP:' (IP address) set to 'Todas las no asignadas' (All unassigned), 'Puerto:' (Port) set to '80', and 'Nombre de host:' (Host name) text box; and a checkbox 'Iniciar sitio web inmediatamente' (Start website immediately) which is checked. At the bottom are 'Aceptar' (OK) and 'Cancelar' (Cancel) buttons.

5. En SQLMANAGMENTSTUDIO hay que otorgar acceso al usuario ya creado. Se Supone que la base de datos ya esta creada.



Seleccionar New Login...

En Login name ingresar el que se obtuvo mas arriba

"IIS APPPOOL\NOMBREDELAAPP"

Clickear en serverRoles, y elegir Sysadmin

Login - New

Script ? Help

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Connection

Server: DESKTOP-9PA8KG4\SQLEXPRESS

Connection: DESKTOP-9PA8KG4\diego

[View connection properties](#)

Progress

Ready

Login name:

Search...

☒ Windows authentication
☐ SQL Server authentication

Password:

Confirm password:

☐ Specify old password

Old password:

☒ Enforce password policy
☒ Enforce password expiration
☒ User must change password at next login

☐ Mapped to certificate
☐ Mapped to asymmetric key
☐ Map to Credential

Mapped Credentials

Credential	Provider
------------	----------

Add

Remove

Default database: master

Default language: <default>

OK Cancel

Login - New

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Script ? Help

Server role is used to grant server-wide security privileges to a user.

Server roles:

- ☐ bulkadmin
- ☐ dbcreator
- ☐ diskadmin
- ☐ processadmin
- ☒ public
- ☐ securityadmin
- ☐ serveradmin
- ☐ setupadmin
- ☒ sysadmin

6. Ir a IPconfig (Ctrl + r / cmd / ipconfig) y copiar la ip de la maquina 1.
7. En la raiz de la WebApi, buscar el archivo Main, abrirlo con Notepad++ y buscar apiUrl , cambiar la ip por la IP de la maquina donde esta corriendo la Web Api se debe bajar el proxy de la maquina donde esta corriendo la webApi para asi se puede hacer la conexion.
8. Al ingresar a la IP de la maquina 1 donde esta corriendo la AplicacionWeb, con el puerto seleccionado, debe aparecer la pagina principal de la app

