

Universidad ORT Uruguay
Facultad de Ingeniería

Diseño de Aplicaciones 2

Obligatorio 2

Documentación

Mauro Teixeira(211753) Rodrigo Hirigoyen (197396)

Docentes: Gabriel Piffaretti, Nicolas Fierro, Nicolas Hernandez

<https://github.com/mauroteix/Teixeira-Hirigoyen>

Entregado como requisito de la materia Diseño de Aplicaciones 2

17 de junio de 2021

Índice

1. Descripción general del trabajo	2
Errores conocidos	3
Funcionalidades no implementadas	3
2. Vista lógica	4
Lógica de negocio	4
Capa de acceso a datos	5
Manejo de excepciones	5
Diagramas de interacción	6
Descripción de jerarquías de herencia utilizadas	7
3. Vista desarrollo	8
Diagrama de paquetes	8
Responsabilidad de cada paquete	8
Diagrama de componentes	13
Modelo de tablas de la estructura de la base de datos	14
4. Vista de proceso	14
Diagrama de Actividad	15
5. Patrones de diseño y métricas	16
Abstracciones estables	18
Interpretación de resultados	19
Interpretaciones de valores de distancia	20
Dependencias estables	21
6. Cambios con respecto a la primer entrega	23
7. Características del frontend	23
Anexo	25
Cobertura de pruebas	25
Descripción de la api	26
Datos de prueba	31

1. Descripción general del trabajo

Nuestro proyecto es una aplicación para el Ministerio de Salud Pública (MSP) que permita a las personas aprender a sobrellevar el estrés de una manera más sana para que cada persona y sus seres queridos puedan desarrollar una mejor capacidad de adaptación y resiliencia. Dicha aplicación es denominada BetterCalm.

Se cumplió con los siguientes requerimientos especificados para este sistema. Los mismos se listan a continuación.

- **Reproductor - listar categorías**

Ya definidas dormir, meditar, música y cuerpo. De cada categoría se puede navegar entre sus playlist y su contenido reproducible(track y video) de forma pública.

- **Agenda con psicólogos**

Se permite seleccionar la condición médica dentro de las establecidas, para luego realizar la reserva con un psicólogo, indicando la duración(1,1.5 o 2 horas) de la consulta. Luego de realizada el usuario podrá visualizar el costo de la misma.

- **Manejo de sesiones**

Se realiza login a la aplicación mediante email y password solo para usuarios administradores. Se recibe un token para autorizaciones.

- **ABM de usuario administrador**

Debe estar logueado.

Se identifican por su email

- **ABM de contenido reproducible (track y video)**

Debe estar logueado.

Se identifican por nombre. Deben estar asociadas al menos una categoría, y pueden estar o no asociadas a una o varias playlists.

- **ABM de playlist**

Debe estar logueado. Se identifica por nombre. Deben estar asociadas al menos una categoría, y pueden estar o no asociadas a contenido reproducible.

- **ABM de psicologo**

Debe estar logueado

Debe estar asociado al menos una condición médica y como máximo tres. A su vez se agregó que cada psicólogo tiene una tarifa(500,750,1000,2000). El formato de consulta puede ser virtual o presencial.

- **Aplicar descuentos**

Debe estar logueado

De una lista de usuarios los cuales tienen 5 o más reuniones agendadas, se puede o no aplicar un descuento de 15%,25% o 50%. Este beneficio si existiese será aplicado en la próxima agenda que el usuario realice.

La aplicación está desarrollada en .NET Core, utilizando tecnologías como Entity Framework Core y SQL Server para la persistencia de los datos y arquitecturas web REST que brindan los servicios web que la plataforma utiliza. Y para el frontend se utilizó Angular, para las vistas de el frontend se utilizó el framework de CSS Bootstrap 4. Si bien la mayoría de estas tecnologías fueron un requisito obligatorio, consideramos que son tecnologías modernas, que permiten extensibilidad y asegurar la entrega de software de calidad.

Errores conocidos

No hay errores conocidos que incapaciten el correcto funcionamiento de la api. En cada sección de justificaciones de diseño se comentarán posibles mejoras respecto al mismo en caso de identificarlos.

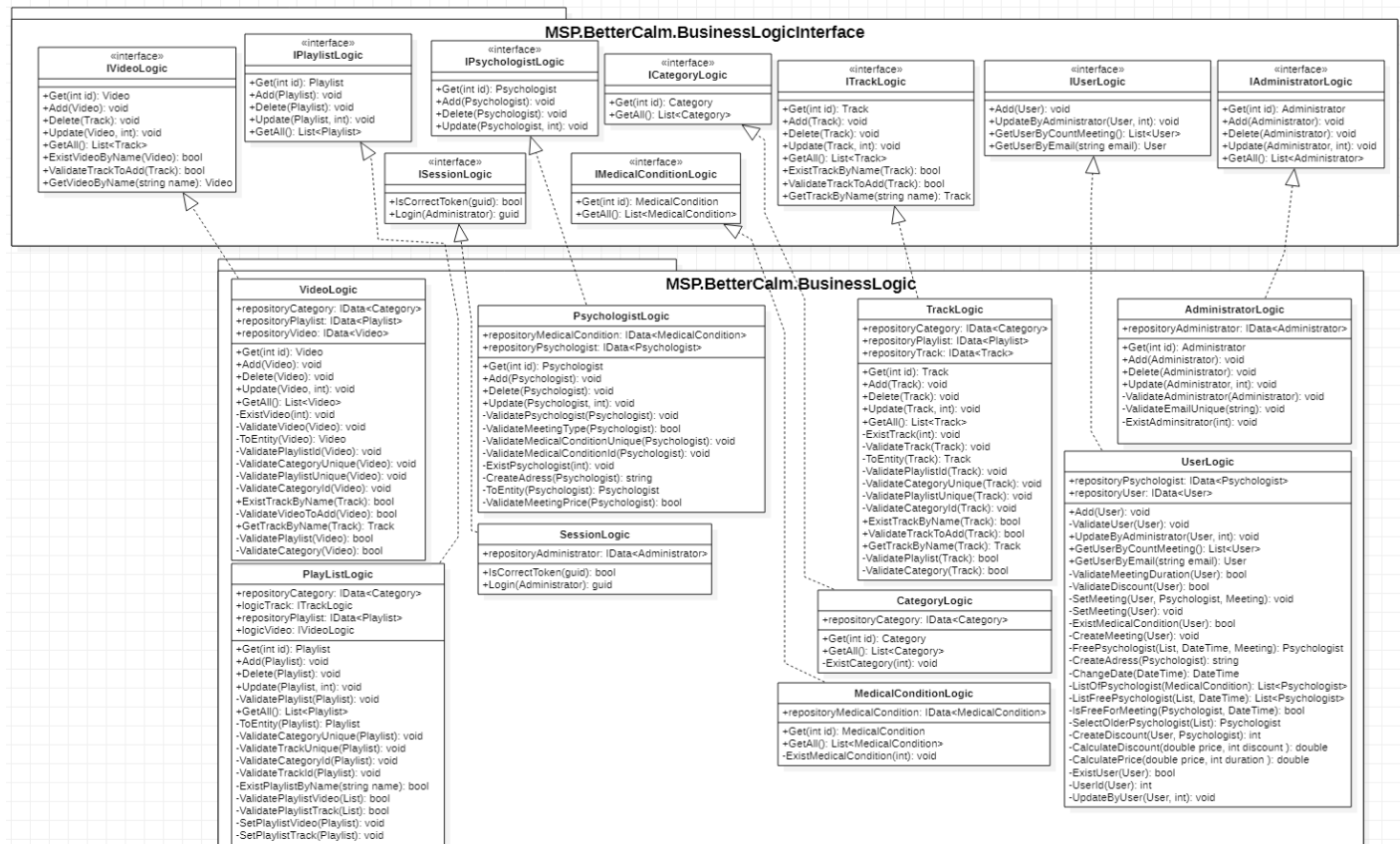
Funcionalidades no implementadas

No fue implementada la funcionalidad del importer.

2. Vista lógica

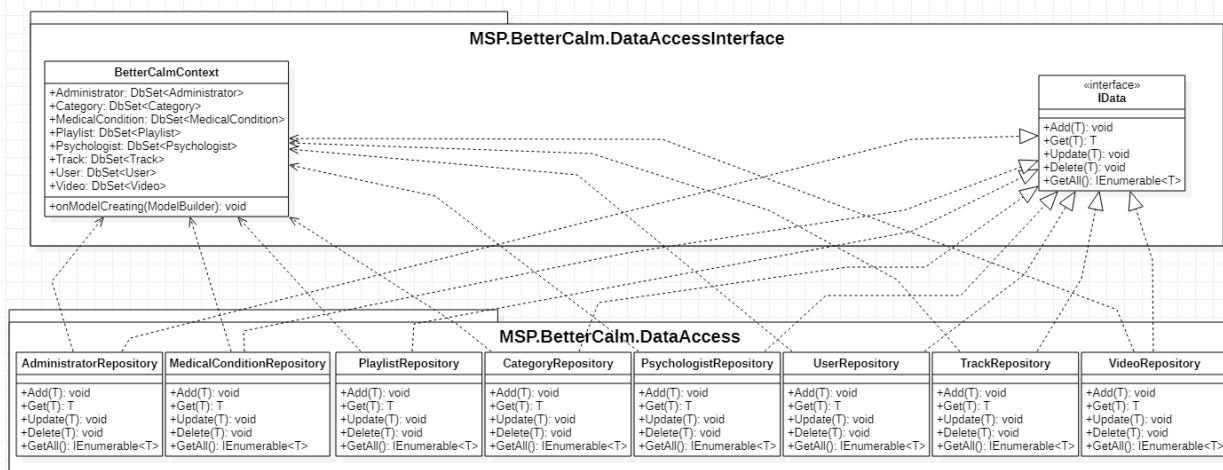
Lógica de negocio

La lógica de negocio se encarga de definir el comportamiento de cada una de nuestras entidades, definimos interfaces las cuales son requeridas por la API para realizar la validación de los datos recibidos y actuar como con la capa de acceso a datos. Se optó por encapsular la lógica del sistema en la carpeta llamado BusinessLogic, luego adentro tenemos los paquetes, por un lado la interfaz y por otro la lógica con sus clases concretas. Consideramos que lo mejor era tener varias interfaces pequeñas para una correcta aplicación del principio de segregación de interfaces para evitar que las implementaciones se vean obligadas a implementar métodos que no sean utilizados y por otra parte para que las clases tengan alta cohesión.



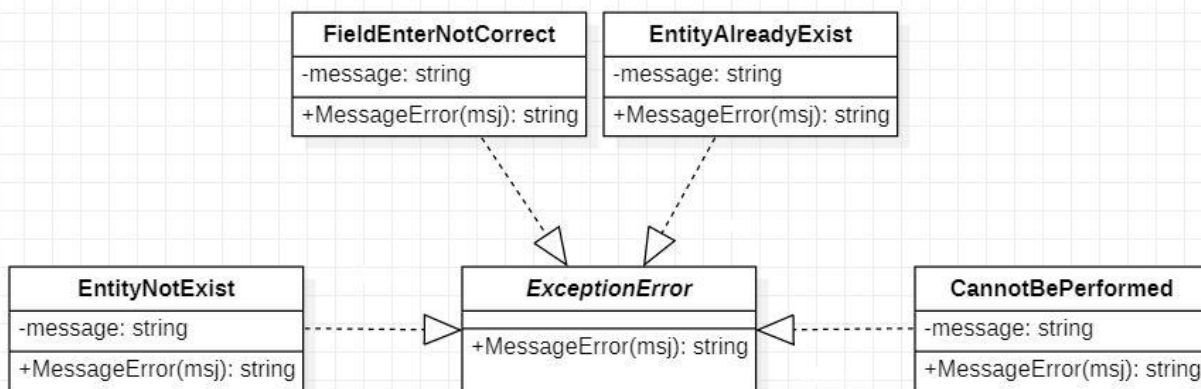
Capa de acceso a datos

En el I acceso a datos se utilizó SQLServer con Entity Framework Core approach Code-First y mantuvimos la base de datos actualizada mediante el uso de migrations. Para no depender necesariamente de la base de datos Sql Server que utilizamos, porque el día de mañana podría cambiar, creamos una interfaz para depender de la misma y no importarnos que es lo que sucede por atras . Creamos una clase abstracta, para dos cosas: La primera tener un contrato la cual toda clase que implemente un repositorio deba cumplir con las operaciones que se le especificaron y la segunda compartir código entre todas, ya que como hablábamos anteriormente, mucho código se repite. Cada uno de los repositorios implementa a IData.



Manejo de excepciones

A continuación mostramos como esta armada nuestra arquitectura para el manejo de errores:



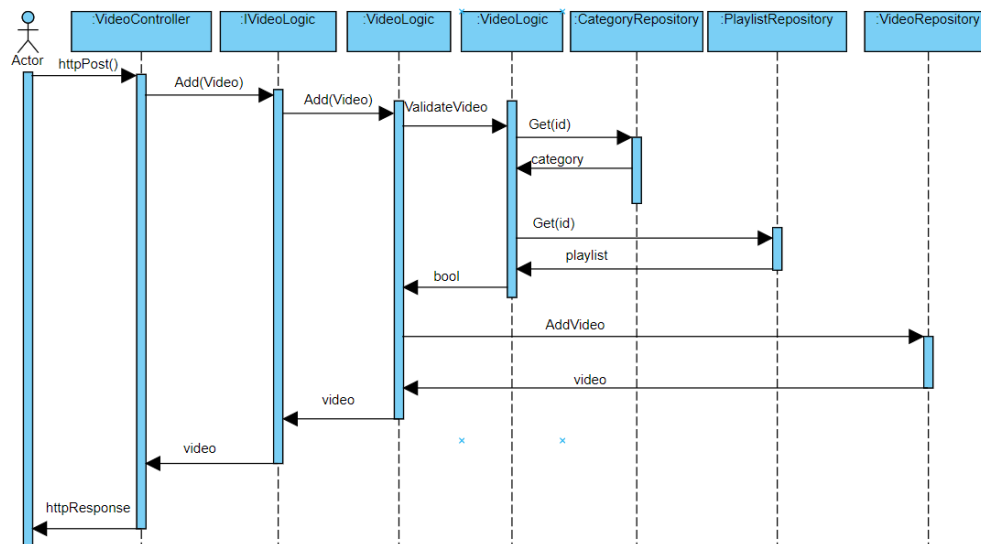
Para reforzar el concepto de división de responsabilidades, notamos que en diversos lugares de nuestra aplicación se lanzaban las excepciones, que al fin y al cabo no eran más que las mismas excepciones por situaciones en nuestros casos de uso. Por este motivo decidimos empaquetar las mismas dentro de un proyecto de manejo de excepciones, representando cada

excepción común en forma de clase la cual recibe por parámetro el mensaje de error. Como existen varios códigos de error relacionados al mundo de las api, tuvimos que tener mucho cuidado de cómo manejamos nuestros errores internamente para que al usuario que está consumiendo nuestra api le pueda llegar el error específico que sucedió y además un texto con una descripción lo más acertada posible para que pueda darse cuenta de que fue lo que realmente sucedió. Por este motivo decidimos crear una biblioteca de clases en nuestra solución, la cual cada clase que creemos será una error diferente, estos errores heredarán de una clase abstracta la cual la creamos para que todos los errores tengan un mensaje de error y además esta clase abstracta hereda de Exception ya que en nuestro sistema, cada vez que necesitamos tirar un error el throw exception necesita que se hereda de Exception lo cual es evidente. Las excepciones son lanzadas en la capa de acceso a datos o en la lógica de negocio dependiendo el caso, y capturados en la api para lanzar mensajes de error al usuario lo más acertados posibles, este manejo de errores por parte de los controladores es de notar ya que siguiendo con quien es el experto en información decidimos dejarle este control a los controladores.

Diagramas de interacción

➤ Agregar Video

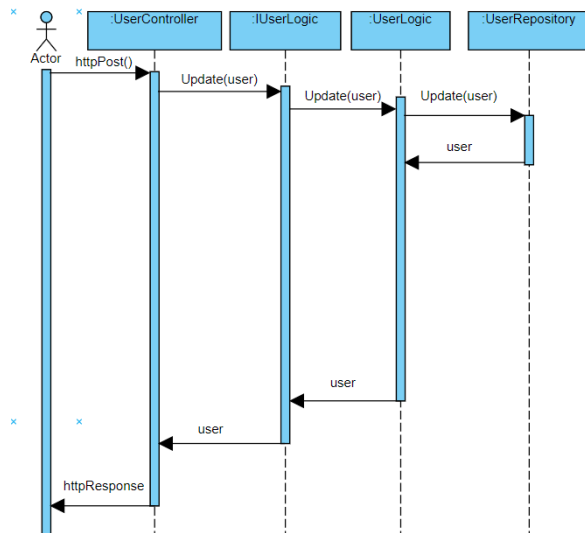
En este diagrama representamos como agregar una video,nueva funcionalidad pedida en esta segunda entrega. El controller de video llama a la interface IVideoLogic que luego llama a la videoLogic para hacer el add de video. Tenemos dos videologic porque el método validateVideo tiene varias llamadas a otras clases, tanto categoryrepository y playlistrespository para validar y verificar que existan las clases asociadas a esta video. Para finalizar crea y agrega la video desde videoRepository y devuelve la respuesta.



➤ Update User

Otra de las nuevas funcionalidades pedidas, este update mas que nada es un update de discount, da de alta o baja el discount elegido mediante el cual se lo pasas por el user. Una vez

cambiado el discount por el administrador y la meeting count el usuario al agendar una nueva meeting se le aplicará el descuento.



Descripción de jerarquías de herencia utilizadas

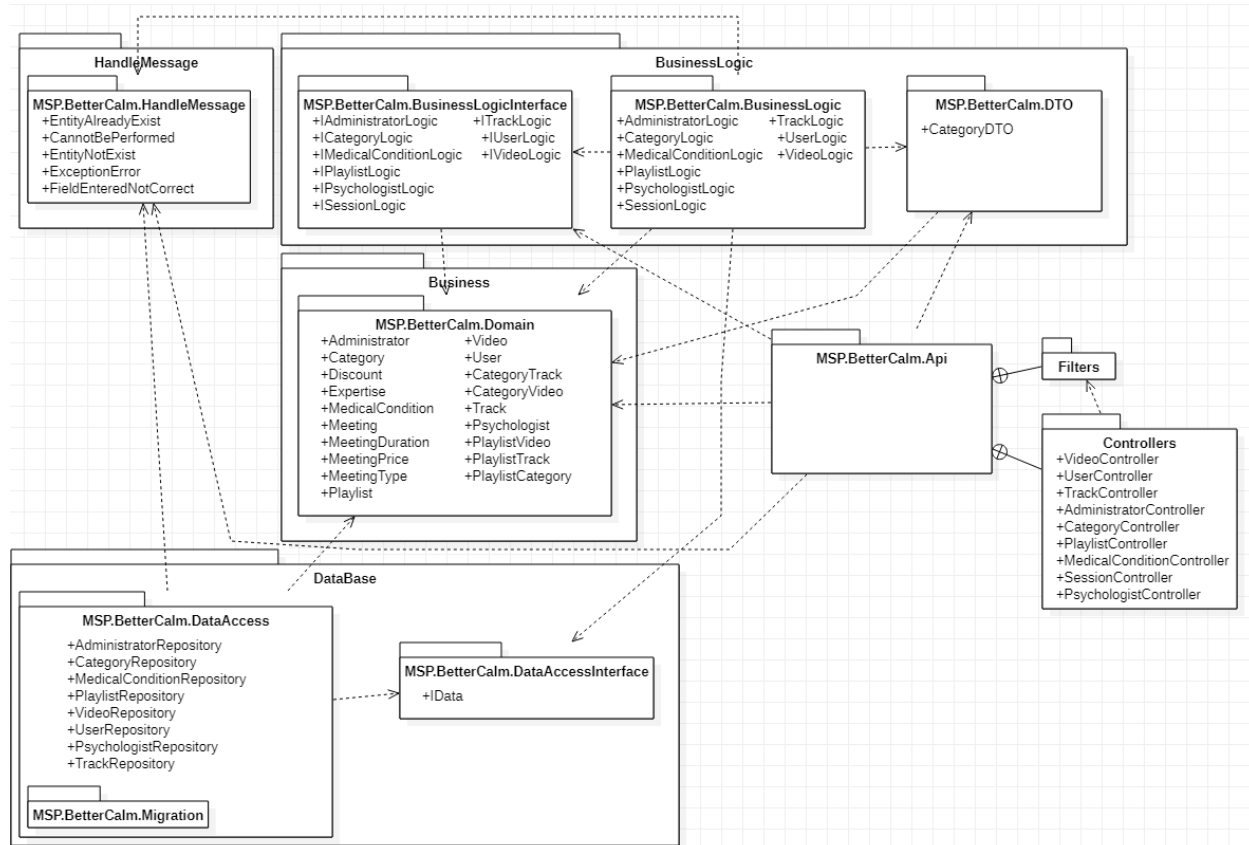
En nuestro obligatorio no se utilizaron herencias, sino que se optó por utilizar composición. ¿Por qué no se utilizaron herencias? Si bien el contenido reproducible tiene dos tipos hubiese sido una correcta representación utilizando herencia, ya que entre track y video comparten atributos. Por motivos de tiempo y de cambios significativos en la implementación previamente realizada se optó por no aplicarlo.

A su vez, la mantenibilidad de nuestro proyecto fue uno de los principales puntos a tener en cuenta a la hora de tomar las decisiones de diseño, no nos pareció adecuado plantear soluciones con modelos de herencia, ya que da muy poca flexibilidad a futuro.

En su lugar utilizamos composición, recibiendo objetos (interfaces) en los constructores de los métodos, delegando responsabilidades de que cada clase sepa hacer lo que debe hacer. La combinación de composición con patrones como inyección e inversión de dependencias nos permitieron tener clases que utilicen interfaces bien definidas, delegando las implementaciones a clases concretas que se asignan mediante inyección de dependencias, generando una solución mantenible.3. Vista de desarrollo

3. Vista desarrollo

Diagrama de paquetes

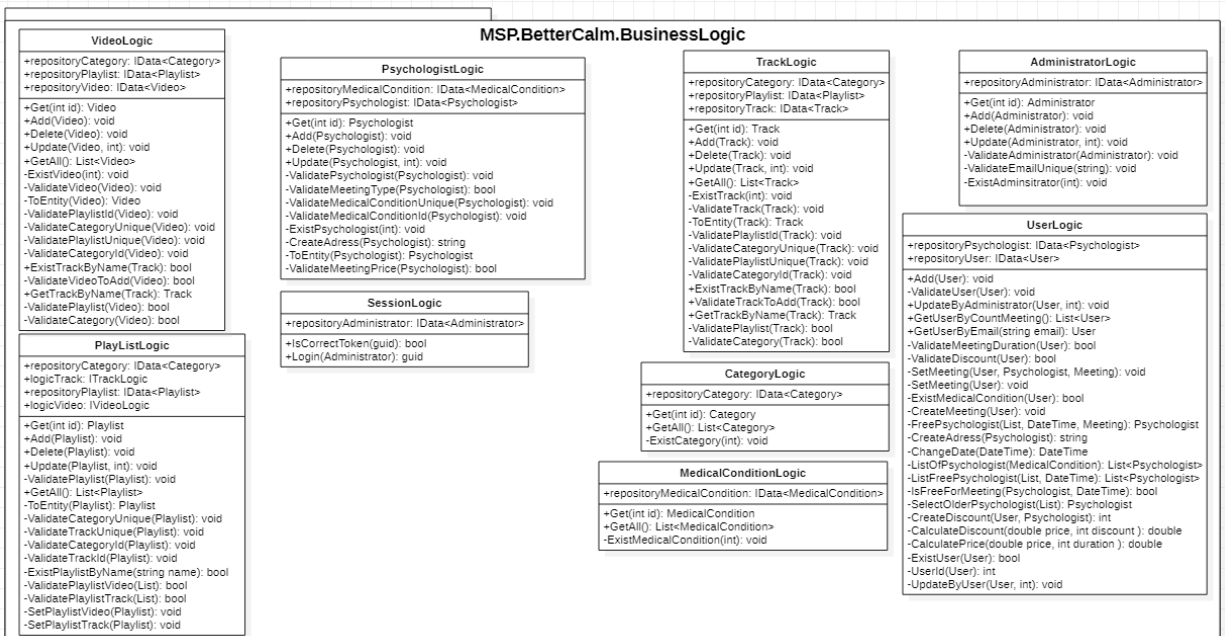


Responsabilidad de cada paquete

MSP.BetterCalm.Api

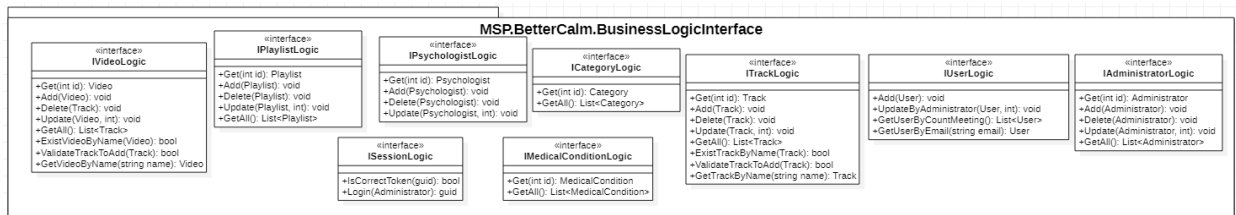
Capa de WebApi. Se encarga de exponer la información del sistema al mundo exterior mediante los endpoints (se pueden ver en detalle en el documento de la API). En la clase Startup de este paquete se realizan las inyecciones de dependencias que nos permiten tener la capa de la API desacoplada con respecto al resto del sistema, ya que las clases de la API dependen de interfaces y no de implementaciones concretas (Inversión de dependencias). Este paquete también se encarga de realizar la seguridad utilizando filtros de autorización para asegurarnos que los endpoints para administradores no devuelvan datos en caso de que la consulta no posea un token válido en el header.

MSP.BetterCalm.BusinessLogic



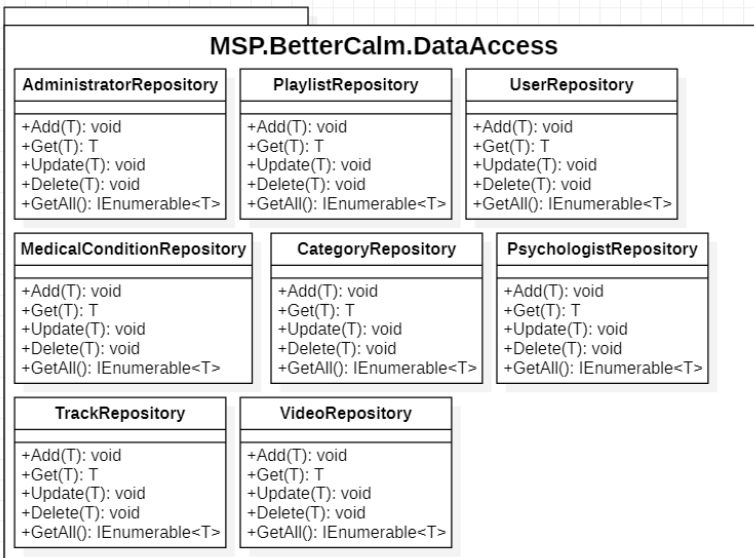
La capa de lógica de negocio se asegura de que los datos que se pasan desde la WebApi hacia la capa de acceso a datos sea válida. Cada entidad tiene su propia lógica, la cual conoce a la entidad que representa y a la lógica del negocio y se asegura que los datos sean válidos. También se encarga de realizar cálculos propios de la lógica de negocios como podría ser el cálculo del costo de una consulta con un psicólogo, o el descuento aplicado a los usuarios solicitado en el nuevo requerimiento.

MSP.BetterCalm.BusinessLogicInterface



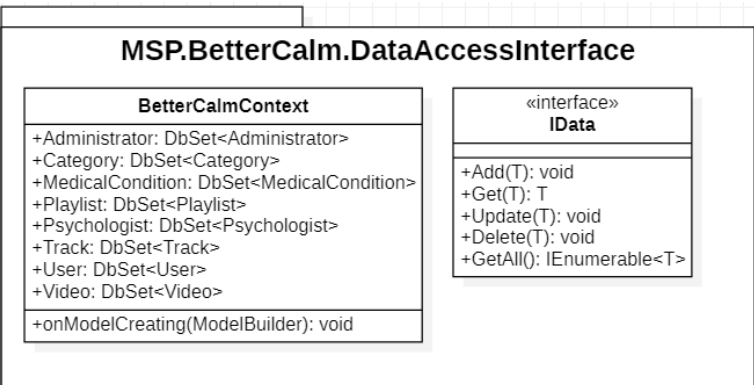
Este paquete se encarga de definir el comportamiento que debe tener la lógica de negocio. La razón de existencia de este paquete que posee únicamente interfaces es aumentar la mantenibilidad y disminuir el acoplamiento de nuestra solución, ya que gracias al mismo se puede realizar inversión e inyección de dependencias, haciendo que nuestras clases de la api utilicen estas interfaces en los controladores, inyectando las dependencias concretas (implementaciones) en la clase Startup.

MSP.BetterCalm.DataAccess



Este paquete se encarga de implementar la capa de acceso a datos, se implementan los métodos concretos para realizar operaciones para agregar, eliminar, obtener y editar datos. En este caso el obligatorio solicitó implementar el motor de base de datos SQLServer utilizando entity framework Code First, por lo que la sintaxis de nuestras implementaciones fueron acordes a dichas herramientas.

MSP.BetterCalm.DataAccessInterface



Este obligatorio tuvo como requisito implementar un determinado motor de base de datos y ORM, pero en un futuro estas herramientas pueden ser modificadas por otras.

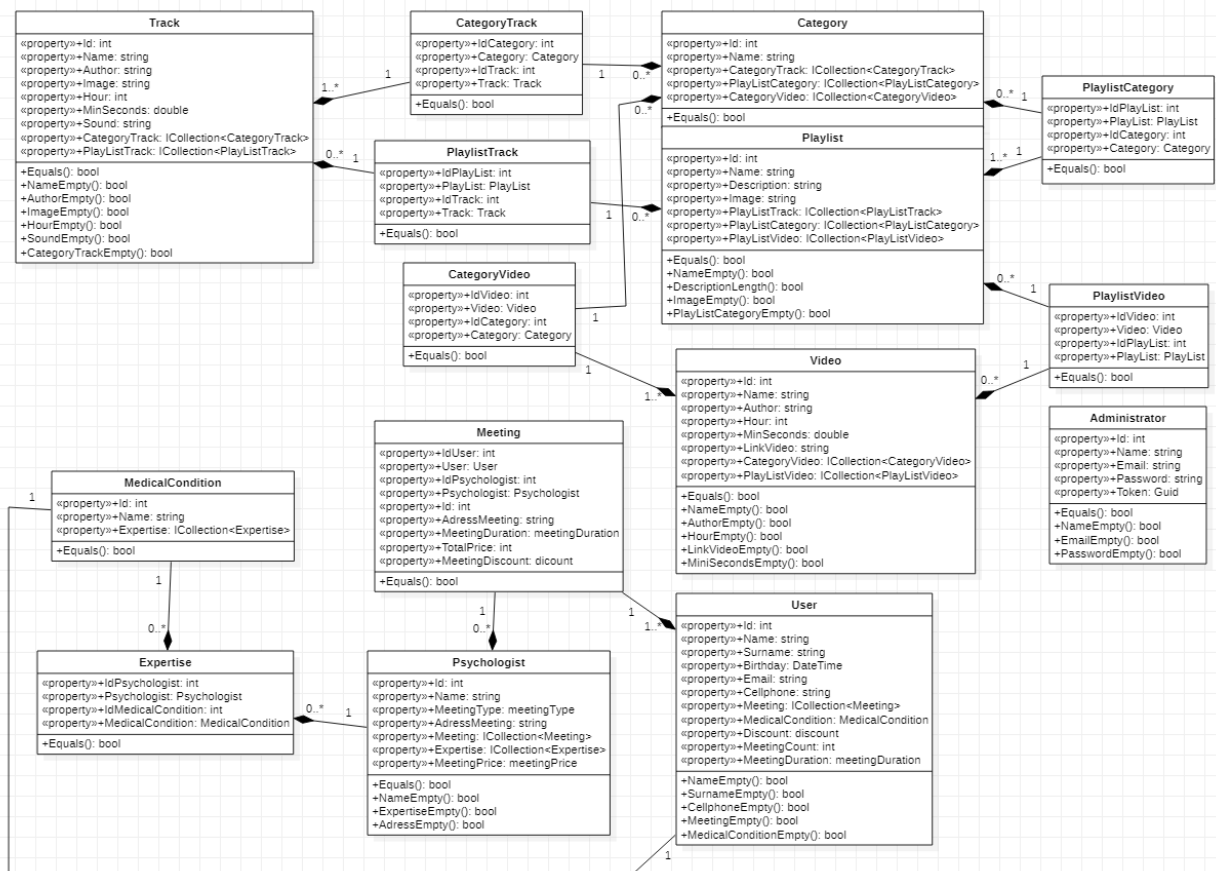
Este paquete se encarga de definir interfaces para la capa de acceso a datos, la cual es utilizada por el paquete MSP.BetterCalm.BusinessLogic mediante (nuevamente) inyección e inversión de dependencias. Esto permite poder en un futuro implementar un motor distinto de base de datos, debiendo implementar una nueva DataAccess para la misma, pero sin cambiar la lógica de negocio, ya que la misma utiliza la interfaz y no la implementación concreta.

Manejo del contexto de EF: Como mencionamos anteriormente la persistencia de los datos se realiza en una base de datos SQL Server, a la cual nos conectamos desde nuestro backend por medio del ORM Entity Framework. La conexión a la base se da mediante el contexto definido en la clase BetterCalmContext.cs, se mantiene una única que se crea en la clase Startup.cs mediante inyección de dependencias. Allí se indica el connectionString definido en appsettings.json y se especifica que el motor de base de datos a utilizar será SQL Server (options.UseSqlServer..).

Recordemos que nuestras dependencias son hacia interfaces, también en la capa de acceso a datos, por lo que sería “sencillo” en un futuro cambiar la base de datos, modificando las implementaciones de las interfaces y modificando el método options, indicando la nueva base de datos a utilizar.

MSP.BetterCalm.Domain

Este paquete se encarga de realizar las representaciones de los objetos de nuestra solución. La explicación de cada una de las entidades se realizó para la entrega anterior y por motivos de largo del documento decidimos no incluirlas nuevamente. Describiremos únicamente las nuevas entidades agregadas.



Video

Se solicitó para esta entrega agregar la funcionalidad de realizar un nuevo tipo de contenido: video. Estos siguen las mismas reglas que los audios, pertenecen a una o más categorías, pueden pertenecer a una o varias playlists, tienen un nombre, una duración en horas, el nombre del creador del contenido, y una url al archivo de video a reproducir. Para manejar los videos también se agregaron las correspondientes clases de lógica, acceso a datos y endpoint en la web api, estas clases no corresponden a esta sección y serán mostradas más adelante en el documento.

Psychologist

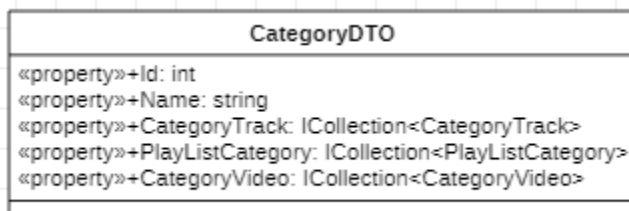
También se solicitó agregar para cada psicólogo un nuevo campo “tarifa” que puede ocupar los siguientes valores: [\$500/h, \$750/h, \$1000/h, \$2000/h]. Se decidió implementarlo con un enum para poder tener mayor escalabilidad y menor impacto en el cambio.

User

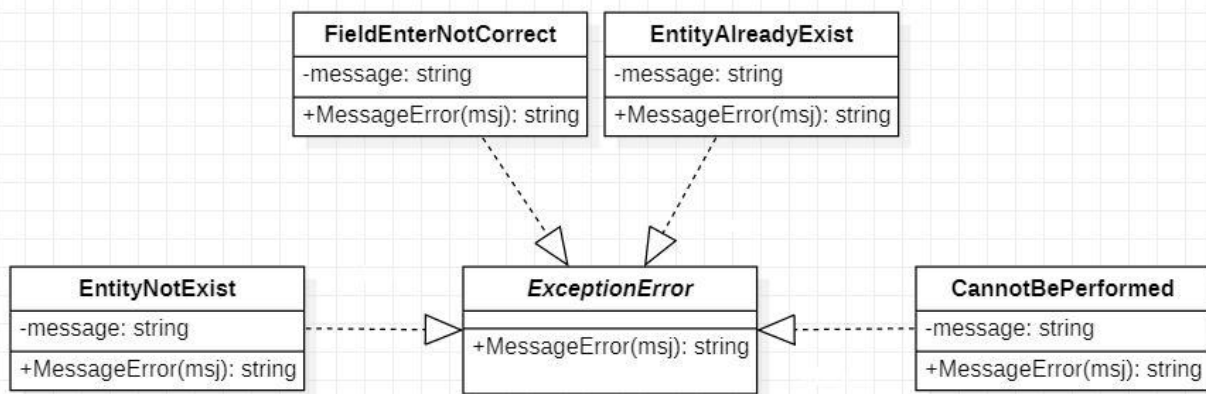
La nueva funcionalidad agregada es que cada usuario tiene que especificar la duración deseada de su consulta, que ocupa los posibles valores: [1h, 1.5h, 2h]. En este caso también decidimos hacer un enum para poder tener mayor escalabilidad y menor impacto en el cambio. Por otra parte, se agregó la funcionalidad para “bonificar” a los pacientes que tienen consultas regularmente. La idea es que una vez que un paciente logre tener cinco o más consultas, este pueda recibir un código por un cierto porcentaje de descuento en su próxima consulta. Se decidió agregar otro enum para el porcentaje de descuento, con la misma justificación que los anteriores.

MSP.BetterCalm.DTO

Los DTO son representaciones reducidas en cantidad de argumentos de nuestras clases de dominio, que nos permiten pasar únicamente los datos que se utilizan de dichos objetos, aumentando la performance.



MSP.BetterCalm.HandleError



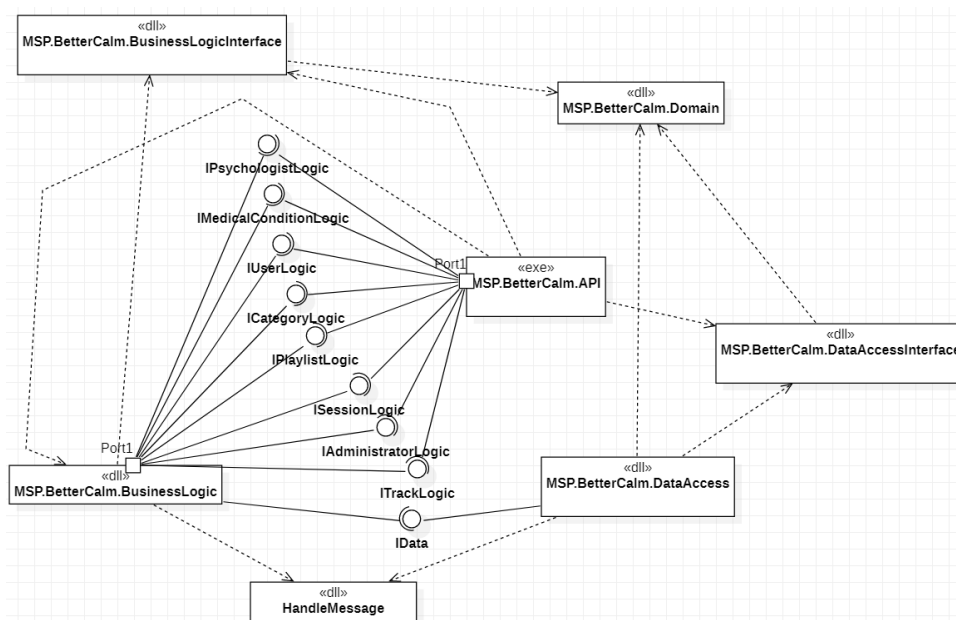
Para estandarizar las excepciones manejadas a lo largo de toda la aplicación decidimos crear un paquete con clases que representen cada uno de dichos errores con su respectivo mensaje. Las excepciones son transversales a toda la aplicación y son utilizadas por la lógica de negocio, la webapi y la capa de acceso a datos.

MSP.BetterCalm.Test

Paquete de pruebas. Se divide en paquetes particulares para cada uno de los paquetes descritos arriba (excepto las interfaces)

Diagrama de componentes

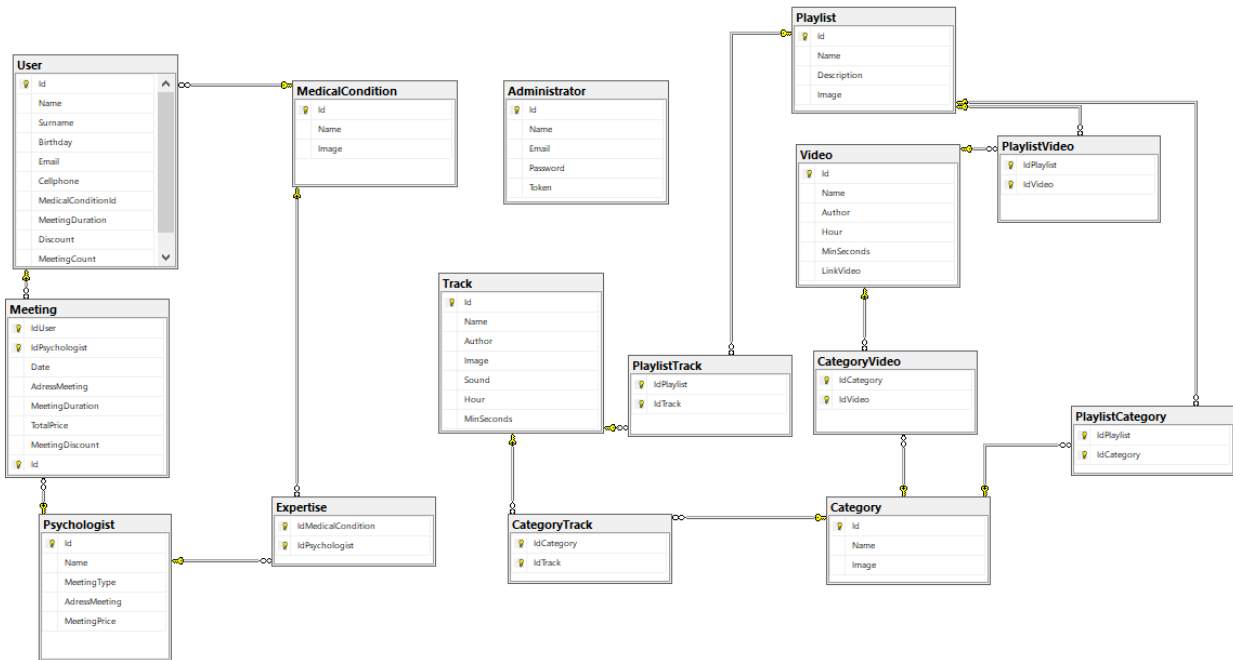
En el siguiente diagrama de componentes podemos observar cómo se comunican los distintos componentes mediante el uso de interfaces.



Modelo de tablas de la estructura de la base de datos

Este modelo de base de datos muestra la estructura lógica de la base, incluidas las relaciones y limitaciones que determinan cómo se almacenan los datos y cómo se accede a ellos.

El modelo de tablas de la estructura de la base de datos quedó de la siguiente manera:

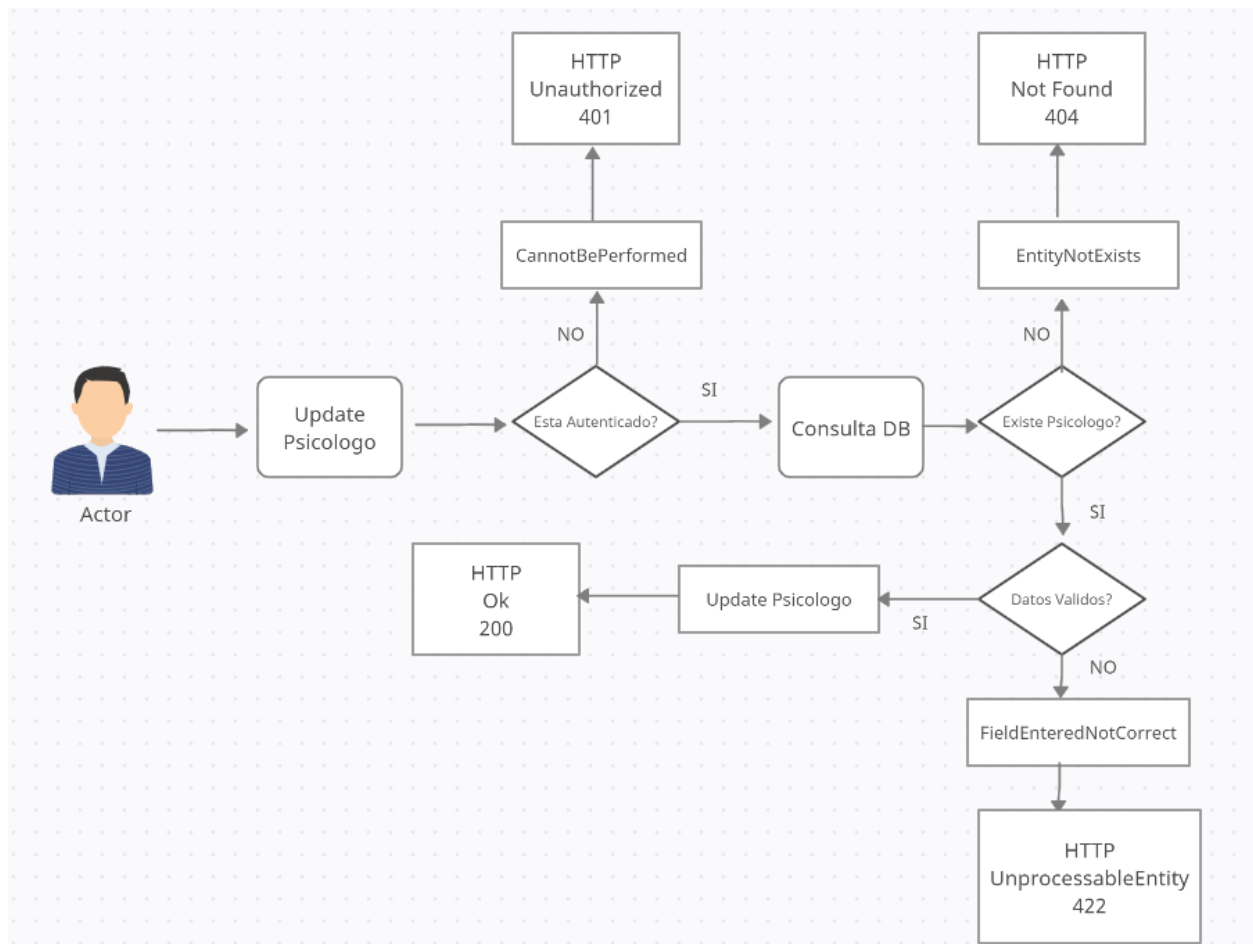


4. Vista de proceso

Representa los aspectos dinámicos de nuestro sistema, cómo funcionan los procesos y cómo se comunican entre ellos. Vamos a explicar mediante un diagrama de actividad para representar una funcionalidad para administradores en nuestro sistema. Si bien por motivos de tiempo representaremos solo un flujo en la documentación, cabe aclarar que en general la mayoría de los procesos de nuestra aplicación funcionan de esa forma. Una request a una api, la cual puede o no requerir permisos de administrador, seguida de validaciones de campos a cargo de la lógica de negocio de la entidad que representa al recurso y consultas a la base de datos para verificar la existencia y generar retorno de los datos o en otro caso mensajes HTTP con códigos de errores pertinentes, las excepciones son lanzadas en la lógica de negocio y utilizadas por la api para enviar mensajes de error acordes al caso.

Diagrama de Actividad

Realizamos diagrama de actividad de la funcionalidad actualizar psicólogo. Utiliza distintos tipos de excepciones y respuestas HTTP, 401 para cuando el usuario no está autenticado 404 para cuando el psicólogo que queremos actualizar no existe, 422 para cuando los campos ingresados no son correctos y 200 cuando todo se realizó correctamente.

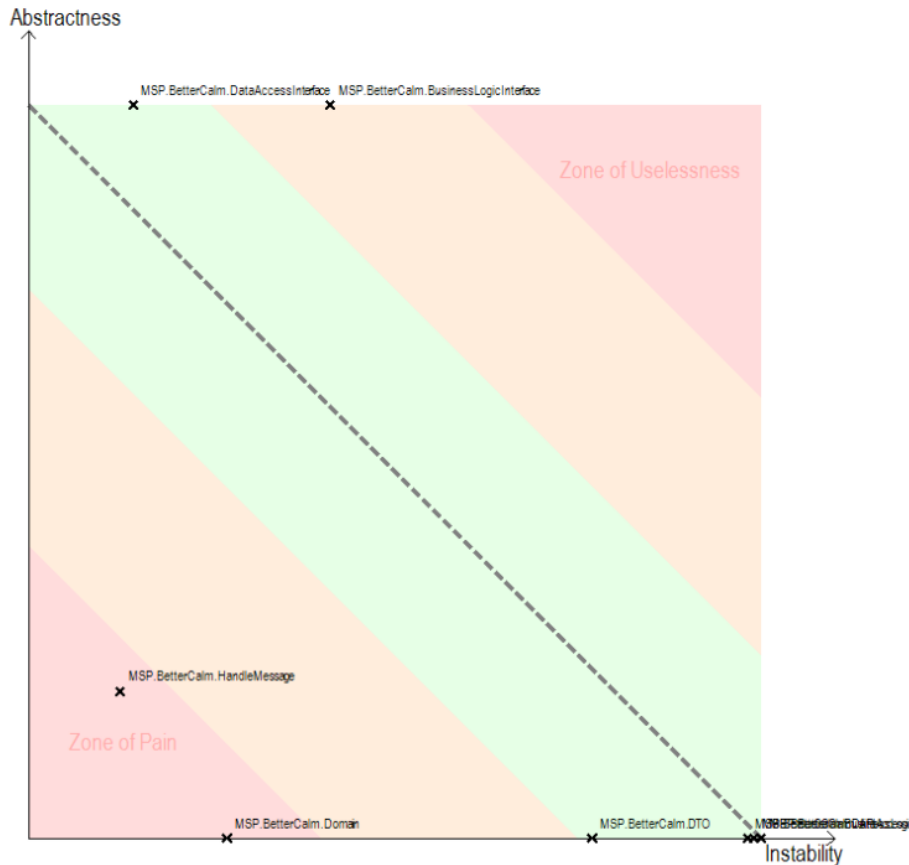


5. Patrones de diseño y métricas

Nombre patrón utilizado	Objetivo	Justificación de implementación
Alta cohesión	Tener un sistema fácil de mantener, donde ante nuevos cambios no haya que recurrir a cambios en distintas partes de la solución.	Agrupando en paquetes las clases que tengan relación lógica entre sí e implementando en clases los métodos que realicen tareas relacionadas.
Bajo acoplamiento	Minimizar el impacto del cambio. Aumentar la reutilización.	Agrupando responsabilidades de los módulos y aplicando otros patrones (DIP) para depender de abstracciones y no de implementaciones.
Inyección de dependencias	Lograr hacer intercambiables las implementaciones concretas.	Haciendo que las clases utilicen interfaces y asignando las implementaciones en la clase Startup.cs.
Inversión de dependencias	Depender de interfaces y no de implementaciones, para lograr código más sencillo de cambiar.	Para cada clase de lógica creamos una interface, la cual define el comportamiento que debe tener la clase que la implementa. De esta forma logramos que las clases que las utilicen trabajen con interfaces y no con clases concretas. De esta forma, si cambia la implementación, solo cambia la clase que implementa la interfaz y no las que la utilizan.
Sustitución de Liskov	Poder utilizar las implementaciones como si fuesen clases padre.	Si la implementación cumple el contrato de la interfaz podemos asegurar que es sustituible por la interfaz que implementa
Clausura común y SRP	Obtener paquetes cohesivos y	La descripción de los paquetes

	mantenibles.	explicada más arriba muestra que por ejemplo en caso de cambiar la capa de acceso a datos, deberíamos modificar todas las clases del paquete DataAccess pero ninguno más, lo mismo en caso de la lógica de negocio (BusinessLogic). Cada paquete es responsable de una parte de la aplicación, que si cambia, afecta solo a dicho paquete. Lo mismo sucede con las clases, las cuales tienen una única responsabilidad y razón para cambiar
ADP. Principio de dependencias acíclicas	Tener únicamente dependencias necesarias entre paquetes	Podemos observar en el diagrama de paquetes que no existen ciclos de dependencia entre los mismos. Nos ayudó hacer uso del patrón DIP para generar intermediarios en las dependencias de los paquetes.
Proxy remoto	Acceder a nuestra capa de acceso a datos desde el frontend. Representar localmente un objeto que se encuentra en un servidor remoto.	Creamos un servicio para cada uno de los recursos de la API.

Abstracciones estables



Utilizamos NDepend, una herramienta para calcular las métricas en función del código.

La abstracción es una medida de la rigidez de un sistema de software. A mayor abstracción, menor rigidez (o mayor flexibilidad) y viceversa. Si los componentes del sistema dependen de clases o interfaces abstractas, dicho sistema es más fácil de extender y cambiar que si dependiera directamente de clases concretas.

La estabilidad es una medida de tolerancia al cambio, así como de qué tan bien el sistema de software permite cambios sin romperlo. Esto se determina analizando las interdependencias de los componentes del sistema.

Ca: Acoplamientos aferentes: El número de clases fuera de esta categoría que dependen de las clases dentro de esta categoría.

Ce: Acoplamientos eferentes: El número de clases dentro de esta categoría que dependen de clases fuera de estas categorías.

I: Inestabilidad: $(Ce \div (Ca + Ce))$: Esta métrica tiene el rango [0,1]. I = 0 indica una categoría máximamente estable. I = 1 indica una categoría máximamente inestable.

A: Abstracción: ($\# \text{ clases abstractas en la categoría} \div \text{total } \# \text{ de clases en la categoría}$). Este rango métrico es [0,1]. 0 significa concreto y 1 significa completamente abstracto.

A continuación veremos estos datos para nuestros paquetes:

Nombre de paquete	Abstraccion	Inestabilidad	Distancia	¿Concreto o abstracto?	¿Estable o inestable?
MSP.BetterCalm.Handle Message	0.2	0.12	0.48	Concreto	Estable
MSP.BetterCalm.Domain	0	0.27	0.52	Concreto	Estable
MSP.BetterCalm.DTO	0	0.77	0.16	Concreto	Inestable
MSP.BetterCalm.BusinessLogic	0	0.98	0.01	Concreto	Inestable
MSP.BetterCalm.DataAccess	0	0.99	0.01	Concreto	Inestable
MSP.BetterCalm.API	0	1	0	Concreto	Inestable
MSP.BetterCalm.DataAccessInterface	1	0.14	0.1	Abstracto	Estable
MSP.BetterCalm.BusinessLogicInterface	1	0.41	0.29	Abstracto	Más estable que inestable

Interpretación de resultados

MSP.BetterCalm.HandleMessage

Es concreto (0.2) y estable (0.12). Este paquete contiene clases que representan los distintos tipos de excepción, las cuales no dependen de ninguna otra clase ya que su existencia tiene sentido de forma independiente. Las excepciones son utilizadas a lo largo de toda la aplicación, por eso se encuentran en la zona de estabilidad. Un cambio en alguna de las excepciones nos implicaría realizar cambios en varios lugares de la aplicación, como por ejemplo en la BusinessLogic, donde se crean las excepciones para lanzarlas, o en la WebApi, donde se capturan. Valor de distancia: 0.48

MSP.BetterCalm.Domain

Es concreto (0) y estable (0.27). Si bien lo ideal es que los paquetes concretos sean inestables, en caso de estos paquetes que representan entidades, las mismas no dependen de ninguna otra clase. No las afecta el cambio de ninguna clase del sistema, pero su cambio afecta a muchas clases, por lo que el costo de modificarlo es alto. Por estos motivos podemos observar en la zona de dolor roja de la gráfica. Valor de distancia: 0.52

MSP.BetterCalm.BusinessLogic:

Es concreto (0) e inestable (0.98). Al igual que la implementación de DataAccess, la implementación concreta de la lógica de negocios es un paquete concreto e inestable, ya que sigue el comportamiento de sus interfaces definidas en BusinessLogicInterface, ninguna clase depende de ella directamente ya que las dependencias son hacia interfaces y no hacia clases concretas (nuevamente, mediante inyección e inversión de dependencias). Creemos que las métricas son correctas y no es pertinente realizar cambios en este paquete. Valor de distancia: 0.01.

MSP.BetterCalm.DataAccess

Es concreto (0) e inestable (0.99). Al ser un paquete inestable y concreto, es una fiel representación de una implementación concreta de la capa de acceso a datos, la idea es que poca gente dependa de él y que de esta forma sea fácil de cambiar. En este caso, el objetivo de invertir las dependencias es que justamente las implementaciones concretas sean fácilmente intercambiables, por lo que consideramos que es una buena métrica y la clase está bien implementada. Valor de distancia 0.01

MSP.BetterCalm.Api

Es concreto (0) e inestable (1). Al ser el punto de acceso al mundo exterior y la parte final de nuestro flujo desde la base de datos, ninguna clase depende de la WebApi, por eso es un paquete concreto, del cual nadie depende y si cambiamos este paquete, ninguna clase va a tener que ser modificada. Creemos que las métricas son correctas y no es pertinente realizar cambios en este paquete. Valor de distancia: 0

MSP.BetterCalm.DataAccessInterface

Es abstracto (1) y estable (0.14). Si bien este paquete posee Interface en su nombre ya que contiene la IData, lugar donde se definen los métodos de la capa de acceso a datos. Sus valores se encuentran dentro de lo esperado (abstracto y estable). Este paquete fue diseñado así por motivos de responsabilidades. Valor de distancia 0.1.

MSP.BetterCalm.BusinessLogicInterface

Es abstracto (1) y más estable que inestable (0.41). Este paquete únicamente contiene interfaces ya que es el responsable de definir los contratos para la lógica de negocio. La lógica de negocio se utiliza en muchas clases de la WebApi, ya que es la responsable de la implementación de las funcionalidades y actuar como nexo entre la WebApi y la capa de acceso a datos. Por este motivo si bien es abstracta se encuentra en un punto medio en impacto de cambio. Valor de distancia: 0.29.

Interpretaciones de valores de distancia

Los valores de distancia de los paquetes son aceptables en general ya que ninguno supera el 0.7 que es el valor máximo recomendado, en nuestro caso el máximo que tuvimos fue 0.52 . De todas formas, como podemos ver en la gráfica, sí poseemos algunos paquetes dentro de la zona de dolor. Creemos que es bastante difícil evitar tener paquetes en dicha zona, ya que la representación de las entidades de la aplicación en general no va a depender de otros paquetes pero va a ser utilizada en diversos puntos de la aplicación, por lo que su cambio indudablemente va a impactar a muchas clases dentro de otros paquetes. Luego de eso, poseemos paquetes con distancia 0, los cuales son las implementaciones concretas de las

funcionalidades de nuestra aplicación, son muy concretos y muy inestables, ya que dependen de muchos otros paquetes y cambios en casi cualquier paquete de la solución los va a afectar. Finalmente, tenemos paquetes casi 100% abstractos, los cuales son paquetes que contienen las interfaces que definen los comportamientos de nuestro sistema y nos permiten utilizar la composición.

Dependencias estables

Ahora que conocemos la estabilidad de cada uno de nuestros paquetes, podemos verificar que se cumpla el patrón de dependencias estables, que nos indica que un paquete debe depender solo de paquetes que son más estables que el. Con ayuda del diagrama de paquetes vamos a verificar si se cumple esta regla.

Paquete a evaluar	Inestabilidad	Dependencias	Inest. Dependencia.	¿Es correcto?
MSP.BetterCalm.HandleMessage	0.12	Ninguna		
MSP.BetterCalm.Domain	0.27	Ninguna		
MSP.BetterCalm.DTO	0.77	MSP.BetterCalm.Domain	0.27	Si
MSP.BetterCalm.BusinessLogic	0.98	MSP.BetterCalm.HandleMessage	0.12	Si
		MSP.BetterCalm.DTO	0.77	Si
		MSP.BetterCalm.BusinessLogicInterface	0.41	Si
		MSP.BetterCalm.DataAccessInterface	0.14	Si
MSP.BetterCalm.DataAccess	0.99	MSP.BetterCalm.DataAccessInterface	0.14	Si
		MSP.BetterCalm.Domain	0.27	Si

		MSP.BetterCalm.HandleMessage	0.12	Si
MSP.BetterCalm.API	1	MSP.BetterCalm.BusinessLogic	0.98	*Discusión
		MSP.BetterCalm.DataAccess	0.99	*Discusión
		MSP.BetterCalm.DTO	0.77	Si
		MSP.BetterCalm.HandleMessage	0.12	Si
		MSP.BetterCalm.BusinessLogicInterface	0.41	Si
		MSP.BetterCalm.DataAccessInterface	0.14	Si
MSP.BetterCalm.DataAccessInterface	0.14	Ninguna		
MSP.BetterCalm.BusinessLogicInterface	0.41	MSP.BetterCalm.Domain	0.27	Si

*Discusión:

Podemos ver un primer punto en la aplicación en donde no se cumple el principio de dependencias estables. Esto se debe a que tanto BusinessLogic como DataAccess son paquetes con clases concretas que se encargan de la implementación de la lógica de negocio y la capa de acceso a datos respectivamente.

WebApi por su parte, se encarga de exponer al mundo exterior la lógica interna de nuestra aplicación, por lo que necesita depender de algún paquete concreto que devuelva o procese datos de la realidad. No podría depender únicamente de abstracciones ya que de esta forma no habría ninguna implementación que mostrar, por lo que consideramos que esta desviación en los parámetros esperados de las métricas es aceptable.

En general se logró cumplir con la regla que dice que los paquetes siempre deben depender de paquetes más estables. Creemos que logramos esto cumpliendo correctamente con el principio de inversión de dependencias, que dice que las clases deben depender de abstracciones, y a su vez como vimos más arriba, las abstracciones deben tender a ser más estables, por lo que las clases al depender de abstracciones logran depender de paquetes más estables.

6. Cambios con respecto a la primer entrega

Se agregaron nuevas funcionalidades:

Video

Se solicita agregar un nuevo tipo de contenido reproducible. Se solicita mantenimiento de una nueva entidad llamada video en la cual también se podría reproducir en video el contenido desde la página, pero sin redirigir a otro sitio. En esta implementación como se mencionó anteriormente por un tema de tiempos, decidimos no aplicar herencia, y se justificó en esa sección. Para implementar la nueva funcionalidad debimos agregar una nueva clase de Video en nuestro dominio. También tuvimos que agregar la implementación en el repositorio, la lógica de los métodos para el mantenimiento y por último el controlador, realizamos la migración correspondiente para actualizar la base de datos.

Descuento con usuario y psicólogo.

Psychologist

Se agregaron nuevos atributos al psychologist para poder lograr cierto costo a las meeting. Específicamente un enum ya que los precios son fijos y otro enum más para los horarios que también por el momento son fijos, pero nuestra implementación fue creada de manera que si a futuro se desea modificar o agregar nuevos horarios o precios, los mismos se pueden introducir con un impacto muy pequeño en nuestro sistema. De esta manera al agendar una meeting se podrá calcular el precio que tiene para el usuario.

User

Tiene varios atributos para poder cumplir con la funcionalidad del discount. Entre ellos un enum discount, el cual también fue pensado para realizar cambios futuros, y que el mismo tenga el menor impacto posible, que luego es aplicado al precio de la meeting. También otro atributo meeting count por el cual se va a contar las meeting antes de que el administrador aplique descuento. Por último creamos un nuevo método por el cual el administrador lo utilizara para updatear el usuario con su nuevo discount.

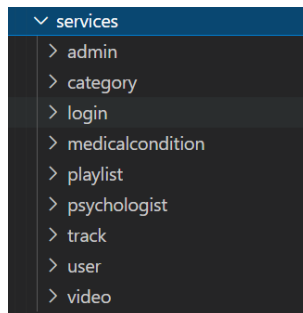
Por último se agregó frontend, utilizando angular para mostrarla por la web. En el punto 7 a continuación se abarca más sobre el tema.

7. Características del frontend

Frontend, la aplicación fue desarrollada en Angular. Planteamos una solución basada en componentes, los cuales se compone cada uno de un archivo .html que se encarga de la vista, otro .ts que se encarga de la lógica del componente y de ser necesario un archivo .css para los estilos. En ocasiones no fue necesario utilizar un archivo CSS ya que utilizamos la librería Bootstrap 5 para darle estilos a nuestra aplicación, la cual fue instalada, ofrece estilos relativamente buenos y facilidad de uso. Dividimos nuestra solución en 3 carpetas principales:

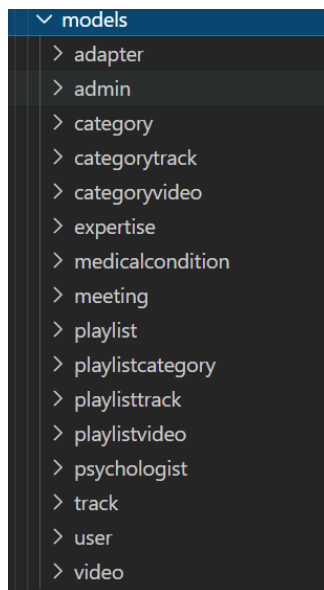
- **Services**

Carpeta encargada de contener todos los servicios para la capa de acceso a datos. En este punto es donde se utiliza el patrón proxy remoto, siendo estas clases una capa de acceso a datos para acceder a los endpoints de la API. Consideramos correcto generar una capa de acceso a datos separada de la lógica de cada componente para tener código más limpio y reutilizable.



- **Models**

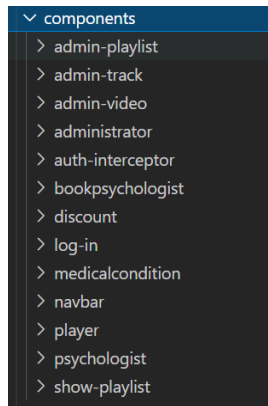
La carpeta models representa las distintas entidades que van a ser manejadas en nuestra aplicación. Son la representación de las clases del dominio o DTOs en el frontend. Son objetos que se utilizan para el manejo de entidades.



- **Components**

Los componentes representan pantallas principales, las cuales contienen muchos componentes más pequeños dentro. La utilización de componentes no solo nos permitió reutilizar código, sino que también tener componentes con los mismos estilos a lo largo de toda la aplicación, generando una mejor interfaz de usuario. Por otra parte, creamos un componente navbar,

encargado de manejar las rutas de la aplicación, para las cuales utilizamos la librería AppRoutingModuleModule. Decidimos utilizar rutas para poder acceder a puntos de nuestra página web, mediante la url. Si bien intentamos componentizar lo más posible los distintos elementos de la aplicación, admitimos que algunos componentes quedaron un poco más grandes de lo que nos hubiese gustado, violando SRP.



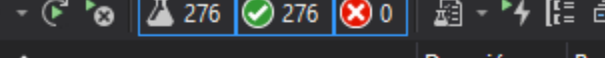
Anexo

Cobertura de pruebas

En cuanto a la cobertura de pruebas, obtuvimos un porcentaje de 89,71% en promedio cumpliendo con lo solicitado. En la api nos muestra un porcentaje menor debido (Startup y program). Si bien realizamos TDD para el desarrollo de la aplicación, a último momento surgieron algunos cambios que impactan la cobertura del paquete de BusinessLogic .

Jerarquía	No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
mauro_LAPTOP-00S4OKL9 2021...	620	10,29 %	5404	89,71 %
msp.bettercalm.api.dll	80	33,33 %	160	66,67 %
msp.bettercalm.apitest.dll	0	0,00 %	16	100,00 %
msp.bettercalm.businesslog...	386	24,79 %	1171	75,21 %
msp.bettercalm.businesslog...	42	3,17 %	1285	96,83 %
msp.bettercalm.dataaccess...	52	5,05 %	977	94,95 %
msp.bettercalm.dataaccess...	1	0,15 %	672	99,85 %
msp.bettercalm.domain.dll	53	13,49 %	340	86,51 %
msp.bettercalm.domaintest...	0	0,00 %	751	100,00 %
msp.bettercalm.dto.dll	3	15,79 %	16	84,21 %
msp.bettercalm.handlemess...	3	15,79 %	16	84,21 %

Si bien podemos ver algunos bloques que no se alcanzaron a cubrir totalmente, seguramente fue por algún test no especificado e implementado al 100 %, sobre todo en las nuevas implementaciones.



Prueba	Duración	Rasgos
✓ MSP.BetterCalm.APITest (72)	557 ms	
✓ MSP.BetterCalm.BusinessLogicTest (70)	1,6 s	
✓ MSP.BetterCalm.DataAccessTest (28)	2,9 s	
✓ MSP.BetterCalm.DomainTest (106)	84 ms	

Descripción de la api

Códigos de error

Se utilizaron los siguientes códigos de errores:

Code	Description
200	Ok. Return the request object
401	Unauthorized. You do not have permissions to perform this action
404	NotFound. The requested object was not found
422	Unprocessable. Error in the semantics
500	InternalServerError. The server could not handle an exception in the system.

Nos permite indicar errores concretos, con mensajes para los usuarios a fin de comprender la situación actual de su petición. Los mismos se pueden observar en la documentación luego de realizar algún endpoint.

Controladores

De acuerdo a las funcionalidades solicitadas en la letra del problema planteado optamos por nueve controladores en nuestro sistema. Cada uno para las diferentes funcionalidades con cada recurso y serán presentados a continuación:

BetterCalm <small>v1 OAS3</small> <small>/swagger/v1/swagger.json</small>	
Administrator	>
Category	>
MedicalCondition	>
Playlist	>
Psychologist	>
Session	>
Track	>
User	>
Video	>

VideoController

También por letra indica que el administrador debe realizar alta y baja de un nuevo contenido reproducible en este caso video de este modo es que decidimos incluir este controlador. También el usuario puede navegar entre los videos.

Descripción de los resources

Video

GET /api/video/{id} Get video by id

Parameters Try it out

Name	Description
id * required integer(\$int32) (path)	<input type="text" value="id"/>

Responses

Code	Description	Links
200	OK. Returns the requested object.	No links
404	NotFound. The requested object was not found.	No links
500	InternalServerError. The server could not handle an exception in the system.	No links

PUT /api/video/{id} Update a video by id

Parameters Try it out

Name	Description
id * required integer(\$int32) (path)	<input type="text" value="id"/>

Request body application/json-patch+json

Example Value | **Schema**

```
{
  "id": 0,
  "name": "string",
  "author": "string",
  "hour": 0,
  "minSeconds": 0,
  "linkVideo": "string",
  "categoryVideo": [
    {
      "idCategory": 0,
      "category": {
        "id": 0,
        "name": "string",
        "image": "string",
        "categoryTrack": [
          {
            "idCategory": 0,
            "category": "string",
            "idTrack": 0,
            "track": {
              "id": 0,
              "name": "string",
              "author": "string",
              "image": "string",
              "hour": 0,
              "minSeconds": 0,
              "sound": "string",
              "categoryTrack": [
                "string"
              ]
            }
          }
        ]
      }
    }
  ]
}
```

Responses

Code	Description	Links
200	OK. Returns the requested object.	No links
401	Unauthorized. You do not have permissions to perform this action.	No links
404	NotFound. The requested object was not found.	No links
422	UnprocessableEntity. Error in the semantics.	No links
500	InternalServerError. The server could not handle an exception in the system.	No links

DELETE
/api/video/{id}
Delete a video by id

Parameters

Try it out

Name	Description
id * required integer(\$int32) (path)	<div>id</div>

Responses

Code	Description	Links
200	OK. Returns the requested object.	No links
401	Unauthorized. You do not have permissions to perform this action.	No links
404	NotFound. The requested object was not found.	No links
500	InternalServerError. The server could not handle an exception in the system.	No links

POST
/api/video
Add a video

Parameters

Try it out

No parameters

Request body

application/json-patch+json

Example Value

Schema

```

{
  "id": 0,
  "name": "string",
  "author": "string",
  "hour": 0,
  "minSeconds": 0,
  "linkVideo": "string",
  "categoryVideo": [
    {
      "idCategory": 0,
      "category": {
        "id": 0,
        "name": "string",
        "image": "string",
        "categoryTrack": [
          {
            "idCategory": 0,
            "category": "string",
            "idTrack": 0,
            "track": {
              "id": 0,
              "name": "string",
              "author": "string",
              "image": "string",
              "hour": 0,
              "minSeconds": 0,
              "sound": "string",
              "categoryTrack": [
                "string"
              ]
            }
          }
        ]
      }
    }
  ]
}

```

Responses

Code	Description	Links
200	OK. Returns the requested object.	No links
401	Unauthorized. You do not have permissions to perform this action.	No links
404	NotFound. The requested object was not found.	No links
422	UnprocessableEntity. Error in the semantics.	No links
500	InternalServerError. The server could not handle an exception in the system.	No links

GET /api/video Get all the videos		
Parameters		Try it out
No parameters		
Responses		
Code	Description	Links
200	OK. Returns the requested object.	No links
500	InternalServerError. The server could not handle an exception in the system.	No links

User

GET /api/user Get all the users with more than 5 meeting		
Parameters		Try it out
No parameters		
Responses		
Code	Description	Links
200	OK. Returns the requested object.	No links
500	InternalServerError. The server could not handle an exception in the system.	No links

PUT /api/user/{id} Update a user by id - discount

[Try it out](#)

Parameters

Name	Description
id <small>* required</small> integer(int32) (path)	id

Request body application/json-patch+json ▾

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "surname": "string",
  "birthday": "2021-06-17T17:52:20.586Z",
  "email": "string",
  "cellphone": "string",
  "meeting": [
    {
      "id": 0,
      "user": "string",
      "idUser": 0,
      "psychologist": {
        "id": 0,
        "name": "string",
        "meetingType": "",
        "meetingPrice": 500,
        "addressMeeting": "string",
        "expertise": [
          {
            "idMedicalCondition": 0,
            "medicalCondition": "string",
            "idPsychologist": 0,
            "psychologist": "string"
          }
        ],
        "meeting": [
          "string"
        ]
      }
    }
  ]
}
```

Responses

Code	Description	Links
200	OK. Returns the requested object.	No links
401	Unauthorized. You do not have permissions to perform this action.	No links
404	NotFound. The requested object was not found.	No links
422	UnprocessableEntity. Error in the semantics.	No links
500		No links

Datos de prueba

Video

Results Messages						
	Id	Name	Author	Hour	MinSeconds	LinkVideo
1	1	Gasolina	Daddy Yankee	1	0	https://www.youtube.com/embed/Ugl4Omo2rRw
2	5	Fitness	desconocido	2	0	https://www.youtube.com/embed/Hlb4hRxiEq4
3	6	VideoDormir	desconocido	5	0	https://www.youtube.com/embed/an60NyBayzE

User

Results Messages										
	Id	Name	Surname	Birthday	Email	Cellphone	MedicalConditionId	MeetingDuration	Discount	MeetingCount
1	34	Usuario 1	usuario1	2000-01-01 00:00:00.0000000	usuario1@usuario1.com	0521984189	6	1	100	0
2	35	Rodrigo	Perez	2000-01-01 00:00:00.0000000	rohigon@hotmail.com	0521984189	2	2	100	0

Track

	Id	Name	Author	Image	Sound	Hour	MinSeconds
1	4	Despacito	Luis Fonsi	https://www.buenamusica.com/media/fotos/discos/l/luis-fo...	https://www.youtube.com/watch?v=kJQP7kiw5Fk	0	3,15
2	7	Gasolina	Daddy Yankee	https://upload.wikimedia.org/wikipedia/en/1/16/Daddy_Ya...	me gusta la gasolina	0	1,1
3	11	Zafar	Vela Puerca	https://4.bp.blogspot.com/_mnH3-39vdww/S3CiEgGbqwl/...	zafar.mp3	0	2,35
4	12	Va a escampar	Vela Puerca	https://1.bp.blogspot.com/-WmWOTyGC5LE/WGwjzCuW0...	escampar.mp3	1	1,07
5	13	Cero a la izquierda	No te va a gustar	https://e-cdns-images.dzcdn.net/images/cover/1e1cb14cc...	cero.mp3	2	0
6	14	EjercicioMusica	desconocido	https://i.scdn.co/image/ab67616d00001e02d892c1fe457...	musicaej.mp3	3	1,13

Psychologist

	Id	Name	MeetingType	AdressMeeting	MeetingPrice
1	29	Rodrigo Hirigoyen	1		500
2	30	Mauro Teixeira	1		500
3	32	Diego Perez	2	Rivera 1515	2000
4	33	Pepe	2	River pepe 1502	1000
5	34	Jose Jose	1		1000

Playlist

	Id	Name	Description	Image
1	7	Polaco	Lo mas nuevo	https://www.buenamusica.com/media/fotos/discos/e/el-p...
2	8	Daddy Yankee	Daddy Daddy	https://upload.wikimedia.org/wikipedia/en/1/15/Daddy-ya...
3	13	Tranquilidad	mozart	https://images.hungama.com/c/1/3b2/881/54275820/542...
4	17	Vela Puerca	viejas canciones	https://images.sk-static.com/images/media/profile_image...
5	18	No te va gustar	no me van a gustar	https://img.discogs.com/u3P2DmSWUBXtP4iOSGjh8EJ7...

Medical condition

	Id	Name	Image
1	1	Depresión	https://statics-cuidateplus.marca.com/sites/default/files/d...
2	2	Estrés	https://www.infobae.com/new-resizer/vdmKvP52mmjmq...
3	3	Ansiedad	https://statics-cuidateplus.marca.com/sites/default/files/...
4	4	Autoestima	https://www.webconsultas.com/sites/default/files/styles/w...
5	5	Enojo	https://assets.entrepreneur.com/content/3x2/2000/2019...
6	6	Relaciones	https://www.esmental.com/wp-content/uploads/2020/03/...
7	7	Duelo	https://www.terapify.com/blog/wp-content/uploads/2020/...
8	8	Otros	https://trascender.uy/wp-content/uploads/2021/01/Emoci...

Category

	Id	Name
1	5	Dormir
2	6	Meditar
3	7	Musica
4	8	Cuerpo

Administrator

	Id	Name	Email	Password	Token
1	1	admin	admin@admin.com	admin	0DC08A9D-2BD6-4DF4-9F21-843B3818895D
2	9	Admin 2	admin2@admin.com	123	C449E258-5646-41EC-A662-69F2A244D656

Meeting

	IdUser	IdPsychologist	Date	AdressMeeting	MeetingDuration	TotalPrice	MeetingDiscount	Id
1	35	29	2021-06-17 12:43:29.1273513	https://bettercalm.com.uy/meeting/8e78245f-941a-4...	2	500	50	6
2	35	30	2021-06-17 12:41:45.1550982	https://bettercalm.com.uy/meeting/91764f8f-8aaf-4d...	2	1000	100	0
3	35	30	2021-06-17 12:42:16.7384753	https://bettercalm.com.uy/meeting/d07755dd-be81-...	1	500	100	1
4	35	30	2021-06-17 12:42:17.5430594	https://bettercalm.com.uy/meeting/37044038-3719-...	1	500	100	2
5	35	30	2021-06-17 12:42:18.7385711	https://bettercalm.com.uy/meeting/94604f5f-f04f-4ea...	1	500	100	3
6	35	30	2021-06-17 12:42:21.2874601	https://bettercalm.com.uy/meeting/d41ad34d-ba3e-...	3	750	100	4
7	35	30	2021-06-18 12:42:22.1849002	https://bettercalm.com.uy/meeting/0b92bba7-279e-...	3	750	100	5