Universidad ORT Uruguay
Facultad de Ingeniería

Diseño de Aplicaciones 2
Obligatorio 1

Descripción del diseño

Diego Asadurian (198874)
Dominique Lachaise(214233)

Docentes: Gabriel Piaretti, Nicolas Fierro y Nicolas Hernandez

Declaraciones de autoría

Nosotros Diego Asadurian y Dominique Lachaise, declaramos que el trabajo que se presenta es este entregable es de nuestra autoridad, en lo cual podemos asegurar que:

- El entregable fue realizado en su totalidad mientras se cursaba la materia Diseño de Aplicaciones 2

Resumen

El presente documento se enmarca en un proyecto para implementar una WebApi para el Ministerio de Salud Pública. Debido a la situación sanitaria que estamos viviendo hoy en día el MSP decidió llevar a cabo una aplicación que le permita a los usuario poder realizar consultas con psicólogos según sus dolencias actuales como así también poder escuchar música según sus necesidades.
En esta primera etapa se diseñan los endpoints necesarios para ejecutar las funcionalidades solicitadas, para en una segunda entrega poder brindarle al cliente una interfaz con la cual poder interactuar.

A Continuación se detallan supuestos efectuados, el proceso de implementación y descripción de diseño, diagrama de paquetes, clases , interacción y componentes, modelo de tablas de la base de datos, etc.

Índice

Descripción del trabajo

Introducción

Como se mencionó anteriormente en esta primera etapa se diseñó el backend del proyecto MPS.BetterCalm el cual permitirá realizar consultas entre pacientes y psicólogos como así también permitirle al usuario escuchar música de su preferencia.

Implementación

El proyecto fue implementado en el lenguaje de programación C# utilizando el framework ASP .NET Core 3.1 dentro de la aplicación Visual Studio Code.
Cumpliendo con lo solicitado se utilizó un repositorio en GitHub para el control de versiones y a modo de prevenir errores en la rama develop, utilizamos SourceTree para llevar a cabo el manejo del repositorio. También se utilizó para la persistencia de las entidades Entity Framework Core 3.1.8 y para visualizar la misma Microsoft SQL Server Management
Studio 18.

Funciones no implementadas

Se logró culminar con todas las funcionalidades correctas, no pudiendo probar la totalidad de las mismas.

No se logró cumplir con toda la documentación solicitada, ya que no contamos con el tiempo suficiente para llevarla a cabo. Sobre todo los diagramas de interacción no pudieron ser llevados a cabo.

Bugs

De momento no encontramos bugs que impidan la utilización de la misma, si bien somos conscientes de que pueden aparecer algunos ya que por falta de tiempo no logramos culminar la ejecución completa de todas.

Supuestos efectuados

- No se puede realizar una consulta sino hay psicólogos ingresados en el sistema, que pertenezcan a esa patología.

Descripción y justificación del diseño

Arquitectura

Nuestro sistema fue dividido en ocho paquetes.

- **MSP.BetterCalm.BusinessLogic:** en este paquete es donde se encontrará toda la lógica de negocio del proyecto, en él se manejan todas las clases del dominio.
- **MSP.BetterCalm.BusinessLogic.Interface:** en este paquete se implementó una interfaz para cada clase del paquete BusinessLogic en las cuales se encargan de ser los manejadores de los métodos.
- **MSP.BetterCalm.Domain:** se encuentran todas las clases necesarias para poder implementar los requerimientos solicitados, cuyas funciones satisfacen los requerimientos de negocio.
- **MSP.BetterCalm.DataAccess:** en este paquete se llevó a cabo la conexión entre las diferentes clases de la lógica de negocio a la base de datos, implementando la persistencia**.**
- **MSP.BetterCalm.DataAccess.Interface:** en este paquete se implementó una interfaz para el Repository del Data Access
- **MSP.BetterCalm.Webapi:** en este paquete es donde están ubicados los controllers en la que se exponen los servicios (endpoints) para que puedan ser consumidos desde una página web mediante llamadas HTTP aplicando los diferentes verbos.

- **MSP.BetterCalm.BusinessLogic.Test:** en este paquete se llevó a cabo todos los test que consideramos necesarios para poder verificar que la logica de negocio cumplia con lo establecido.
- **MSP.BetterCalm.DataAccess.Test:** en este paquete se lleva a cabo todos los test que consideramos necesarios para testear el correcto desempeño de nuestro data access.

Las implementaciones en todos los paquetes fueron implementadas tratando de cumplir con la mayoría de los principales principios de diseño, los más cumplidos fueron SRP (Single responsability principle) y OCP (Open/Close principle), para así lograr tener un sistema que a futuro se puedan realizar modificaciones sin necesidad de tocar código ya escrito como así también lograr tener un sistema con alta cohesión y bajo acoplamiento.

Repositorio

El paquete MSP.BetterCalm.DataAccess es el que se comunica directamente con la base de datos. En dicho paquete es donde se llevan a cabo todos los métodos requeridos por los servicios de la lógica de negocio. En dicho paquete es donde se creó la clase DataContext la cual hereda de DbContext y es la encargada del mapeo de las tablas utilizando el ORM Entity Framwork Core 3.1.8.

Principales funcionalidades

Pacientes

Pueda reproducir playlist o audios

Para llevar a cabo esta funcionalidad no se necesita ningún control de acceso, el paciente puede ingresar libremente y navegar por las playlist o simplemente reproducir audios.
Por lo cual para llevar a cabo esta funcionalidad desarrollamos métodos en la lógica de Audio y Playlist los cuales se encargan de llevar a cabo ei proceso.
Se detalla a continuación el diagrama.

Agregar consultas con psicólogos

La creación de consultas es una funcionalidad clave ya que es la que tiene que conocer qué psicólogos están disponibles para la fecha que el paciente desea.
El paciente puede agendar consultas con psicólogos de acuerdo a su principal dolencia, acá lo que hicimos fue crearnos una clase Consultation qué es la que se va a encargar de guardar toda la información necesaria a la consulta. Para llevar a cabo este procedimiento es necesario consumir datos de la lógica del negocio del Psychologist ya que es la que se encarga de devolver el psicólogo especialista en dicha patología siempre y cuando tenga disponibilidad para atender el paciente.
Se detalla a continuación el diagrama.

Administradores

Iniciar sesión

Para poder manejar las funcionalidades que brinda el sistema es necesario que el administrador esté loggeado en el sistema y sea quien se encargue de registrar psicólogos, mantener los contenidos reproducibles y administrar los mismo administradores en caso de ser super admin.

La autenticación se realizó en base a tokens, con lo cual una vez que el administrador se loguea en el sistema, el mismo le asigna un token por esa sesión que se guarda en la base de datos seguida de un ID que corresponde al administrador que realizó el login.

Para llevar a cabo la funcionalidad de los admin, creamos una clase Administrator en el dominio, la cual cuenta con los siguientes atributos: Guid Id, string Name, string Email, string Password, bool IsActive. Los atributos como Password y Email fueron validados para

que no sean vacíos. El email tiene una validación la cual no permite ingresar un mail sin @ ni ingresar un mail que tenga más de 3 partes (gmail.com.edu.ey.ort) ni uno con menor a 2 partes, también se llevó a cabo un equals por email para que dos administradores no tengan el mismo Email.

Dar de alta y baja contenido reproducible

Para llevar a cabo dicha funcionalidad manejamos una tabla Audios y otra Category. Audios tiene una lista de categorías y categoría una lista de audios, entonces el administrador al querer ingresar un nuevo audio tiene que consumir la lógica que está en audio verificando antes de ingresar el audio a la categoría que dicho audio ya no pertenezca a la categoría.

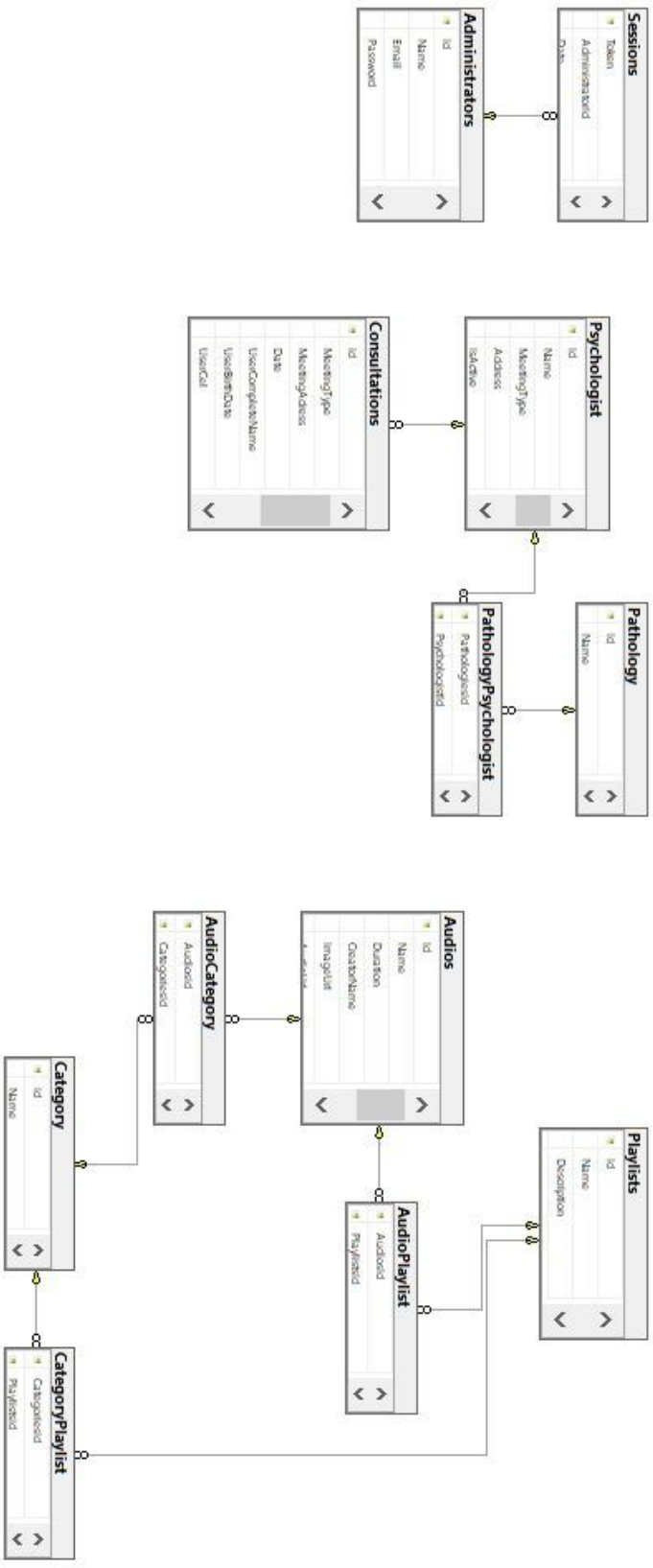
Mantenimiento de un psicólogo

Los administradores son los encargados de llevar a cabo el mantenimiento de los psicólogos, para ello definimos un método Update en la lógica el cual recibe un Guid Id y el psicólogo con sus nuevos datos, y en la clase Psychologist el cual recibe el “nuevo psicólogo” y le actualiza los datos correspondientes.

Mantenimiento de los administradores

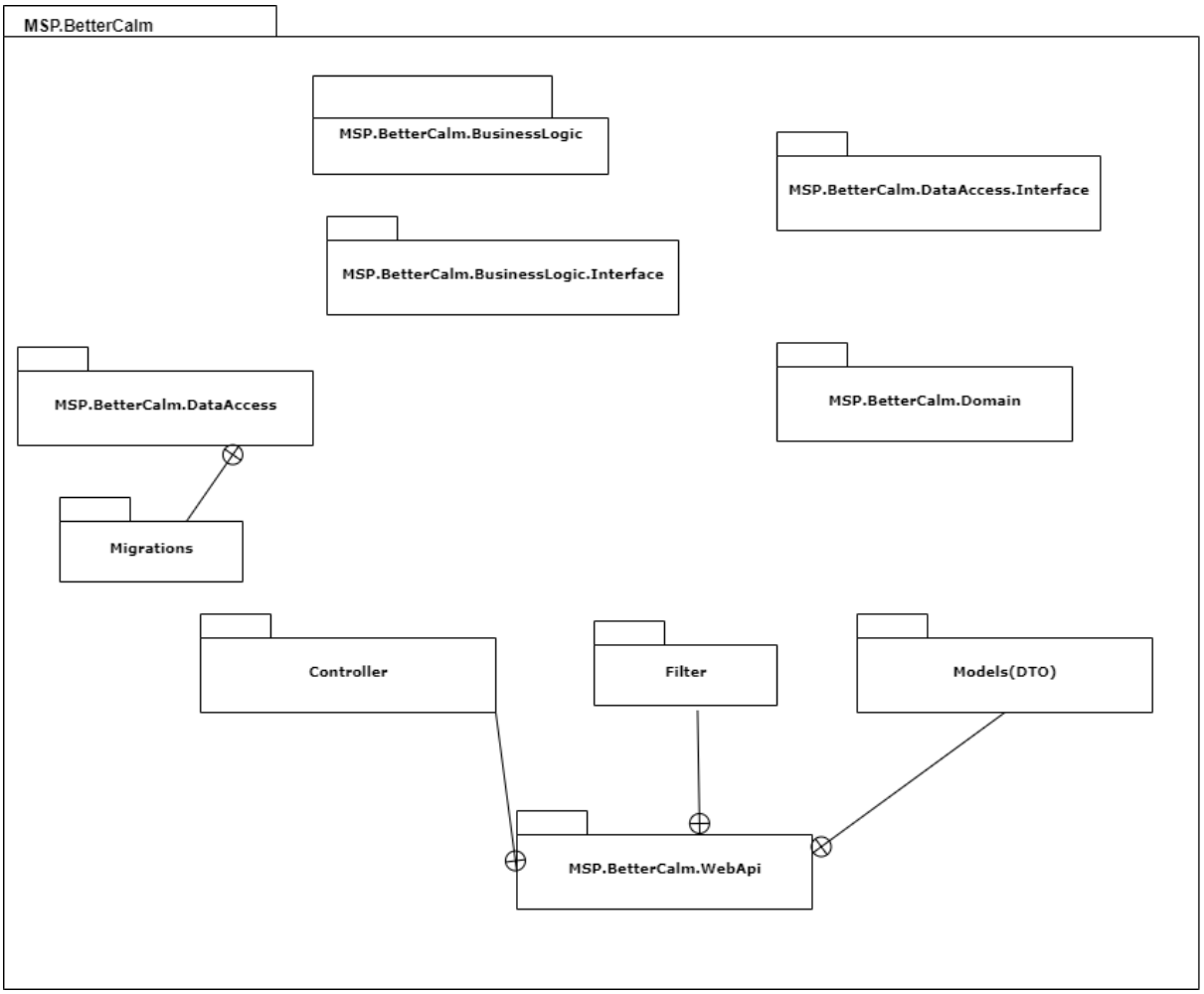
Una vez ingresado al sistema, se tiene un administrador ya cargado en la base de datos, el cual podrá dar de alta nuevos administradores y luego entre ellos podrán realizar sus correspondientes funcionalidades. Para llevar a cabo el proceso de actualizar administradores, definimos un método Update en la lógica del administrador el cual recibe un Guid id y el administrador con sus nuevos datos, y en la clase Administrator el cual recibe el “nuevo administrador” y le actualiza los datos correspondientes.

Modelo de tablas



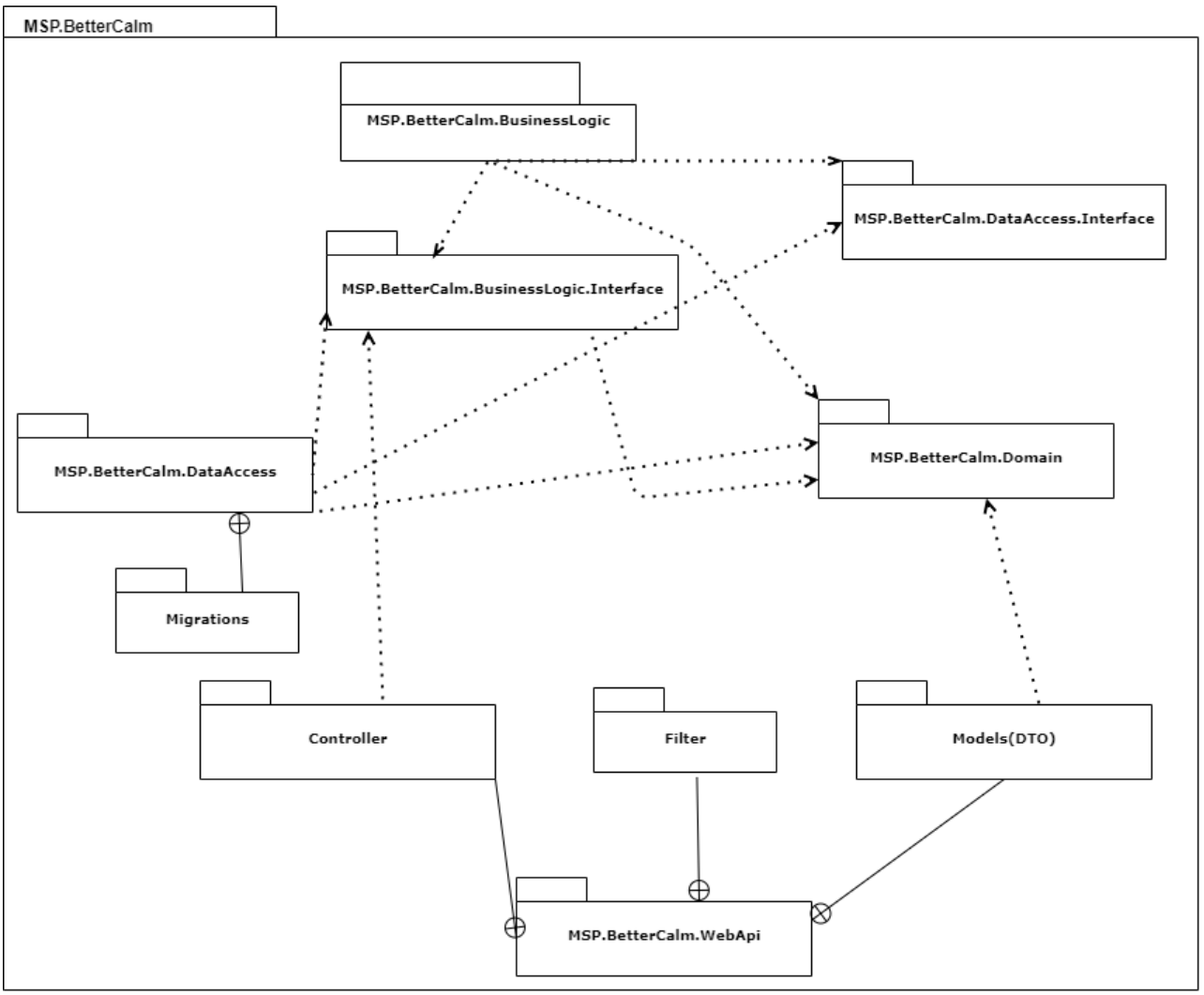
Para llevar a cabo la implementación de la persistencia se utilizó la herramienta Entity Framework y Microsoft SQL Server Management Studio 18. Nos basamos en la metodología Code-First ya que es la que utilizamos en DA1, y es con las que más nos familiarizamos, la cual se crean las tablas a partir del código.
Para cada operación se abre y cierra la conexión a la base de datos.

Diagrama de descomposición



Este diagrama intenta mostrar la jerarquía y organización de los paquetes sin sus dependencias.

Diagrama de paquetes

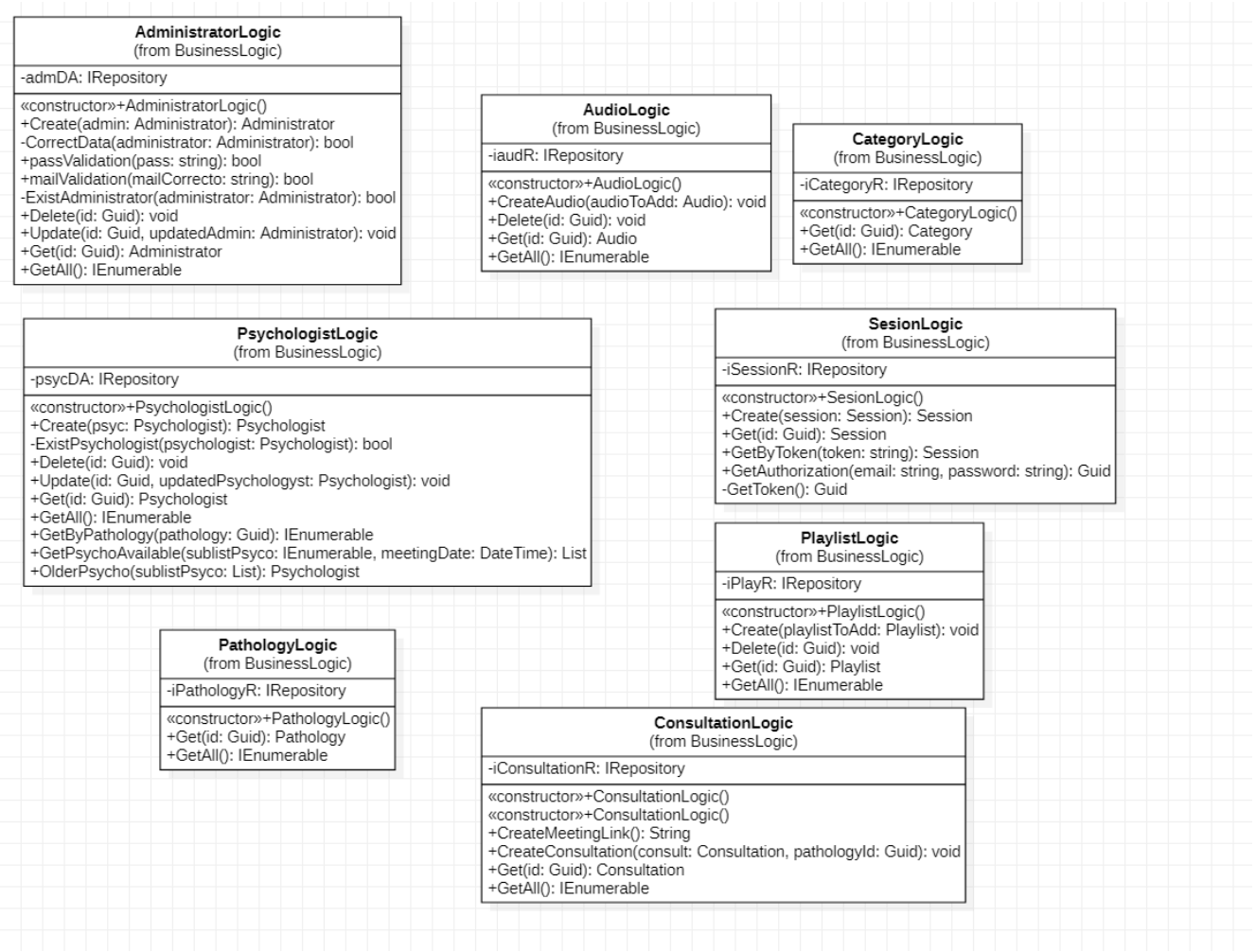


En la imagen se puede observar el diagrama de paquetes completo del obligatorio, cada paquete con sus correspondientes dependencias. También se puede observar los subpaquetes representados por el conector de nesting y se muestran las dependencias entre los mismos.
Lo que tratamos de llevar a cabo fue generar un diseño de bajo acoplamiento, con una alta cohesión entre los paquetes.

Diagrama de clases

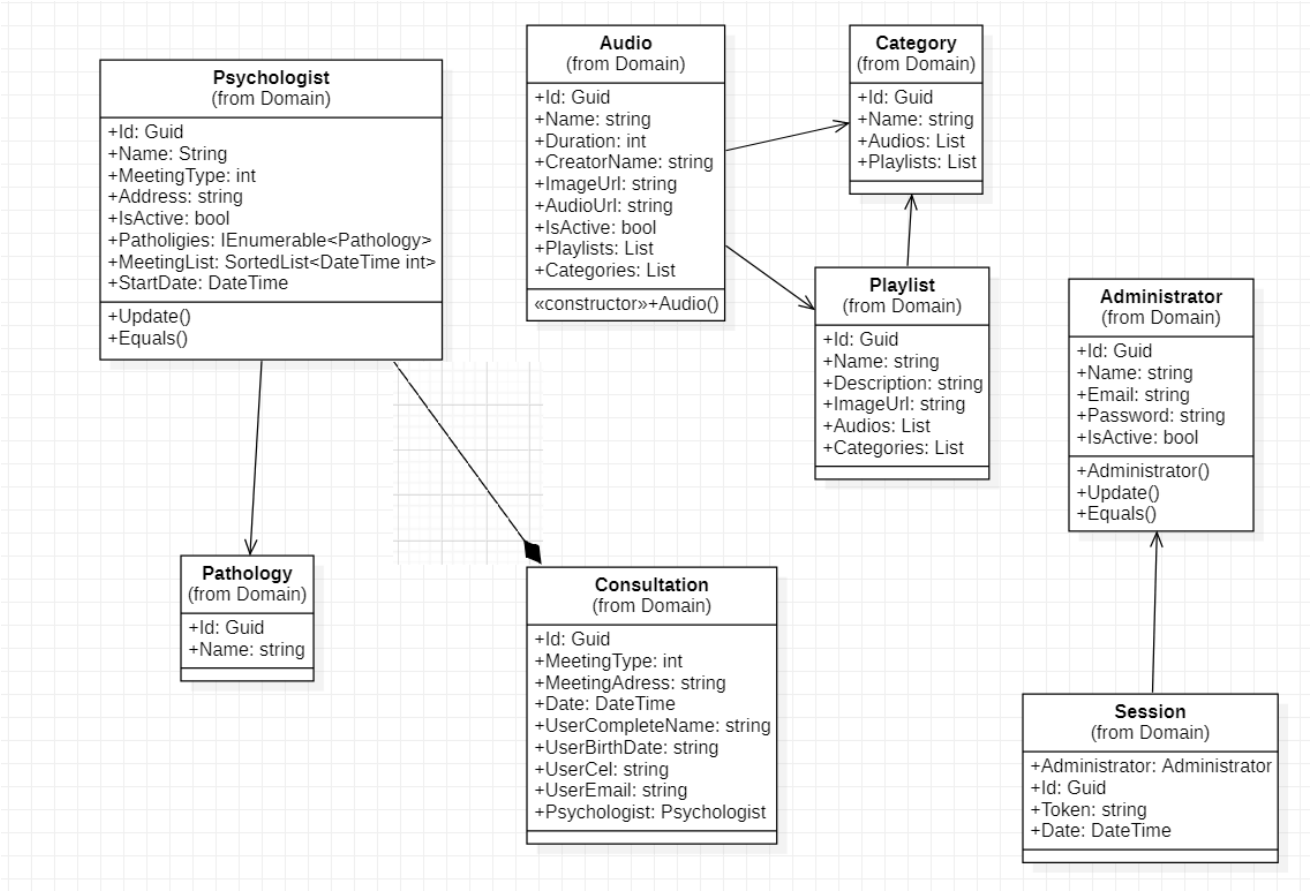
En esta sección llevaremos a cabo los diagramas de clase del obligatorio.

Diagrama de MSP.BetterCalm.Business Logic



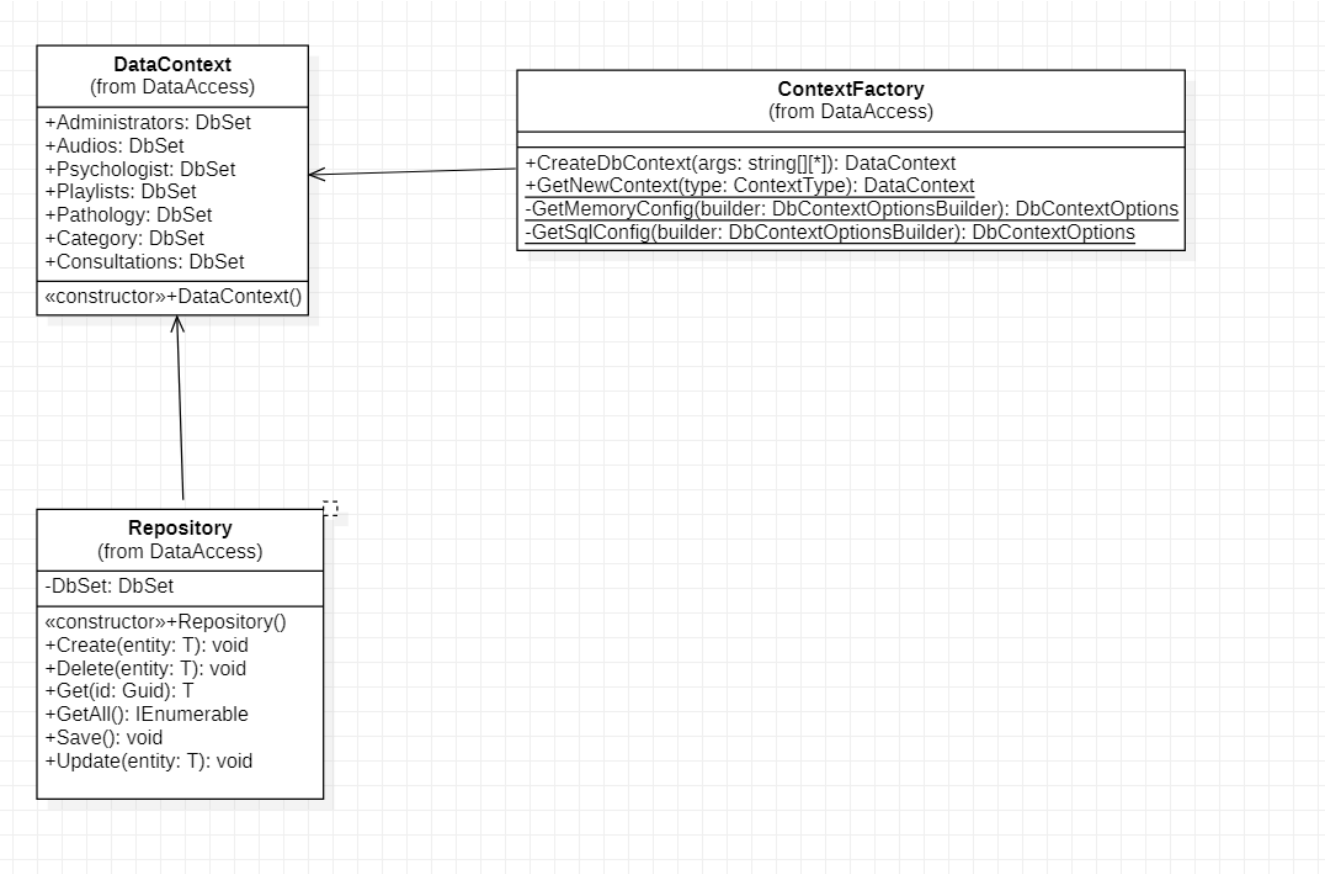
Este paquete es el encargado de manejar toda la lógica de negocio del sistema.
En ella se encuentra una clase por cada clase del dominio, lo que quiere decir que cada una representa las reglas de negocio que cada una debe llevar a cabo.

Diagrama de MSP.BetterCalm.Domain



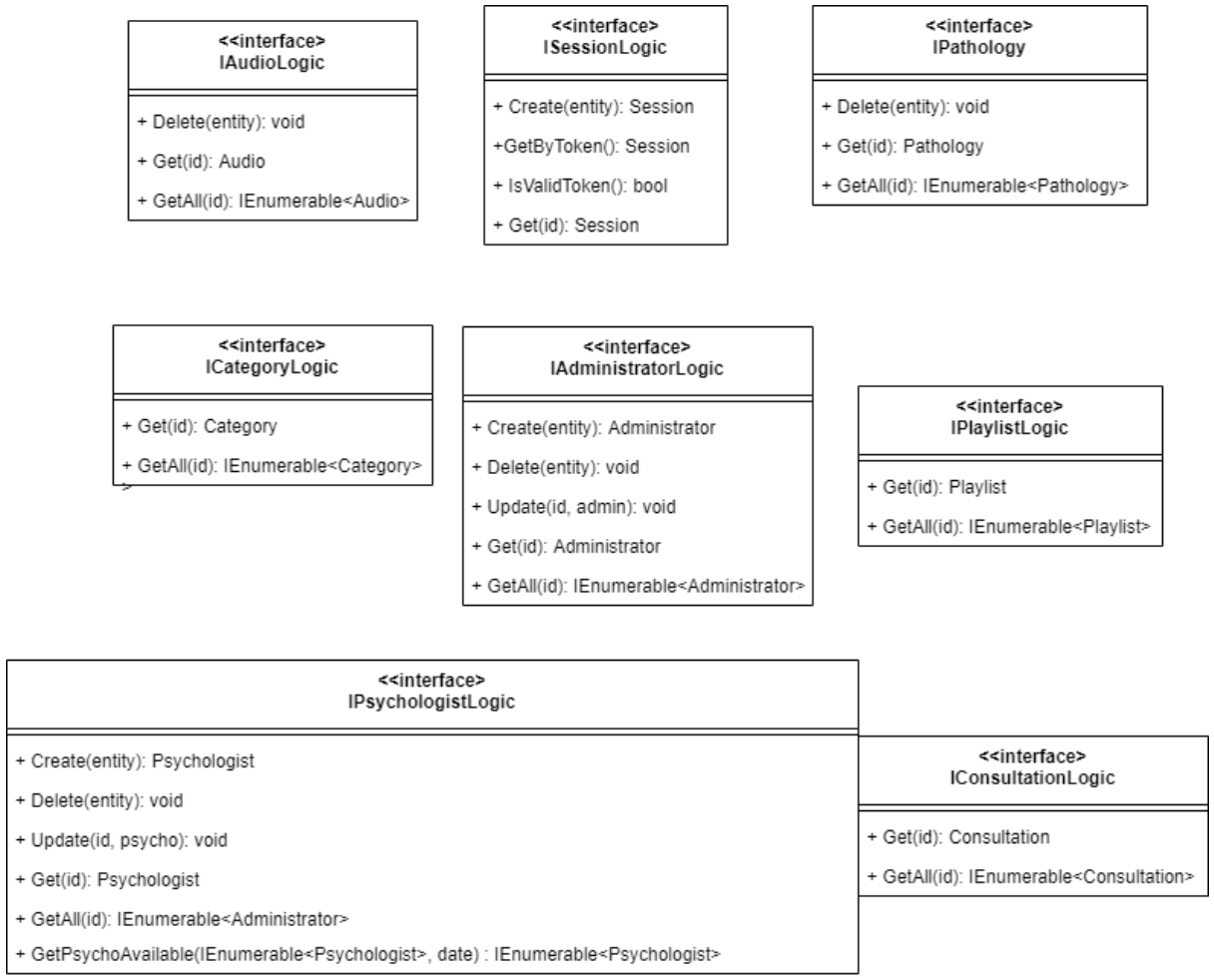
Como se puede ver en la imagen, Consultation es una composición de Psychologist porque si no hay psicólogos no van a ver consultas, por eso es que tiene esa relación de “vida”. Por otro lado, como se puede observar la clase sesión está relacionada con la clase administrador ya que es la encargada de guardar el ID del administrador con su correspondiente token de ingreso con su correspondiente fecha de ingreso.

Diagrama de MSP.Better Calm.DataAccess



La clase DataContext es la que hereda de DbContext, en la que se encuentran todas las listas o “tablas” que maneja nuestro Data Access. Para llevar a cabo el CleanCode y no repetir codigo hicimos una clase llamada Repository de tipo T la cual maneja las diferentes entidades del sistema. Por otro lado la clase ContextFactory es la cual se encarga de crear el contexto y la encargada de crear la base de datos.
Para cargar los datos utilizamos Lazy Loading.

Diagrama MSP.BetterCalm.BusinessLogic.Interface



En esta clase se llevan a cabo los métodos que son utilizados para conectar la lógica con el data access.

Diagrama MSP.BetterCalm.DataAccess.Interface

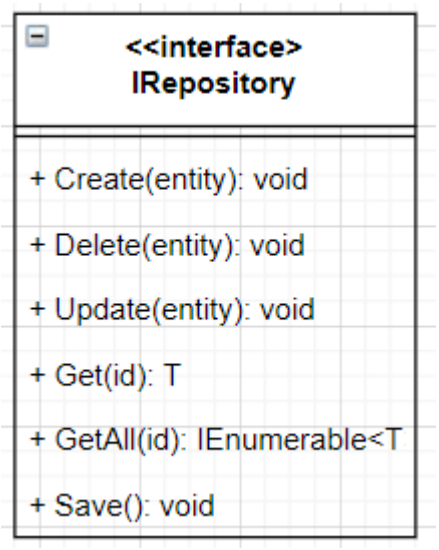
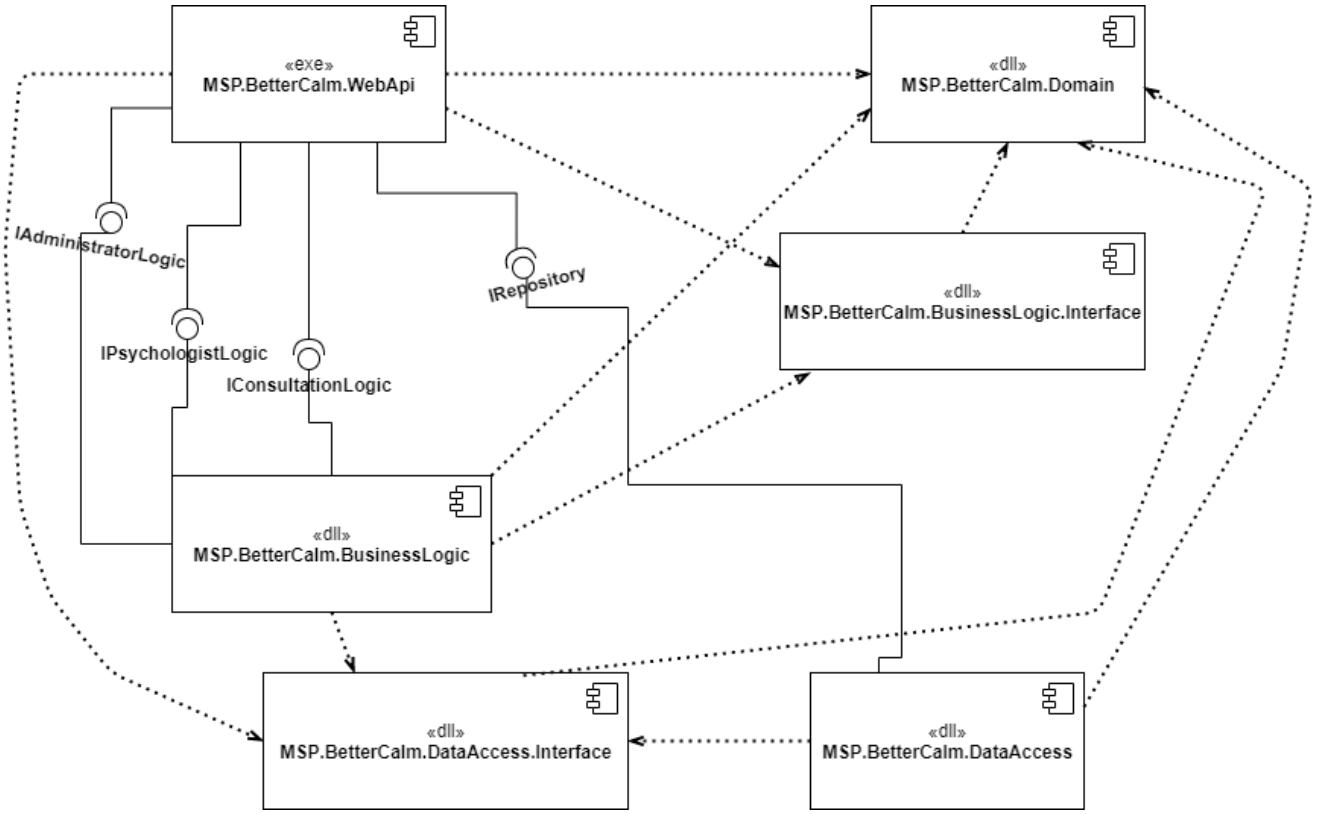


Diagrama de componentes



Para llevar a cabo dicho diagrama primero definimos los componentes que interactúan en el mismo. Los paquetes que se muestran en el diagrama de componente son los paquetes que interactúan en tiempo de ejecución, al querer en este momento consumir la Web Api para poder llevar a cabo las funcionalidades realizadas.

Organización de tareas en el obligatorio

Para llevar a cabo el obligatorio al principio comenzamos trabajando juntos, y luego de unas semanas nos dividimos el trabajo para que cada uno trabaje en un sector diferente para evitar conflictos a la hora de subir el trabajo al repositorio.