UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Aplicación móvil para gestionar las retransmisiones en directo en la plataforma Twitch

Trabajo Fin de Grado

Grado en Tecnologías Interactivas

AUTOR/A: Pardo Casanova, Rubén

Tutor/a: Palanca Cámara, Javier

Cotutor/a: Heras Barberá, Stella María

CURSO ACADÉMICO: 2022/2023

Resumen

El auge del entretenimiento en directo ha permitido a creadores de contenidos y artistas compartir sus trabajos en tiempo real con audiencias de todo el mundo a través de plataformas como Twitch. Sin embargo, moderar el chat y controlar la transmisión puede ser un problema para aquellos streamers que no cuentan con los recursos necesarios para tener varias pantallas o dispositivos. Los usuarios de Twitch a menudo tienen que lidiar con varios aspectos a la vez durante sus transmisiones, como moderar el chat, controlar el software de transmisión e interactuar con los espectadores. Esto puede resultar complicado si no cuentan con los recursos necesarios para tener varias pantallas o dispositivos. Una aplicación que facilite estas tareas podría ser de gran ayuda para estos streamers, especialmente para aquellos que están empezando y no tienen muchos recursos. La aplicación presentada en este proyecto permite a los streamers acceder a las funcionalidades necesarias para moderar el chat y controlar la aplicación OBS (Open Broadcaster Software) desde un único dispositivo móvil, gestionando las transmisiones de manera sencilla y cómoda, sin tener que depender de varios equipos.

Resum

L'auge de l'entreteniment en directe ha permés a creadors de continguts i artistes compartir els seus treballs en temps real amb audiències de tot el món a través de plataformes com Twitch. No obstant això, moderar el xat i controlar la transmissió pot ser un problema per a aquells streamers que no compten amb els recursos necessaris per a tindre diverses pantalles o dispositius. Els usuaris de Twitch sovint han de bregar amb diversos aspectes alhora durant les seues transmissions, com moderar el xat, controlar el programari de transmissió i interactuar amb els espectadors. Això pot resultar complicat si no compten amb els recursos necessaris per a tindre diverses pantalles o dispositius. Una aplicació que facilite aquestes tasques podria ser de gran ajuda per a aquests streamers, especialment per a aquells que estan començant i no tenen molts recursos. L'aplicació presentada en aquest projecte permet als streamers accedir a les funcionalitats necessàries per a moderar el xat i controlar l'aplicació OBS (Open Broadcaster Software) des d'un únic dispositiu mòbil, gestionant les transmissions de manera senzilla i còmoda, sense haver de dependre de diversos equips.

Abstract

The rise of live entertainment has allowed content creators and artists to share their work in real time with audiences around the world through platforms such as Twitch. However, moderating the chat and controlling the stream can be a problem for those streamers who don't have the resources to have multiple screens or devices. Twitch users often have to deal with several aspects at once during their broadcasts, such as moderating the chat, controlling the streaming software and interacting with viewers. This can be complicated if they don't have the resources to have multiple screens or devices. An application that facilitates these tasks could be of great help to these streamers, especially for those who are just starting out and do not have many resources. The application presented in this project allows streamers to access the necessary functionalities to moderate the chat and control the OBS (Open Broadcaster Software) application from a single mobile device, managing the transmissions in a simple and convenient way, without having to rely on several computers.

Índice

Índice **2**
Capítulo 1. Introducción **5**
1.1 Presentación y objetivos 5
1.2 Metodología de Trabajo 6
1.3 Estructura de la memoria 6
Capítulo 2. Entorno y estado del arte **8**
Capítulo 3. Análisis de requerimientos **11**
Capítulo 4. Diseño de la aplicación **17**
4.1 Inició de sesión 18
4.2 Moderación del chat 24
4.3 Control del OBS 30
4.4 Ajustes 33
Capítulo 5. Implementación **36**
5.1 Inicio de sesión 36
5.2 Moderación del chat 37
5.3 Control del OBS 40
5.4 Ajustes 41
Capítulo 6. Manuales **43**
6.1 Compilación de la aplicación en dispositivos Android 43
6.2 Compilación de la aplicación en dispositivos iPhone 43
6.3 Configuración de la Consola de Twitch para la API 43
Capítulo 7. Evaluación **44**
7.1 Pruebas de usabilidad 44
7.2 Pruebas unitarias 47
Capítulo 8. Conclusiones **49**
8.1 Trabajo a futuro 50
8.2 Relación del trabajo desarrollado con las asignaturas con el grado 50

Índice de tablas

Tabla 1. Comparación de las soluciones actuales con la nuestra.
Tabla 3.1 Caso de uso: Inicio de sesión.
Tabla 3.2 Caso de uso: Menú inferior
Tabla 3.3 Caso de uso: Leer el chat de Twitch.
Tabla 3.4 Caso de uso: Leer el chat de Twitch.
Tabla 3.5 Caso de uso: Borrar un mensaje del chat de Twitch.
Tabla 3.6 Caso de uso: Vetar un usuario del chat de Twitch.
Tabla 3.8 Caso de uso: Filtrar mensajes de Twitch, mostrar sólo suscripciones.
Tabla 3.8 Caso de uso: Filtrar mensajes de Twitch, mostrar sólo los de un usuario
Tabla 3.9 Caso de uso: Vincular OBS
Tabla 3.10 Caso de uso: Cambiar escena del OBS
Tabla 3.11 Caso de uso: Modificar pista de audio de OBS
Tabla 3.12 Caso de uso: Cambiar el título de la retransmisión y su categoría
Tabla 3.13 Caso de uso: Sincronización con Twitch
Tabla 3.14 Caso de uso: Sincronización con Twitch
Tabla 3.15 Caso de uso: Sincronización con OBS

Índice de figuras

Figura 1. Esquema de la arquitectura Clean con el patrón BLoC
Figura 2. Wireframe del inicio de sesión
Figura 3. Diseño lógico de la presentación del inicio de sesión
Figura 4. Diseño lógico del domain del inicio de sesión
Figura 5. Diseño lógico de la parte de datos del inicio de sesión
Figura 6. Diagrama de flujo del inicio de sesión
Figura 7. Wireframe de la moderación del chat
Figura 8. Diseño lógico de la parte de presentación del chat
Figura 9. Diseño lógico de la parte de dominio del chat
Figura 10. Diseño del modelo del chat
Figura 11. Diseño de clases de los IRCMessages del chat
Figura 12. Diseño lógico de la parte de data del chat
Figura 13. Wireframe del control del OBS
Figura 14. Diseño de la capa de presentación del control del OBS
Figura 15. Diseño de la capa de dominio del control del OBS
Figura 16. Diseño de la capa de data del control del OBS
Figura 17. Wireframe de los ajustes
Figura 18. Diseño lógico de la presentación de los ajustes
Figura 19. Diseño lógico del dominio de los ajustes
Figura 20. Diseño lógico de la capa de data de los ajustes
Figura 22. Diseño de la pantalla de inicio de sesión de la aplicación
Figura 23. Diseño de las pantalla de la integración del chat

Figura 24. Diseño de las pantalla de la integración del OBS
Figura 25. Diseño de las pantalla de los ajustes
Figura 26. Gráfica de los resultados de “¿Pudiste iniciar sesión con facilidad?
Figura 27. Gráfica de los resultados sobre la opinión del diseño del inicio de sesión
Figura 28. Gráfica de los resultados sobre la facilidad de vincular el OBS
Figura 29. Gráfica de los resultados sobre la usabilidad del control del OBS
Figura 30. Gráfica de los resultados sobre la opinión de la usabilidad del chat de Twitch
Figura 31. Gráfica de los resultados sobre la opinión de la usabilidad de los ajustes
Figura 32. Código ejemplo test unitario Inicio sesión.

Capítulo 1. Introducción

1.1 Presentación y objetivos

La transmisión en línea se ha vuelto cada vez más popular en internet debido al aumento del uso de dispositivos móviles y la disponibilidad de conexiones a internet de alta velocidad. Esta forma de acceso permite a los usuarios ver y compartir contenido en tiempo real, lo que ha dado lugar a una gran variedad de plataformas para la transmisión en línea.

Una de las plataformas más populares para la transmisión en línea es Twitch (Twitch Interactive, 2023), que se ha convertido en uno de los líderes en el área de los juegos en línea y todo tipo de contenido relacionado con los juegos. Twitch ofrece una amplia gama de contenido, incluyendo juegos en vivo, tutoriales, competiciones y contenido creado por los usuarios.

Sin embargo, la transmisión en línea también presenta desafíos importantes para los usuarios que transmiten, conocidos como "streamers". La moderación del chat y el control del software de transmisión son dos de los mayores desafíos a los que se enfrentan los streamers. La moderación del chat puede ser abrumadora debido a la cantidad de mensajes que se llegan a recibir, haciendo muy compleja la tarea de identificar y eliminar mensajes inapropiados. Además, la moderación del chat manual puede ser tediosa y requerir mucho tiempo y esfuerzo.

El control del software de transmisión también puede ser un desafío, ya que los streamers pueden tener dificultades para ajustar la transmisión en tiempo real y pueden requerir varias pantallas para hacerlo. Esto puede ser costoso y requerir mucho tiempo y esfuerzo para configurar y usar la aplicación. Además, la falta de varias pantallas físicas puede dificultar el control del software de transmisión, lo que puede resultar en errores y problemas técnicos durante la transmisión.

El **objetivo general** de este trabajo es desarrollar una aplicación móvil que permita a un usuario moderar el chat de su cuenta de Twitch y controlar remotamente el software de la transmisión. Los objetivos específicos para conseguir este objetivo general son:

- Objetivo 1: Analizar el entorno y el estado del arte para estos problemas identificados con el objetivo de analizar otras posibles soluciones que se están aplicando actualmente.

- Objetivo 2: Acceder a datos en tiempo real, como el chat y sus funcionalidades principales, además de poder moderar los mensajes.

- Objetivo 3: Controlar remotamente el OBS (Open Broadcaster Software, 2023) para permitir a los usuarios controlar el software de transmisión desde su dispositivo móvil.

- Objetivo 4: Desarrollar una aplicación que sea compatible con los sistemas Android e iOS.

- Objetivo 5: Realizar pruebas exhaustivas de la aplicación y asegurar su estabilidad y seguridad.

- Objetivo 6: Documentar todo el proceso de desarrollo y proporcionar una guía detallada para la implementación y el uso de la aplicación.

- Objetivo 7: Analizar el impacto de la aplicación en la experiencia de transmisión.

1.2 Metodología de Trabajo

En este trabajo se ha seguido una metodología SCRUM, la cual se basa en la gestión ágil de proyectos.

SCRUM es una metodología ágil de gestión de proyectos centrada en la colaboración, la flexibilidad y la implementación iterativa. Se basa en la división del trabajo en ciclos cortos llamados sprints en los que se planifican, ejecutan y confirman las tareas. (*Scrum: Conceptos Clave Y Cómo Se Aplica En La Gestión De Proyectos [2023] • Asana*, 2023)

Si bien SCRUM se enfoca en la creación de proyectos en equipos, los principios de SCRUM se pueden usar para dividir los proyectos en partes más cortas y manejables y establecer objetivos claros. Al usar esta adaptación de SCRUM, los proyectos individuales pueden beneficiarse del sistema de planificación, aumentar la productividad y poder cambiar los procesos si es necesario para lograr los objetivos mencionados.

Para llevar a cabo esta metodología, se han establecido sprints de duración aproximada de 2-3 semanas. Estos sprints son periodos de tiempo en los cuales se planifican, desarrollan y entregan incrementos de producto.

Al inicio de cada sprint, se llevan a cabo revisiones con los product owners, que en nuestro caso han sido los tutores del proyecto. Durante estas revisiones, se muestran los avances realizados y se recopilan los comentarios y sugerencias de los product owners para guiar el desarrollo futuro. Posteriormente se revisa el backlog, que es una lista priorizada de historias de usuario pendientes. En base a esta revisión, se seleccionan las historias de usuario las cuales se desglosan en las tareas a realizar en el sprint se priorizan y se le estima un tiempo para realizarla.

1.3 Estructura de la memoria

La memoria de este trabajo está organizada en distintos capítulos, que describen el entorno y estado del arte, el análisis de requisitos, el diseño de la aplicación y su implementación, los manuales de instalación y uso, la evaluación de la aplicación y sus conclusiones

- En el **capítulo 2**, se amplía la descripción del entorno en el que se desarrollará la aplicación, identificando los problemas existentes y analizando soluciones previas.

- El **capítulo 3** define las características, los requisitos y casos de uso que se necesitan para el desarrollo de la aplicación.

- El **capítulo 4** se enfoca en el diseño de la aplicación, incluyendo la arquitectura general, la lógica del negocio y los algoritmos más importantes.

- En el **capítulo 5**, se describe la metodología de trabajo y herramientas de desarrollo utilizadas, la estructura de la aplicación y problemas de implementación resueltos.

- En el **capítulo 6** se incluyen los manuales de instalación y configuraciones del proyecto además de las guías de uso.

- El **capítulo 7** describe cómo se ha evaluado el sistema desde el punto de vista de la usabilidad. Se describen las técnicas y métodos utilizados para evaluar la facilidad de uso y la satisfacción del usuario.

- En el **Capítulo 8** se resumen los hitos y méritos logrados durante el desarrollo de la aplicación. Además, se detallan las áreas en las que se pueden realizar mejoras y se establecen objetivos concretos para el trabajo futuro.

Capítulo 2. Entorno y estado del arte

Twitch es una plataforma de entretenimiento en línea especializada en transmisiones en vivo de juegos, música, arte y programación. Es una de las plataformas más populares en este campo, con millones de usuarios activos y cientos de miles de transmisiones en vivo cada día (*Twitch Statistics 2023 - How Many People Use Twitch?*, 2023). Los streamers, o personas que transmiten en vivo en Twitch, pueden interactuar con su audiencia a través de un chat en tiempo real, y los espectadores pueden hacer donaciones o comprar membresías para apoyar a sus streamers favoritos.

Twitch es una gran comunidad de jugadores, desarrolladores, músicos y artistas, y la plataforma ofrece una gran variedad de contenido para todos los gustos. Los streamers pueden transmitir desde cualquier lugar y en cualquier momento, y pueden elegir entre una gran variedad de juegos y categorías, como videojuegos, música, arte, programación, entre otros.

Sin embargo, transmitir en vivo conlleva un cierto grado de responsabilidad, especialmente en términos de moderación del chat y control del software de transmisión. Los streamers deben moderar el chat para mantener un ambiente seguro y acogedor para su audiencia. Además de la moderación del chat, los streamers también tienen la responsabilidad adicional de actuar como realizadores en vivo. Esto puede implicar cambiar entre diferentes vistas de cámara o escenas, ajustar la configuración de audio y video en tiempo real, y agregar elementos visuales como superposiciones y gráficos. Actualmente esto se realiza mediante programas como OBS (Open Broadcaster Software, 2023). Ser un buen realizador en vivo puede mejorar significativamente la calidad de la transmisión y la experiencia del espectador, y puede requerir un conjunto adicional de habilidades y recursos.

Esto puede ser un desafío, especialmente para aquellos streamers que no tienen los recursos para tener varias pantallas o dispositivos para moderar el chat y controlar el OBS al mismo tiempo.

Los streamers suelen recurrir a diferentes herramientas y soluciones para poder moderar el chat y controlar el software de transmisión de sus directos en Twitch. Algunos de los métodos más comunes incluyen el **uso de bots de Twitch**, como Nightbot (NightDev, LLC., 2023) o Moobot (Moobot, 2023), que automatizan algunas funciones de moderación del chat, cómo silenciar palabras o frases específicas. Otros streamers utilizan herramientas de escritorio, como el **software de transmisión** OBS o Stream Labs (Logitech Services S.A, 2023), que ofrecen una interfaz gráfica para controlar la configuración de la transmisión y moderar el chat.

Los **stream decks** son dispositivos de hardware específicamente diseñados para streamers, que ofrecen una interfaz de control para varias funciones del software de transmisión y moderación del chat. Estos dispositivos suelen tener una serie de botones programables que permiten al usuario acceder rápidamente a funciones comunes, como cambiar de escena en el OBS, silenciar o banear a un usuario del chat, o iniciar o detener la transmisión. Aún así no son asequibles para todo el mundo ya que tiene un precio aproximado de 100 euros (Cervera, 2021).

Sin embargo, estas soluciones pueden ser insuficientes para los streamers nuevos, más concretamente para aquellos que no tienen los recursos para tener varias pantallas o dispositivos para moderar el chat y controlar el software de transmisión al mismo tiempo. Estos streamers pueden tener dificultades para supervisar y moderar el chat en tiempo real, lo que puede afectar negativamente a su experiencia y a la de su audiencia.

Las **aplicaciones móviles** pueden ser muy útiles para streamers con pocos recursos o nuevos en la plataforma de Twitch, ya que les permiten moderar el chat y controlar el software de transmisión desde un único dispositivo móvil. Estas aplicaciones ofrecen una interfaz intuitiva y fácil de usar.

Algunas de las aplicaciones móviles más populares para estos streamers son:

- **Obs Blade**: Es una aplicación móvil que permite a los streamers controlar su software de transmisión OBS desde su dispositivo móvil. Ofrece una interfaz fácil de usar y una gran variedad de funciones, como la posibilidad de cambiar de escena, ajustar el volumen o controlar las cámaras. Sin embargo, no ofrece la posibilidad de interactuar directamente con el chat (Kounex, 2023).

- **La aplicación de Twitch**: La aplicación oficial de Twitch permite a los streamers ver los comentarios en tiempo real y responder a ellos desde su dispositivo móvil, además de ver las estadísticas de la transmisión y recibir notificaciones de nuevos seguidores o donaciones. Sin embargo, no ofrece funciones avanzadas de control del OBS o automatización de tareas (Twitch Interactive, Inc., 2023).

- **Streamer Companion**: Es una aplicación móvil gratuita pero abandonada que permite a los streamers ver el chat en tiempo real y recibir notificaciones de nuevos seguidores o donaciones. Sin embargo, solo se puede conectar a Streamlabs y no tiene la posibilidad de interactuar directamente con el chat o controlar el OBS (Jourdan, 2022).

- **Touch Portal**: Es una aplicación móvil gratuita que permite a los streamers controlar su software de transmisión OBS y moderar el chat desde su dispositivo móvil. Ofrece una interfaz personalizable y una gran variedad de funciones, pero puede ser un poco más compleja de usar que otras aplicaciones. Es muy similar a una stream deck pero virtual, por tanto no puede moderar el chat de Twitch (*Touch Portal*, 2023).

A modo de resumen se muestra la Tabla 1 que compara las soluciones existentes con nuestra aplicación. La primera hace referencia a un usuario que tiene los recursos suficientes para tener varias pantallas y no hace uso de las otras aplicaciones. Por otra parte, la segunda fila se refiere al hacer uso de bots que moderan el chat automáticamente. La última fila (StreaMate) hace referencia a la propuesta de este trabajo.

Solución	Coste	Moderación del chat	Configuración de la transmisión	Dificultad de uso	Plataformas
Usuario con varias pantallas y recursos suficientes	Muy alto	Muy Alta	Muy Alta	Medio	-
Bots de Twitch	Gratis o Bajo	Alta	No lo hace	Medio	Navegador
OBS Blade	Gratis	No lo hace	Alta	Baja - Media	iOS / Android
Aplicación de Twitch	Gratis	Muy Alta	No lo hace	Media	iOS/Android
Stream Companion	Gratis	Baja	No lo hace	Baja	Android
TouchPortal	Gratis o Medio	No lo hace	Media	Media - Alta	iOS/Android
StreaMate	Gratis	Alta	Media - Alta	Baja	iOS/Android

Tabla 1. Comparación de las soluciones actuales con la nuestra.

En general, todas las soluciones tienen sus ventajas y desventajas. Sin embargo, StreaMate se destaca por ser una solución gratuita con moderación de chat y control de las diferentes escenas con una dificultad de uso baja, lo que la hace ideal para usuarios nuevos. Además, es compatible con iOS y Android, lo que la hace accesible para usuarios de dispositivos móviles.

Capítulo 3. Análisis de requerimientos

Nuestro objetivo es desarrollar una aplicación multiplataforma en la que sea posible tener el chat de Twitch integrado y se pueda interactuar con él y moderar, además de gestionar la transmisión.
Se ha llevado a cabo un estudio de caso en el que se ha entrevistado a varios streamers que no cuentan con los recursos necesarios para tener varias pantallas o dispositivos y se ha recogido información sobre sus necesidades y demandas en cuanto a herramientas de moderación y control de transmisión, así como sus experiencias con las soluciones existentes. Con esta información se han extraído un conjunto de requisitos para desarrollar una aplicación que aborde estos problemas. Estos requisitos son los siguientes:

- Desarrollo de una aplicación **multiplataforma**: La aplicación debe ser compatible con dispositivos iOS y Android, lo que implica el uso de herramientas de desarrollo como React Native (Meta Platforms, Inc., 2023), Xamarin (*Xamarin.Forms Documentación - Xamarin*, 2023) o Flutter (*Flutter*, 2023). Estas herramientas permiten crear aplicaciones nativas con un único código base, lo que reduce la complejidad y el tiempo de desarrollo. En nuestro caso se va a realizar el desarrollo con Flutter.

- **Integración con Twitch** para iniciar sesión y la moderación del chat: La aplicación debe poder conectarse a la API de Twitch para acceder a datos como el chat y sus funcionalidades y permitir a los usuarios moderar el chat de Twitch en tiempo real, lo que incluye la capacidad de eliminar mensajes y vetar usuarios, entre otras.

- **Control de OBS**: La aplicación debe ser capaz de controlar el OBS a través de una conexión remota. Esto se puede lograr mediante el uso de protocolos como el protocolo de control de OBS o con websockets.

Dado este planteamiento se han definido los siguientes casos de uso. Estos casos de uso representan la lógica de negocio o funcionalidades de la aplicación:

Caso de uso	1. Inicio de sesión
Descripción	Permite acceder a la aplicación y hacer uso de las funcionalidades relacionadas con Twitch
Precondición	- No estar autenticado anteriormente - No tener un token de acceso guardado en memoria de la app
Flujo básico	- Pulsar el botón de “Iniciar Sesión” - Introducir correo y contraseña de una cuenta de Twitch - Autorizar la aplicación
Postcondición	- Se accede a la vista “Chat” - Se guarda el token de acceso para posteriores inicio de sesión

Tabla 3.1 Caso de uso: Inicio de sesión.

Caso de uso	2. Menú inferior
Descripción	Se muestran 3 botones inferiores que permiten navegar a las demás funcionalidades
Precondición	- Iniciar sesión
Flujo básico	- Se pulsa en algún botón del menú inferior
Postcondición	- Muestra la vista correspondiente a la seleccionada

Tabla 3.2 Caso de uso: Menú inferior

Caso de uso	3. Leer el chat de Twitch
Descripción	Permite visualizar los mensajes del chat de Twitch de su directo.
Precondición	- Iniciar sesión
Flujo básico	- Seleccionar la opción “Chat” en el menú inferior
Postcondición	- Se mostrarán los mensajes de Twitch de su retransmisión

Tabla 3.3 Caso de uso: Leer el chat de Twitch.

Caso de uso	4. Cambiar los ajustes del chat de Twitch
Descripción	Poder cambiar el modo lento, suscripciones, seguidores y emoticonos del chat de Twitch
Precondición	- Iniciar sesión - Acceder a la ventana del chat
Flujo básico	- Seleccionar una opción de los ajustes del chat para cambiar su valor
Postcondición	- Se mostrará un mensaje de que el modo del chat ha cambiado a lo indicado en el mismo chat de Twitch

Tabla 3.4 Caso de uso: Leer el chat de Twitch.

Caso de uso	5. Borrar un mensaje del chat de Twitch.
Descripción	Eliminar un mensaje del chat de twitch
Precondición	- Iniciar sesión - Acceder a la ventana del chat
Flujo básico	- Seleccionar un mensaje - Darle al botón de borrar
Postcondición	- El mensaje ya no aparece en el chat - Se mostrará un mensaje de que el mensaje ha sido borrado en el mismo chat de twitch

Tabla 3.5 Caso de uso: Borrar un mensaje del chat de Twitch.

Caso de uso	6. Vetar a un usuario del chat de Twitch.
Descripción	Veta del chat de Twitch a un usuario
Precondición	- Iniciar sesión - Acceder a la ventana del chat
Flujo básico	- Pulsar el nombre de usuario de un mensaje del chat - Darle a vetar permanente o expulsión temporal
Postcondición	- Se borrarán todos los mensajes de ese usuario

Tabla 3.6 Caso de uso: Vetar un usuario del chat de Twitch.

Caso de uso	7. Filtrar tipo de mensajes de Twitch
Descripción	Se mostrará solo en el chat las suscripciones del canal
Precondición	- Iniciar sesión - Acceder a la ventana del chat
Flujo básico	- Pulsar el botón de filtrar del appbar de la aplicación
Postcondición	- Si el filtro estaba inactivo se mostrarán ahora las suscripciones que hay y las

	nuevas por llegar y pasará a estar activo. - Si el filtro estaba activo se mostrarán todos los mensajes y pasará a estar inactivo.

Tabla 3.8 Caso de uso: Filtrar mensajes de Twitch, mostrar sólo suscripciones.

Caso de uso	8. Filtrar mensajes de Twitch, mostrar sólo los de un usuario
Descripción	Se mostrará los mensajes de un solo usuario
Precondición	- Iniciar sesión - Acceder a la ventana del chat
Flujo básico	- Pulsar el nombre de usuario de un mensaje del chat
Postcondición	- Se mostrará una ventana con su información y un nuevo chat con solo sus mensajes

Tabla 3.8 Caso de uso: Filtrar mensajes de Twitch, mostrar sólo los de un usuario

Caso de uso	9. Vincular OBS
Descripción	Poder acceder a la funcionalidades del OBS
Precondición	- Iniciar sesión - Acceder a la ventana de OBS
Flujo básico	- Pulsar el botón de “Vincular OBS” - Introducir ip, puerto y contraseña del OBS
Postcondición	- Se mostrarán todas las escenas y pistas de audio configuradas en el OBS

Tabla 3.9 Caso de uso: Vincular OBS

Caso de uso	10. Cambiar escena de OBS
Descripción	Cambiar la escena actual de la retrasmisión entre todas las configuradas en el OBS
Precondición	- Iniciar sesión

	- Acceder a la ventana de OBS - Tener vinculado el OBS
Flujo básico	- Seleccionar una de las escenas de la lista
Postcondición	- Se indicará con otro color la seleccionada - Se envía un mensaje al OBS para indicar que se debe cambiar de escena

Tabla 3.10 Caso de uso: Cambiar escena del OBS

Caso de uso	11. Modificar pista de audio de OBS
Descripción	Cambiar la escena actual de la retrasmisión entre todas las configuradas en el OBS
Precondición	- Iniciar sesión - Acceder a la ventana de OBS - Tener vinculado el OBS
Flujo básico	- Modificar el valor de uno de los “sliders” de las pistas de audio o darle al botón de silenciar la pista
Postcondición	- Se modificará el valor de la pista de audio en el OBS

Tabla 3.11 Caso de uso: Modificar pista de audio de OBS

Caso de uso	12. Cambiar el título de la retransmisión y su categoría
Descripción	Cambiar el título de la retransmisión y seleccionar una categoría con un buscador
Precondición	- Iniciar sesión - Acceder a la sección de configuración
Flujo básico	- Modificar el título o la categoría - Pulsar el botón de guardar que solo aparecerá cuando detecte un cambio en uno de estos dos elementos
Postcondición	- Se refleja el cambio en la web

Tabla 3.12 Caso de uso: Cambiar el título de la retransmisión y su categoría

Caso de uso	13. Cerrar Sesión
Descripción	Cerrar sesión
Precondición	- Iniciar sesión - Acceder a la sección de configuración
Flujo básico	- Pulsar el botón de cerrar sesión
Postcondición	- La app redirige a la ventana de Inicio de sesión

Tabla 3.13 Caso de uso: Sincronización con Twitch

Caso de uso	14. Sincronización con Twitch
Descripción	La aplicación escuchará los cambios del chat para modificar la aplicación
Precondición	- Iniciar sesión
Flujo básico	- Se modifica algún ajuste del chat de Twitch que se refleja en la aplicación en cualquier parte fuera de ella
Postcondición	- Se refleja el cambio en la app

Tabla 3.14 Caso de uso: Sincronización con Twitch

Caso de uso	15. Sincronización con OBS
Descripción	La aplicación escuchará los cambios del OBS para modificar la aplicación
Precondición	- Iniciar sesión - Vincular el OBS
Flujo básico	- Se modifica algún ajuste del OBS que se refleja en la aplicación en cualquier parte fuera de ella
Postcondición	- Se refleja el cambio en la app

Tabla 3.15 Caso de uso: Sincronización con OBS

Capítulo 4. Diseño de la aplicación

La arquitectura de la aplicación se basa en la arquitectura limpia (Martin, 2012), que se enfoca en separar la lógica de negocio de la lógica de la interfaz de usuario. Esto se logra mediante el uso de patrones de diseño, en nuestro caso el patrón BLoC (Business Logic Component), un enfoque de programación que separa la lógica de negocio de la interfaz de usuario, utilizando un componente intermedio para gestionar los eventos y los estados de una aplicación (Sánchez, 2020) y la inyección de dependencias (Wikipedia, 2023.) con GetIt (Flutter Community, 2021), para proporcionar los objetos o componentes necesarios a una parte del código, en lugar de crearlos directamente en esa parte, con el fin de facilitar la reutilización, la flexibilidad y el mantenimiento del código.

Usando BloC con arquitectura limpia tenemos el esquema de la Figura 1 en la aplicación:

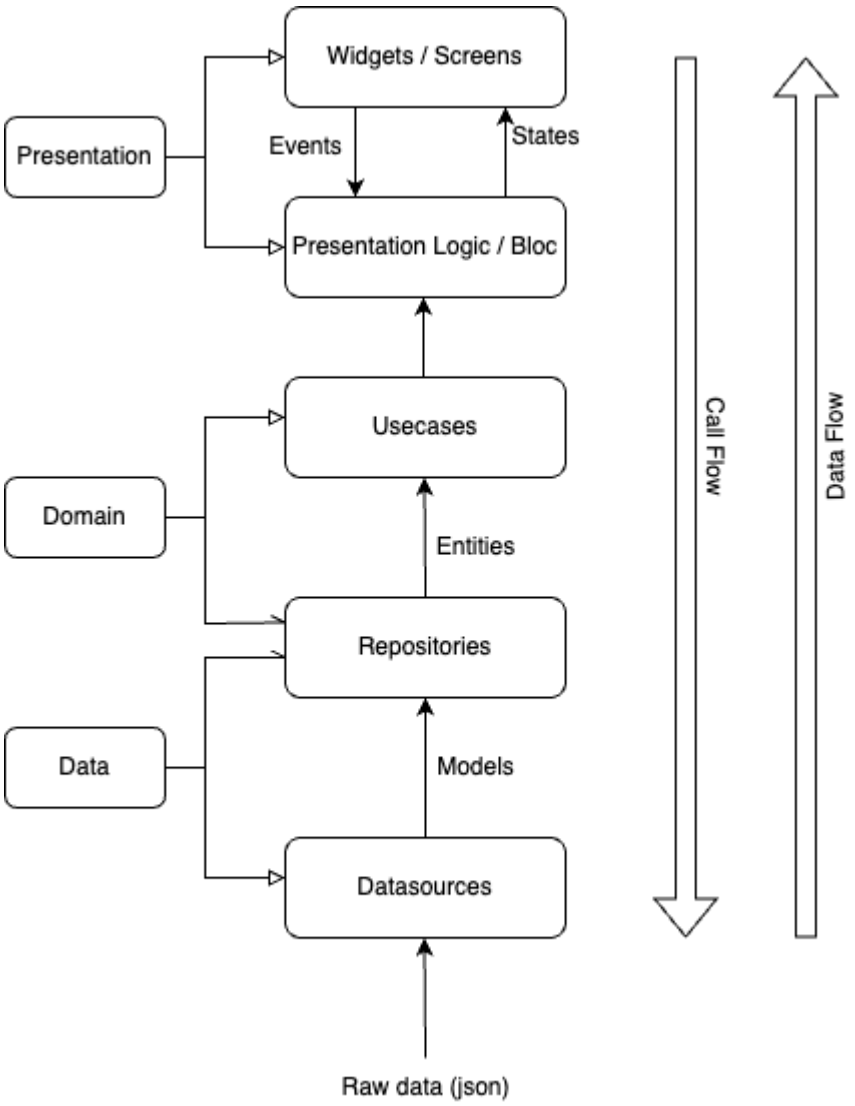

Figura 1. Esquema de la arquitectura Clean con el patrón BLoC

En la capa de presentación las pantallas se comunican con su respectivo BLoC mediante **eventos** y este devuelve **estados** para que la pantalla se represente respecto a estos estados. La comunicación se basa en el patrón Observer (Wikipedia, 2023) el cual permite a un objeto, llamado "sujeto", notificar y actualizar automáticamente a una lista de objetos "observadores" cuando se

producen cambios en su estado, de manera que los observadores puedan responder y realizar acciones en consecuencia.

Los BLoC utilizan los **casos de uso** o use cases, que a efectos prácticos es una clase que representa una acción o tarea que se debe realizar en la aplicación. Estos “use cases” contienen la lógica de negocio necesaria para ejecutar una tarea específica.

Los **repositorios** se encargan de proporcionar acceso a los datos necesarios para ejecutar los “use cases”. Estos pueden obtener los datos de diferentes fuentes, como una base de datos, una API externa o un almacenamiento local. Los repositorios son responsables de traducir los datos a un formato que sea comprensible para los casos de uso.

Los **data sources** (o “services”) son las fuentes de datos específicas de donde los repositorios obtienen los datos. Por ejemplo, una base de datos, una API externa, un archivo local, etc. Los data sources son específicos para cada plataforma o sistema y pueden ser implementados de manera diferente para cada uno.

La inyección de dependencias con GetIt ayuda a mantener la aplicación organizada y fácil de mantener. Permite que los componentes de la aplicación se comuniquen entre sí de forma clara y fácil de entender, reduciendo el acoplamiento y permitiendo una mayor escalabilidad en el futuro.

A continuación se explicará el diseño lógico y de interfaz que se ha realizado para cada sección de la aplicación:

4.1 Inició de sesión

Diseño de la interfaz

La interfaz del inicio de sesión, que se puede ver en la Figura 2, tiene dos ventanas: la ventana principal con el formulario para iniciar sesión y el webview que realiza el proceso con el servidor de twitch

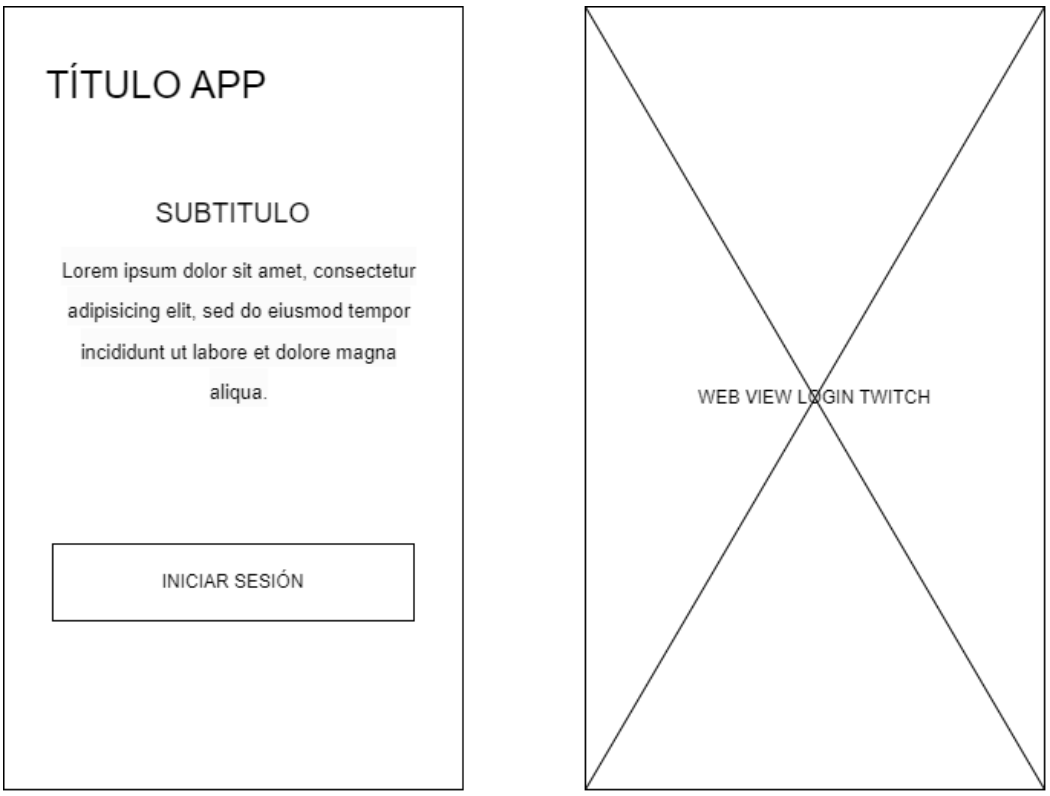

Figura 2. Wireframe del inicio de sesión

Diseño Lógico

En la Figura 3 se define el diseño lógico del inicio de sesión por el que se dispone de dos archivos, “LoginScreen” y “LoginWebViewScreen”, encargados de representar la vista. Estos archivosse comunican con “AuthBloc” a través de los eventos “AuthEvent” y este devuelve estados “AuthState”.

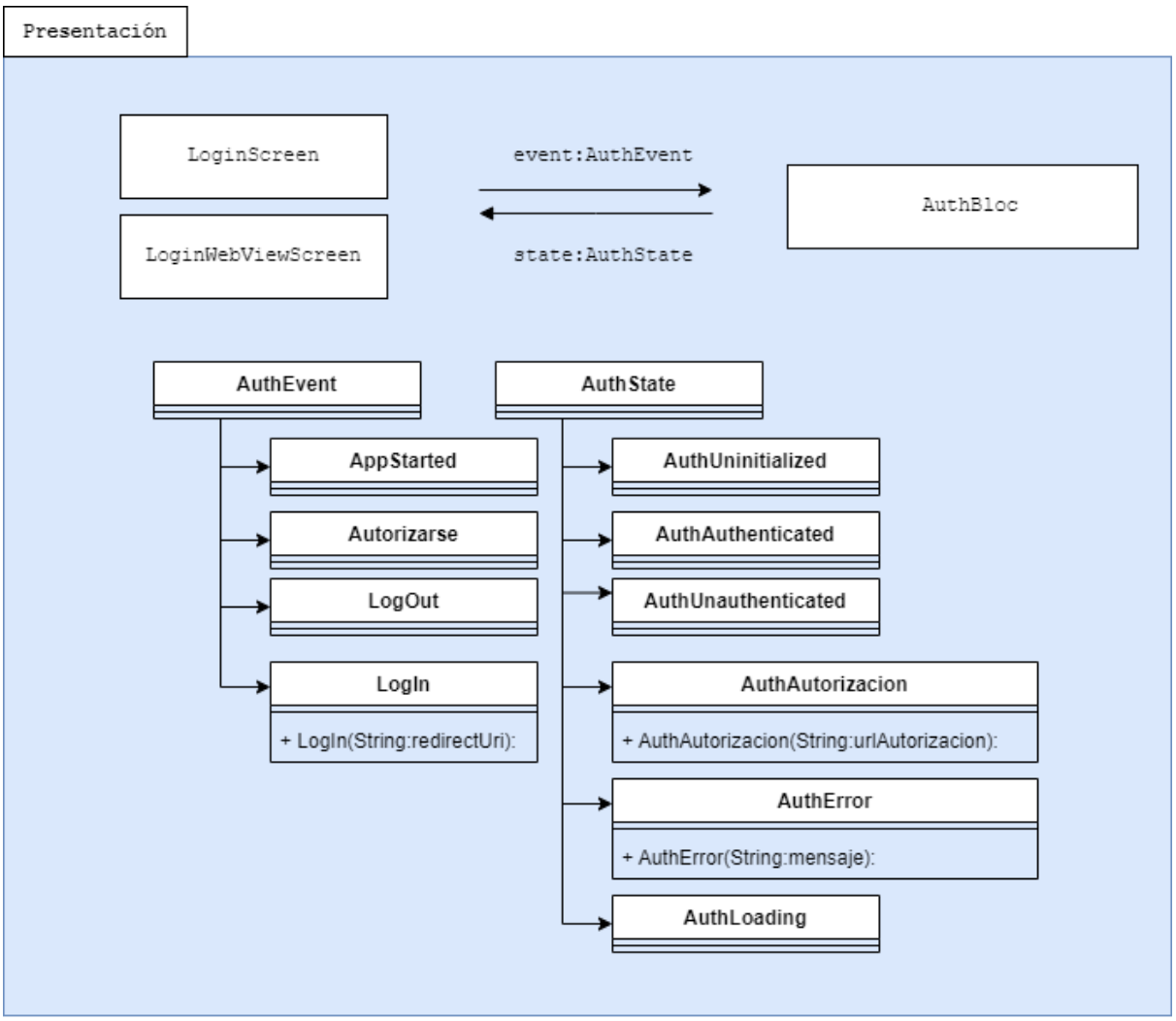

Figura 3. Diseño lógico de la presentación del inicio de sesión

La lógica de la presentación de toda la aplicación se ha realizado de la misma forma. Archivos que representan la parte visual, en la Figura 3 “LoginScreen” y “LoginWebViewScreeen”, que se comunican por eventos a un archivo BloC que gestiona el estado de estas pantallas mediante el patrón de observer, patrón de diseño de comportamiento que te permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando, en este caso AuthBloc.

Para la autenticación principalmente se dispone de los eventos login y logout y el BloC devuelve estados dependiendo de si se ha producido un error o si el usuario está autenticado o no.

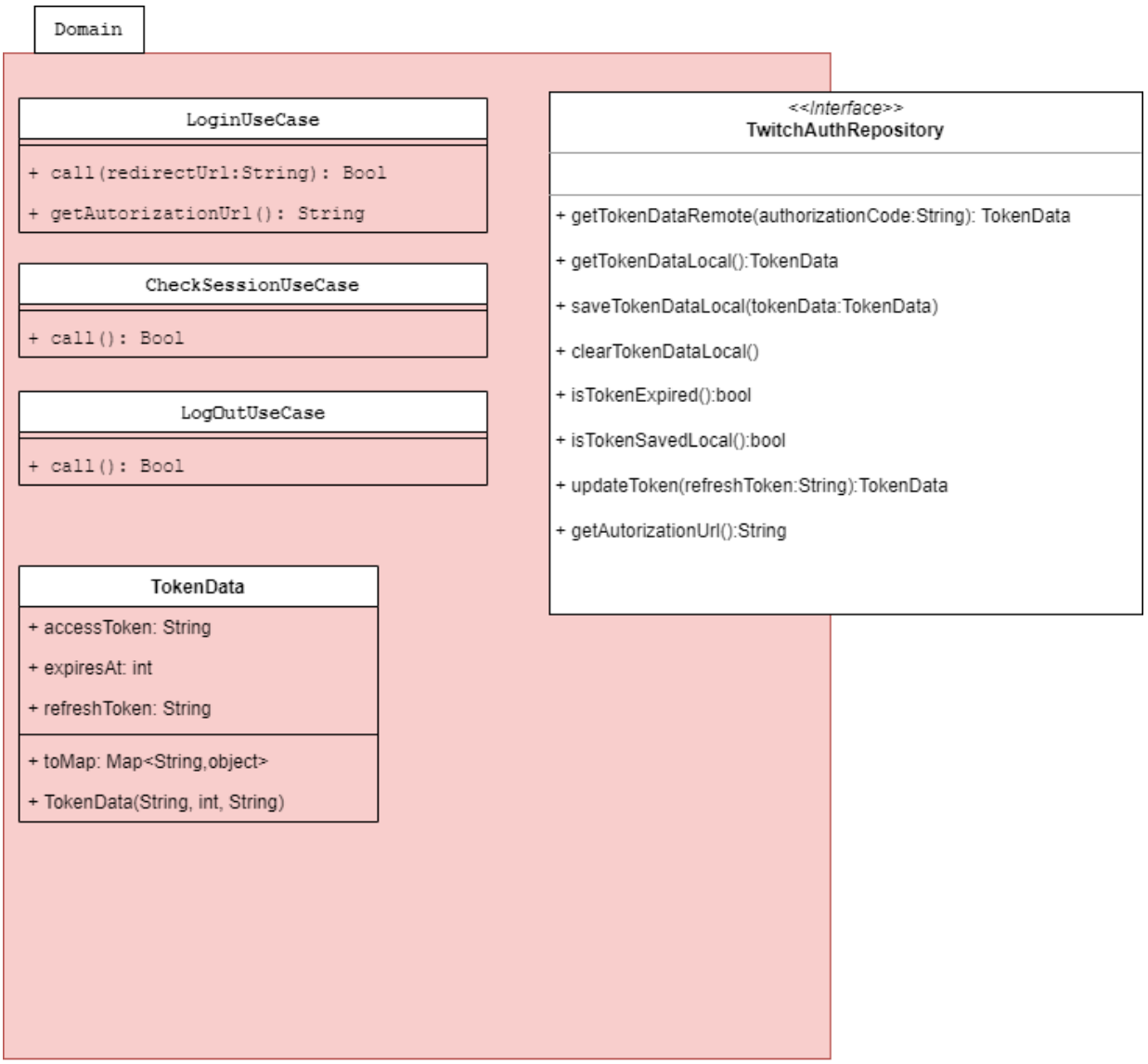

Figura 4. Diseño lógico del domain del inicio de sesión

En la sección "domain" de la aplicación, se encuentran los casos de uso que son utilizados por el BLoC para comunicarse (Figura 4). Estos casos de uso, a su vez, hacen uso de las funciones proporcionadas por los repositorios. En el registro, la sección "domain" alberga los casos de uso y los objetos necesarios para acceder a la aplicación. En este caso el objeto “TokenData”, hace referencia al token de acceso para poder iniciar sesión y realizar las posteriores peticiones a la API de Twitch. Además, se dispone de la clase "TwitchAuthRepository", la cual contiene todos los métodos requeridos para implementar OAuth 2.0.

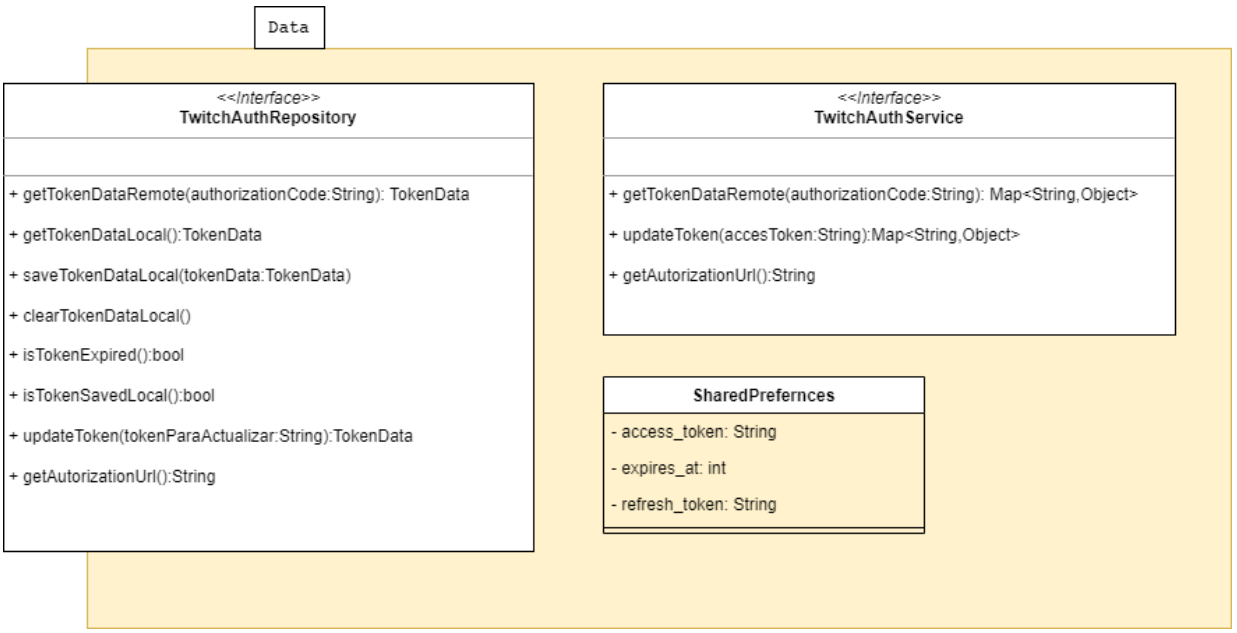

Figura 5. Diseño lógico de la parte de datos del inicio de sesión

Por último tenemos la sección “data”, conjunto de clases y métodos que acceden directamente a los datos, en nuestro caso la clase “TwitchAuthService”.

“TwitchAuthRepository” también está en “data” porque es la clase que sirve como punto medio entre los casos de uso y la API y formatea las respuestas de los servicios a objetos entendibles por el resto de la aplicación.

La clase “SharedPreferences” representa qué valores se guardan en memoria interna del dispositivo. En el caso del login hace referencia al token de acceso.

Diagrama de flujo

En la Figura 6 se muestra el diagrama de flujo para acceder a la aplicación:

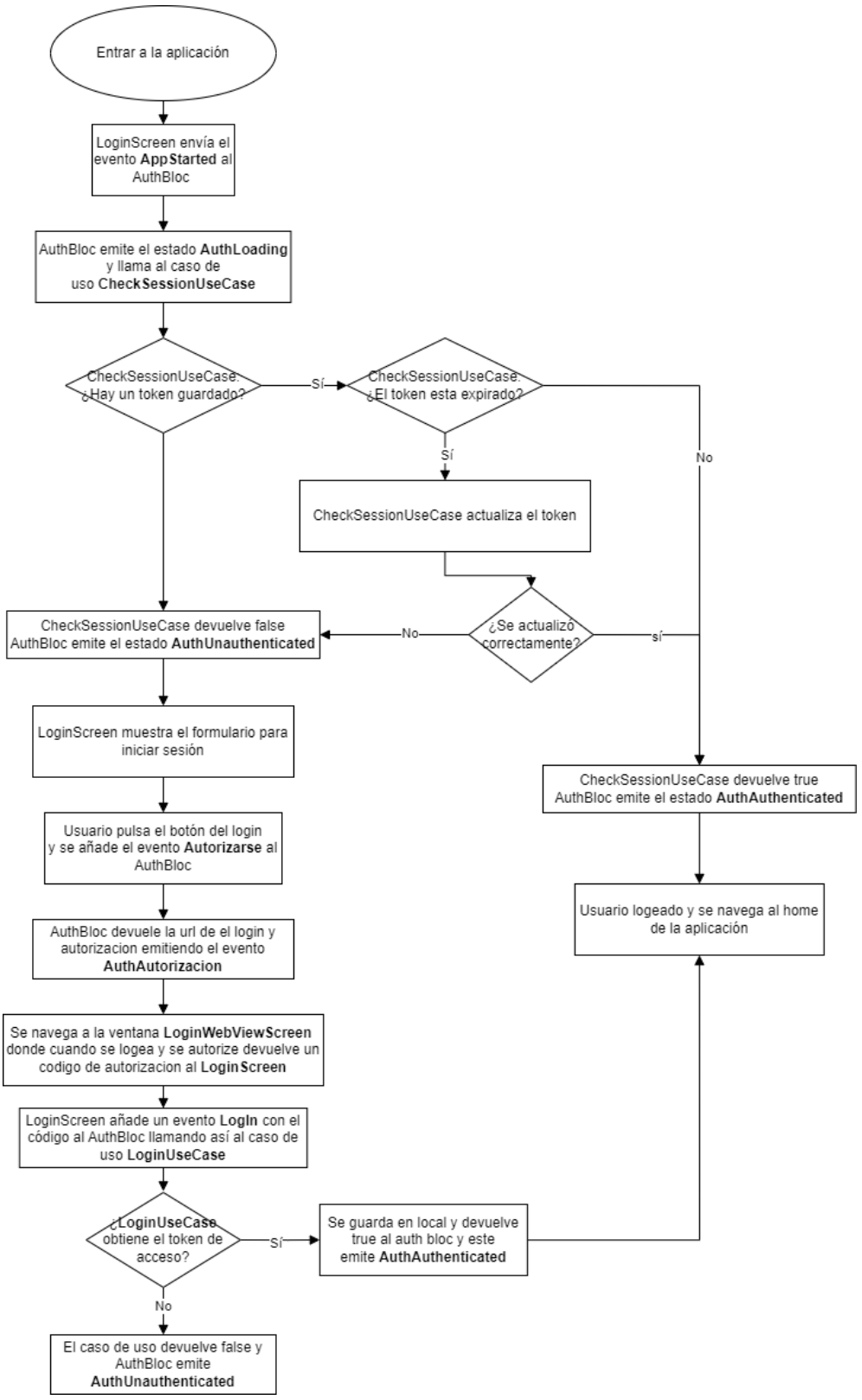

Figura 6. Diagrama de flujo del inicio de sesión

Como se puede ver en la Figura 6, al iniciar el BLoC se comprueba si hay un token guardado y si es válido. Si no está actualizado, lo actualiza y podríamos dar por autenticado al usuario. Si no hay ningún token guardado, significa que no hay usuario registrado.

4.2 Moderación del chat

Diseño de la interfaz

Para implementar la moderación del chat se necesita una ventana donde se representen los mensajes del chat, botones para cambiar los ajustes del chat y un menú con la información de un usuario que aparece al pulsar el nombre del usuario para poder ver sólo sus mensajes y poderlo vetar del chat. Tal y como se muestra en la Figura 7.

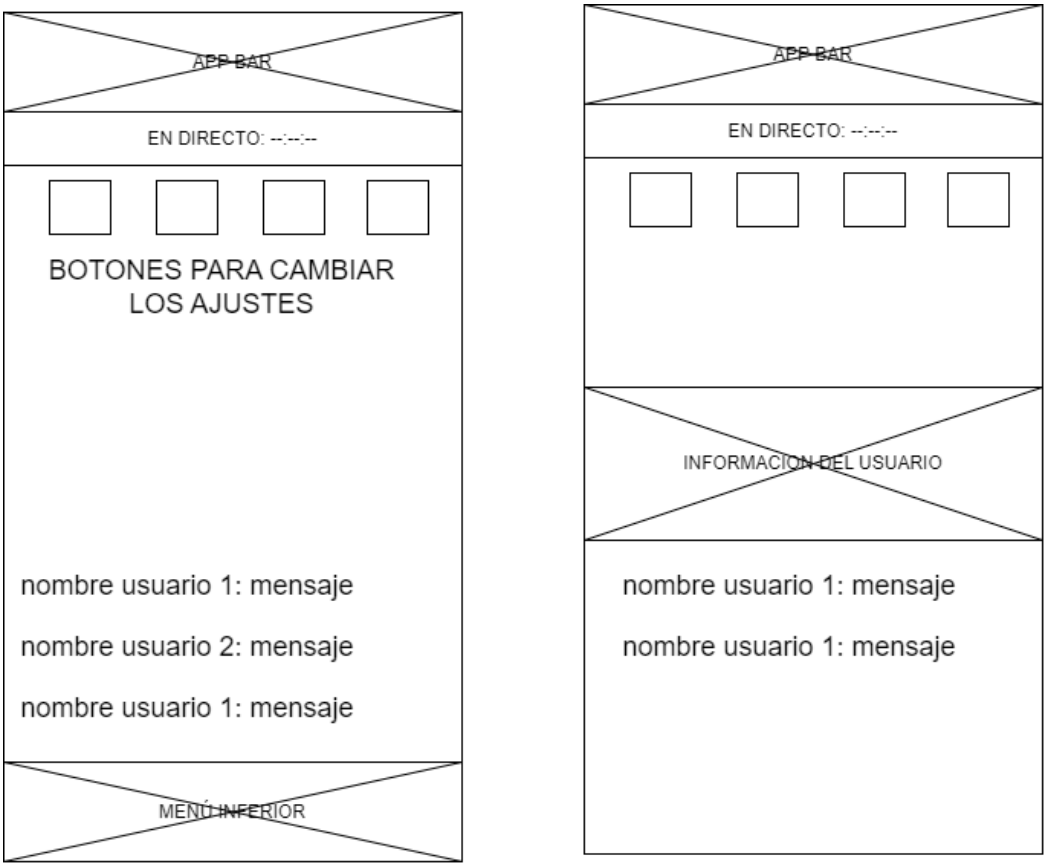

Figura 7. Wireframe de la moderación del chat

Diseño Lógico

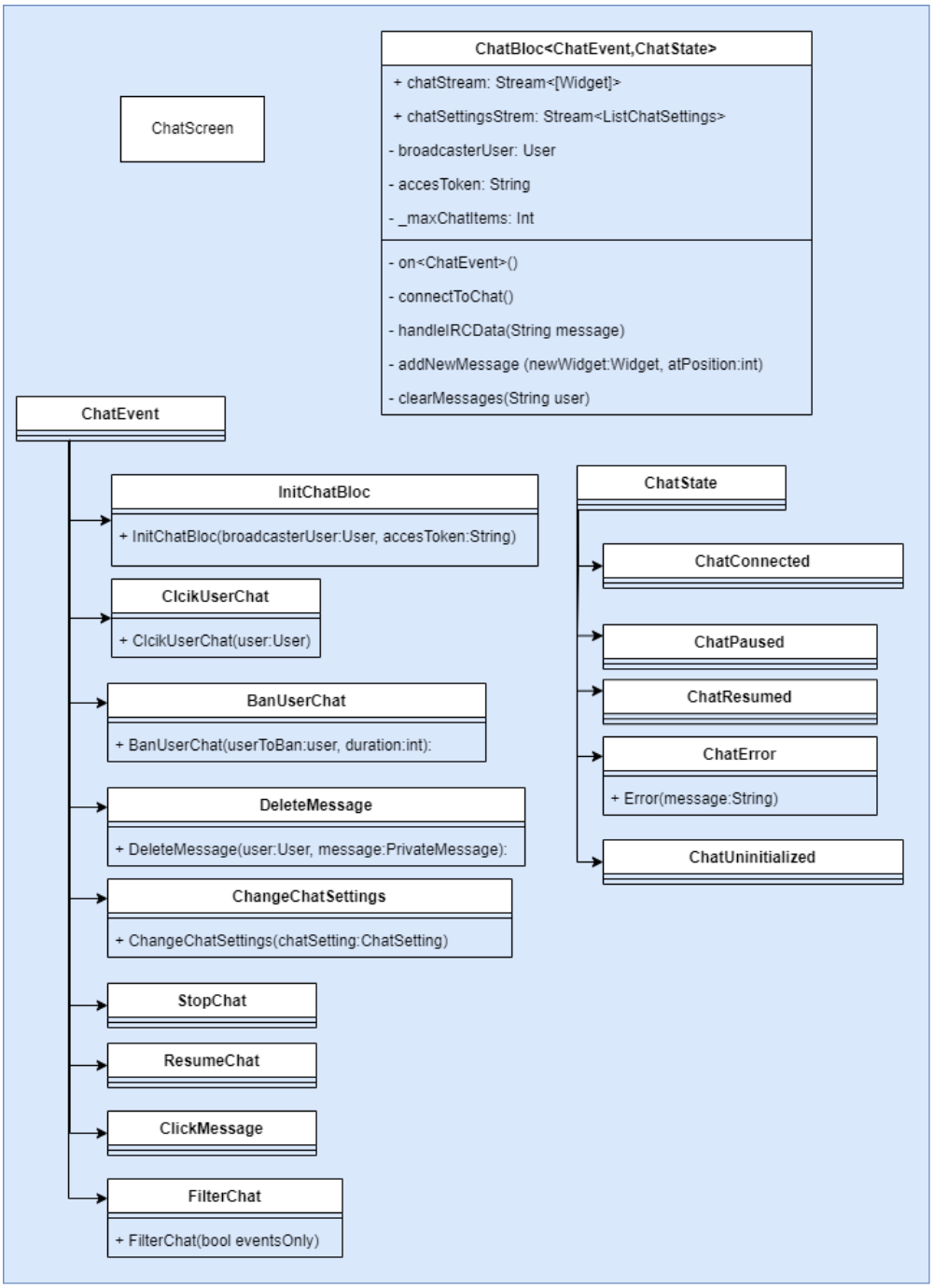

Figura 8. Diseño lógico de la parte de presentación del chat

Para desarrollar las funcionalidades del chat de Twitch se ha definido el diseño lógico mostrado en la Figura 8. Para esta funcionalidad, se ha diseñado únicamente la pantalla “ChatScreen” y un BLoC asociado, “ChatBloc”, que se comunican mediante “ChatEvents” y “ChatStates”. Un punto a destacar de este diseño es que, a parte del observador de estados del bloc, es también el observador del Stream de Widgets “chatStream”. La vista “escuchará” los cambios del “chatStream” para mostrar por pantalla los diferentes widgets (elementos visuales) que el bloc mapea según le lleguen de la conexión una vez que el bloc emita un “ChatConnected”.

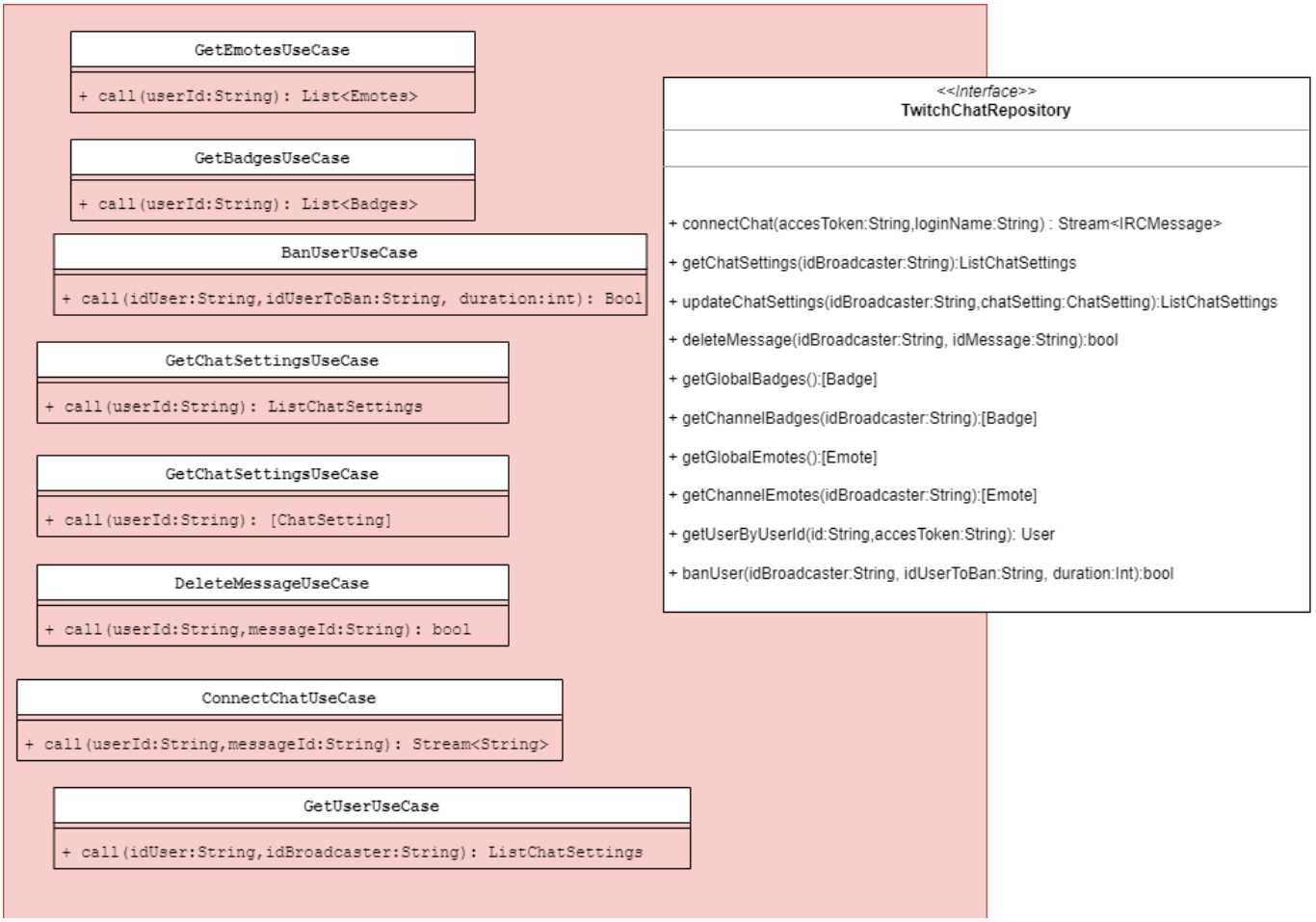

Figura 9. Diseño lógico de la parte de dominio del chat

En la capa de Domain, Figura 9 se encuentran los casos de uso que representan la lógica de negocio de la funcionalidad de integración del chat de Twitch y los objetos que representan el modelo. Tales como “GetChatSettingsUseCase” para obtener los ajustes del chat de Twitch o “BanUserUseCase” para vetar del chat a un usuario en concreto.

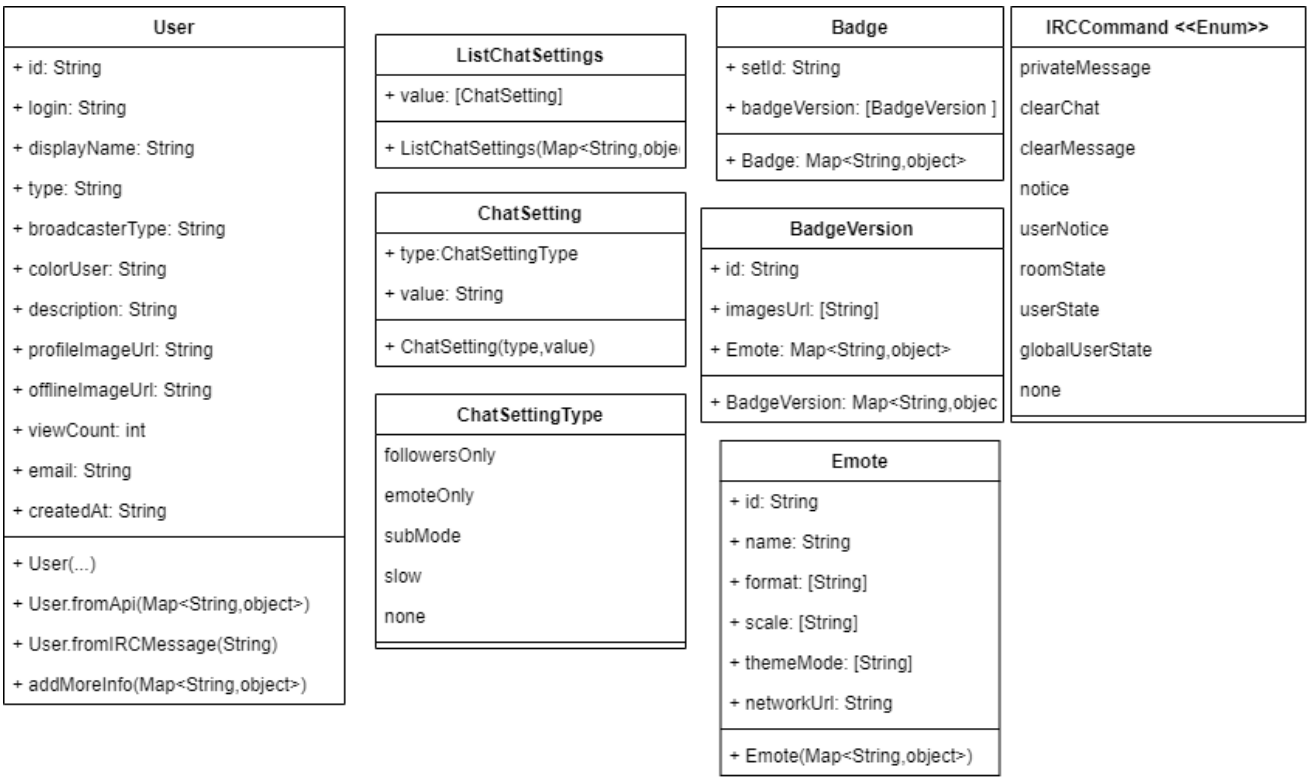

Figura 10. Diseño del modelo del chat

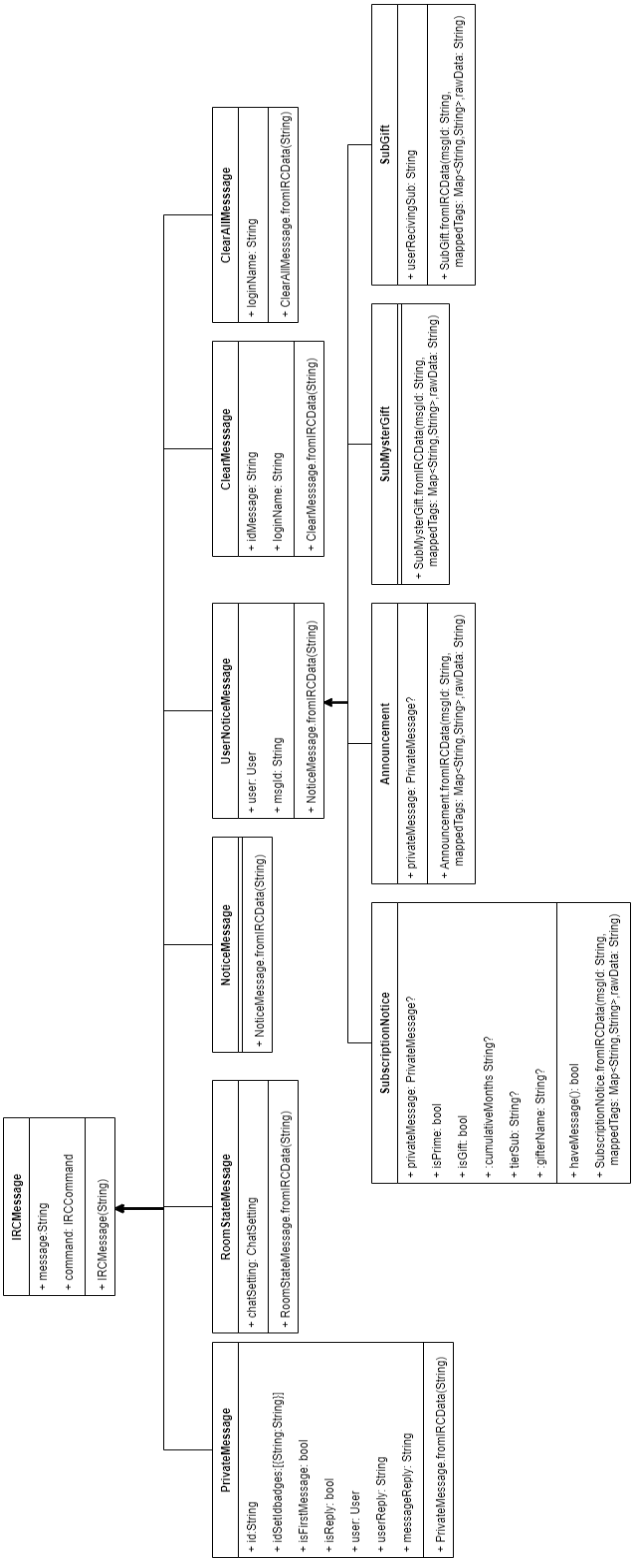

Figura 11. Diseño de clases de los IRCMessages del chat

Para poder implementar las funcionalidades del chat de Twitch se han definido los siguientes modelos de datos. En la Figura 10 se definen los comandos del chat IRC, los emoticons, los emblemas, los usuarios y los ajustes del chat. En la Figura 11 se muestran los tipos de mensajes que pueden venir desde el servicio IRC.

Figura 12. Diseño lógico de la parte de data del chat

En la Figura 12 también está definida la capa de datos del chat. Se representan dos servicios, “TwitchApiService” qué es el encargado de acceder a la API de Twitch con todos los métodos necesarios para el funcionamiento de nuestra app y “TwitchIRCService” que es el encargado de acceder a la conexión IRC de Twitch para obtener los mensajes del chat.

4.3 Control del OBS

Diseño de la interfaz

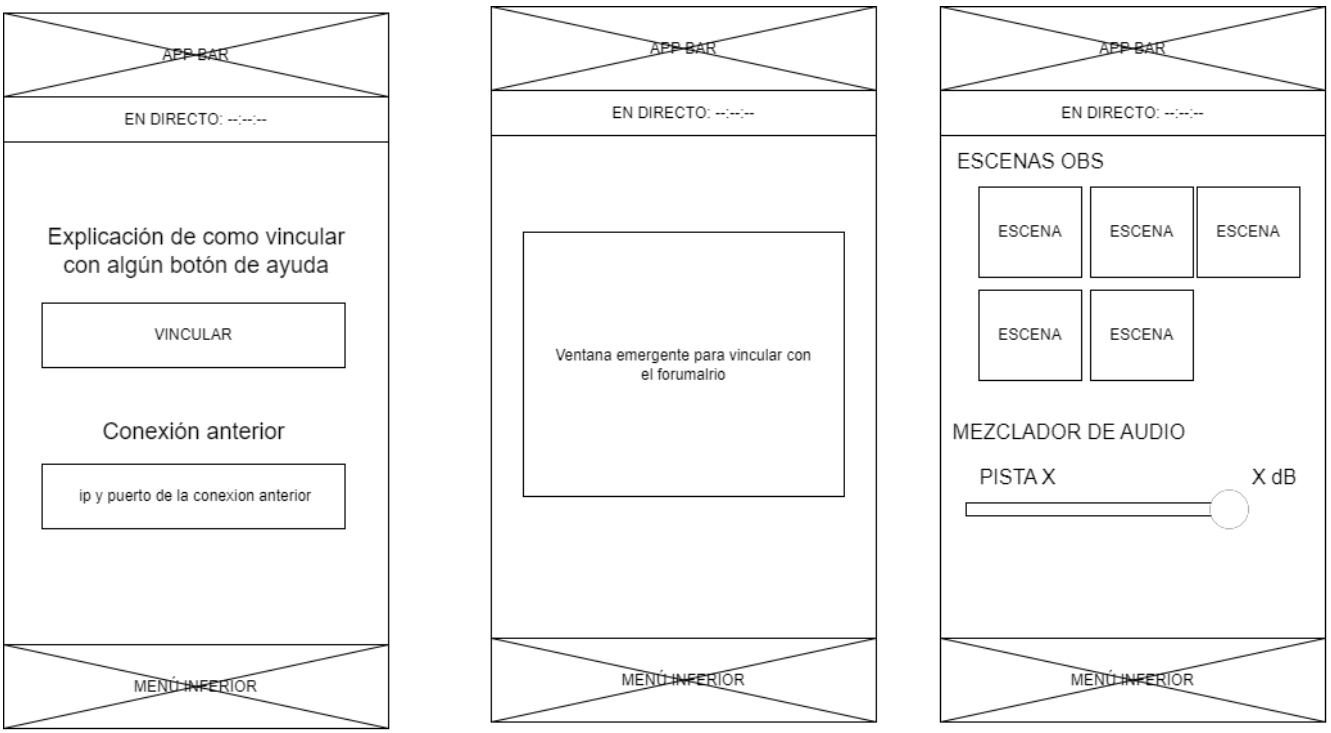

Figura 13. Wireframe del control del OBS

Para implementar el control del OBS necesitamos las siguientes funcionalidades:
una ventana con botones para poder vincular al usuario al OBS y otra con todos los controles para cambiar las escenas y cambiar el volumen de las pistas de audio. En la Figura 13 se ha definido estas funcionalidades en un wireframe.

Diseño Lógico

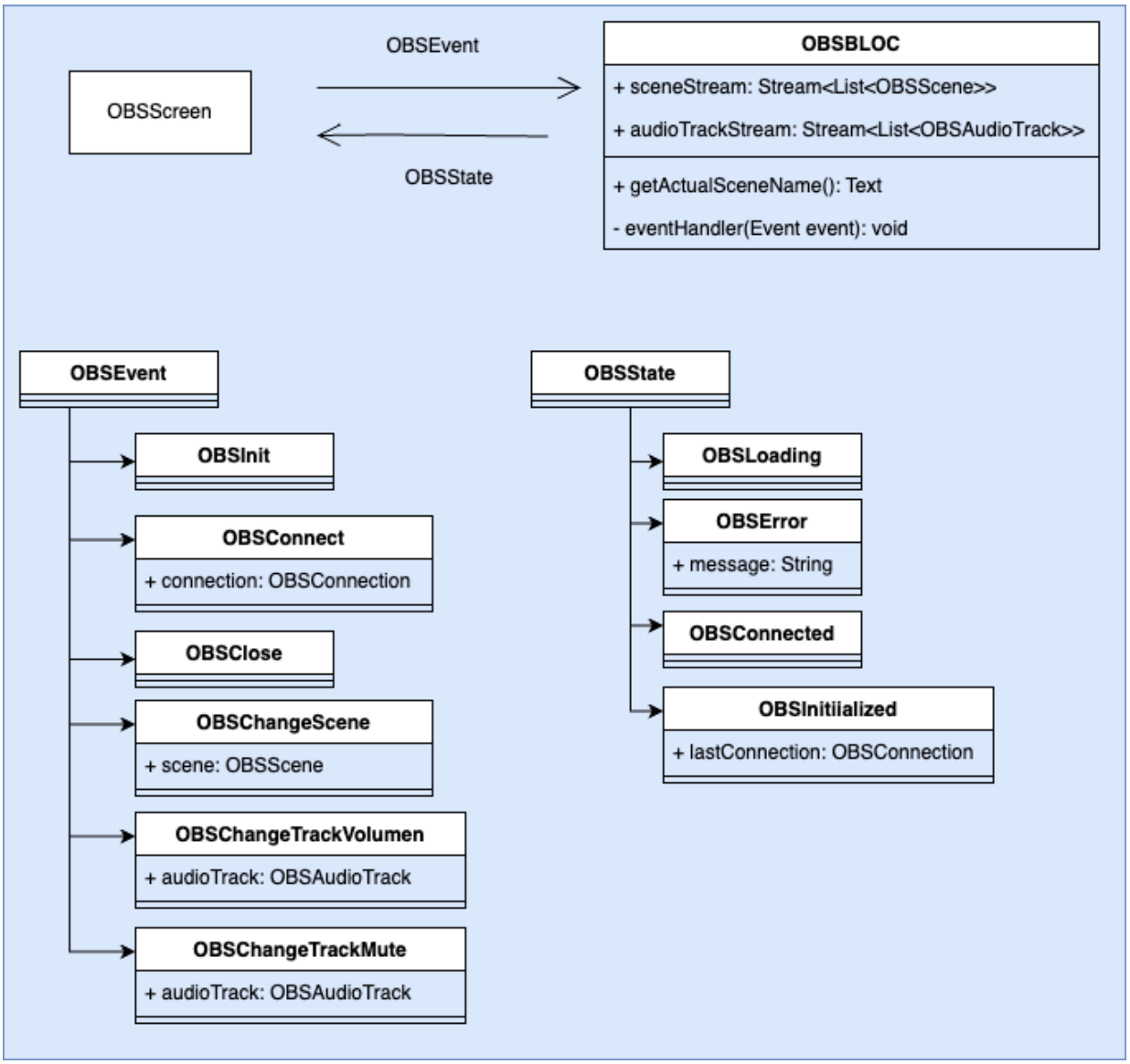

Figura 14. Diseño de la capa de presentación del control del OBS

En la Figura 14 se ha definido el diseño lógico de la capa de presentación para el control del OBS. Tenemos una sola pantalla “OBSScreen” que se comunica con eventos del tipo “OBSEvent”, para conectarse, cambiar la escena, cambiar el volumen de una pista de audio o silenciar una pista de audio. Además, una vez el BloC devuelve un estado de “OBSConnected”, el BloC escuchará los cambios a tiempo real del OBS para actualizar la vista a través de los atributos “sceneStream” y “audioStream”, los cuales la vista escuchará para actualizarse.

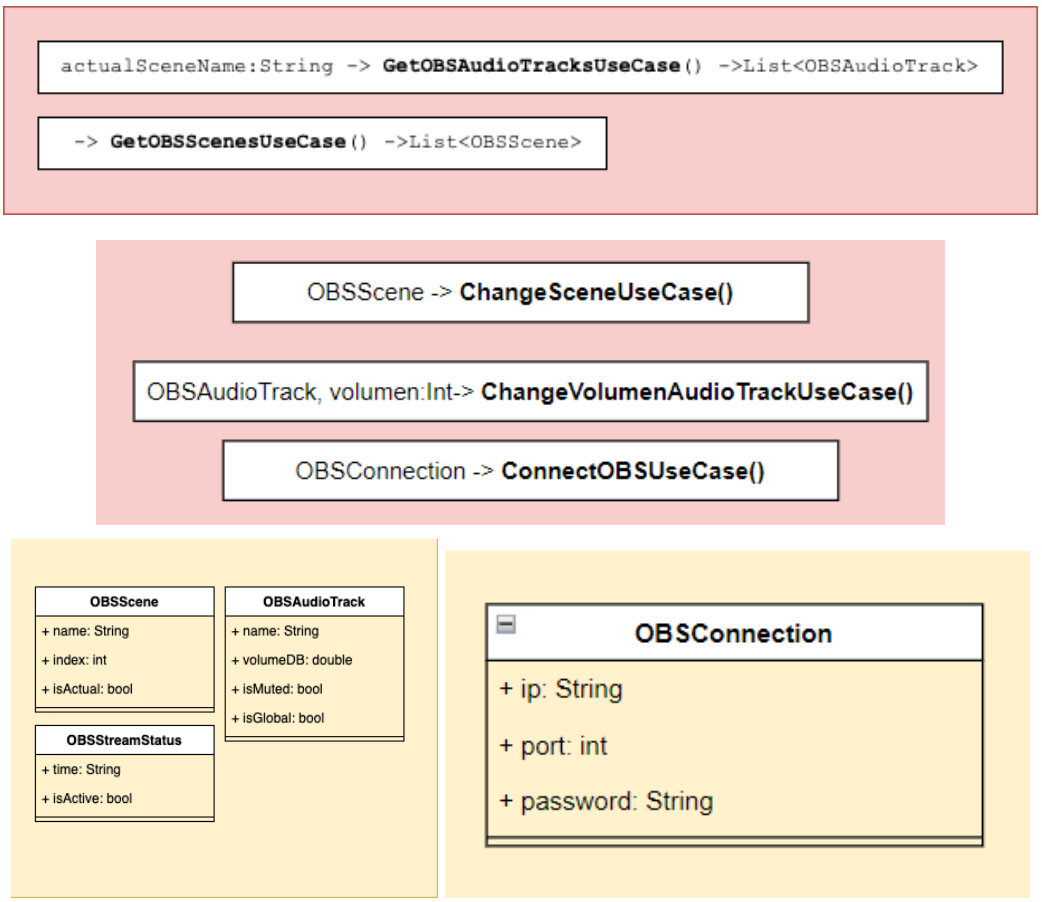

Figura 15. Diseño de la capa de dominio del control del OBS

La Figura 15 representa el dominio del control del OBS con los casos de uso necesarios para controlar el OBS: conectarse, obtener las pistas de audio y las escenas y poder cambiar la escena y el volumen de las pistas. Además, se definen los objetos que representan una escena, una pista de audio, una conexión y el estado del directo, que se emitirá a través de un flujo de datos o stream para representar si está en directo o no y, si lo está, cuanto tiempo lleva.

Figura 16. Diseño de la capa de data del control del OBS

En la Figura 16 se muestra el diseño de la capa de data del OBS con el servicio y todos sus métodos para comunicar la aplicación mediante websocket con el OBS. Como se puede ver, a diferencia de las otras implementaciones, no se ha realizado repositorio debido a la simpleza de la implementación.

4.4 Ajustes

Diseño de la interfaz

Para los ajustes de la aplicación, Figura 17, se ha diseñado una interfaz que muestra un formulario para poder cambiar el título y la categoría del directo y un botón para cerrar sesión y salir de la aplicación. El formulario está compuesto por un campo de texto para cambiar el título del directo y el campo que muestra la categoría del directo que al pulsar abre una ventana nueva con un buscador de categorías del directo. Al modificar uno de los dos campos se mostrará un botón para guardar los cambios.

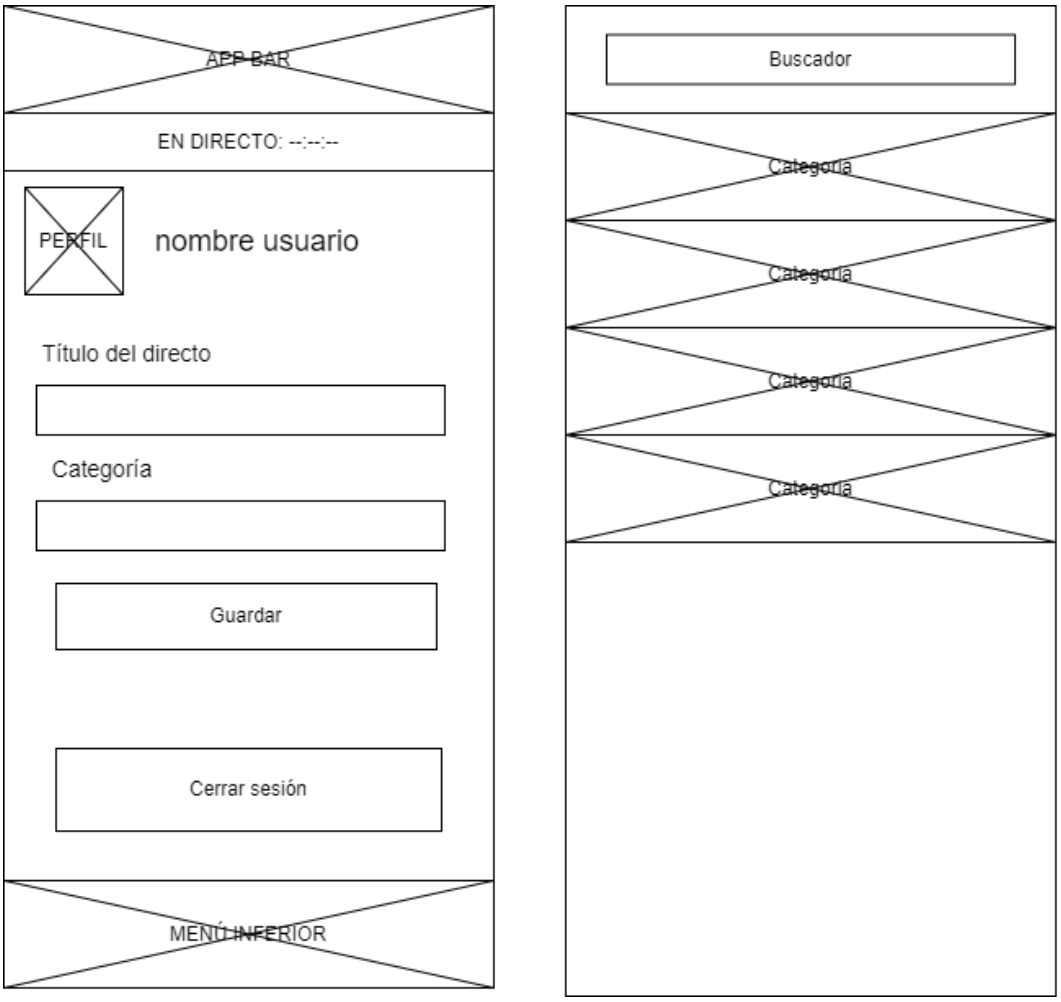

Figura 17. Wireframe de los ajustes

Diseño Lógico

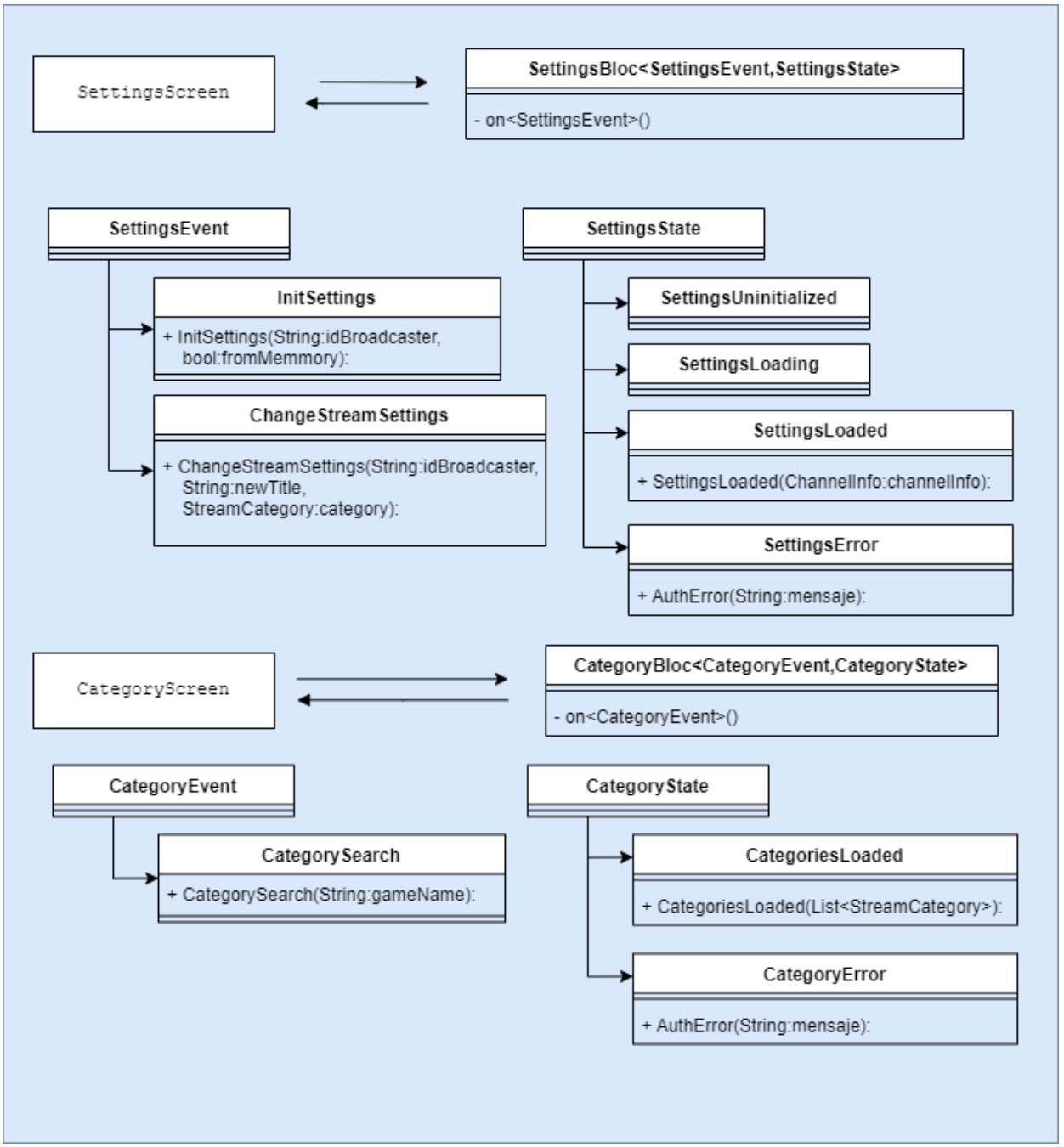

Figura 18. Diseño lógico de la presentación de los ajustes

La Figura 18 muestra el diseño lógico de la capa de presentación de los ajustes de la aplicación. Para implementar esta funcionalidad se incluye una clase BLoC por cada ventana y sus respectivos eventos y estados para poder gestionar el flujo de datos. Uno para obtener y guardar los ajustes, “SettingsScreen” y “SettingsBloc” y otra ventana para buscar una categoría “CategoryScreen” con su “CategoryBloc”. Ambos con sus respectivos eventos y estados.

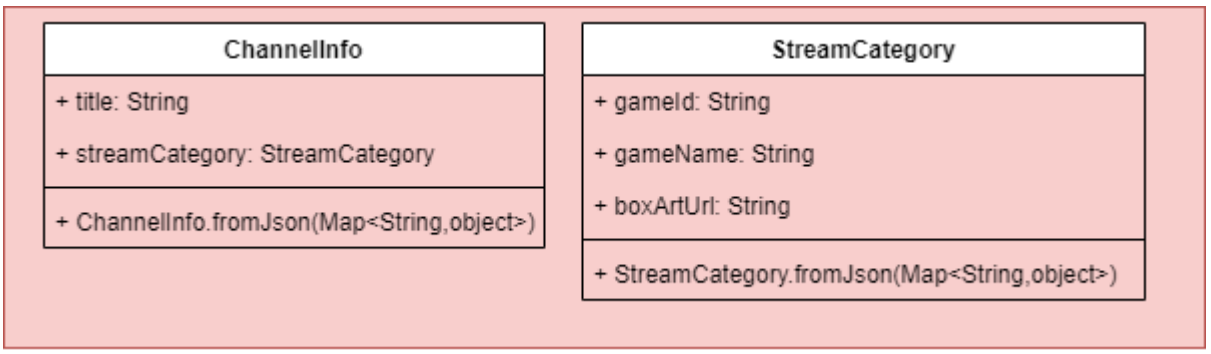

Figura 19. Diseño lógico del dominio de los ajustes

Debido a la poca complejidad de la funcionalidad, se han obviado los casos de uso para agilizar el desarrollo.
La capa de dominio, Figura 19, sólo estará compuesta del modelo que representa la información del canal y una categoría del directo.

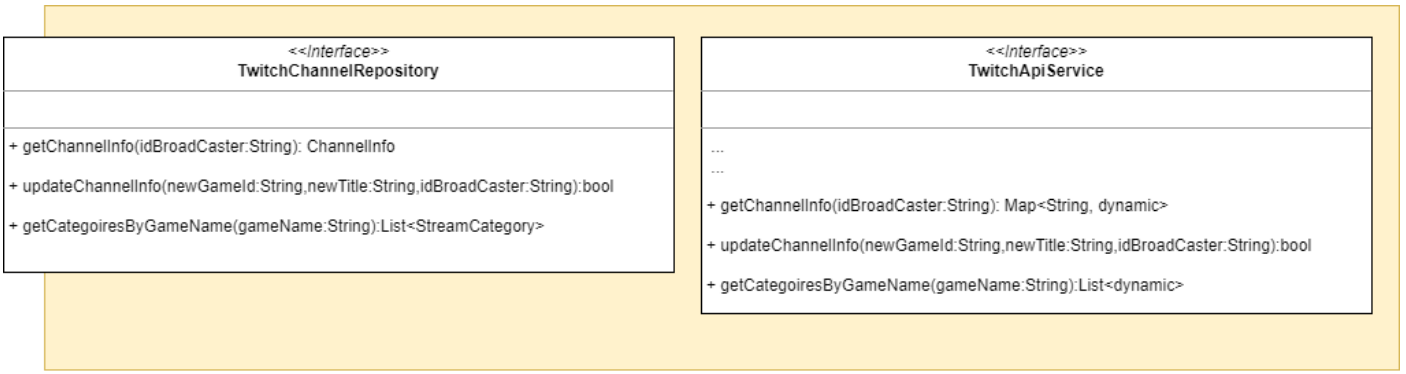

Figura 20. Diseño lógico de la capa de data de los ajustes

La Figura 20 representa la capa de data con un repositorio “TwitchChannelRepository” con los métodos necesarios para obtener la información del canal y las categorías por nombre. El repositorio accede al “TwitchApiService” al cual se le han añadido tres métodos más para obtener la información que necesita el repositorio.

En este capítulo dedicado al diseño de la aplicación, se presentaron los wireframes, los diseños lógicos y los diagramas UML de la aplicación. Estos diseños son importantes para una correcta implementación, la cual se va a explicar con más detalle en el siguiente capítulo.

Capítulo 5. Implementación

En esta sección se muestra cómo se han implementado los casos de uso definidos en el apartado 3 siguiendo los diseños lógicos planteados en la sección 4.

5.1 Inicio de sesión

El inicio de sesión de la aplicación se ha realizado mediante la API de Twitch. Los pasos que se han seguido para implementarlo se detallan a continuación.

Primero, debemos registrar nuestra aplicación en el portal de desarrolladores de Twitch (https://dev.twitch.tv/). Esto implica crear una cuenta de desarrollador y proporcionar detalles como el nombre de la aplicación, la descripción y la URL de redireccionamiento después del inicio de sesión. Una vez que la aplicación esté registrada, obtendremos las credenciales de autenticación necesarias para interactuar con la API de Twitch, el **Cliente ID y un Cliente Secreto** asociados a nuestra aplicación.

En la interfaz de usuario de nuestra aplicación al hacer clic en el botón de inicio de sesión, nuestra aplicación redirige al usuario a la **página de inicio de sesión de Twitch** mediante un webview. Una vez que el usuario haya iniciado sesión en Twitch, se le solicita que **otorgue permisos** a nuestra aplicación para acceder a ciertos datos y funcionalidades.

Después de que el usuario conceda los permisos, Twitch redirige al usuario de vuelta a la **URL de redireccionamiento** configurada en nuestra aplicación.

Una vez que el usuario haya sido redirigido a nuestra aplicación, recibiremos un código de autorización en la URL de redireccionamiento. Utilizando este **código de autorización**, nuestra aplicación realizará una solicitud a la API de Twitch para intercambiarlo por un **token de acceso válido**. Este token de acceso se utilizará posteriormente para autenticar y autorizar las solicitudes a la API en nombre del usuario.

Es importante almacenar de manera segura el token de acceso obtenido para futuras solicitudes a la API en nombre del usuario. En este caso se guarda en *sharedpreferences* de la aplicación, que es un almacenamiento interno encriptado.

Una vez que el usuario ha iniciado sesión correctamente, podemos utilizar el token de acceso para acceder a los recursos de la API de Twitch.

Esta implementación está relacionada con el diagrama de flujo de la figura 8.

El resultado final se muestra en la Figura 22.

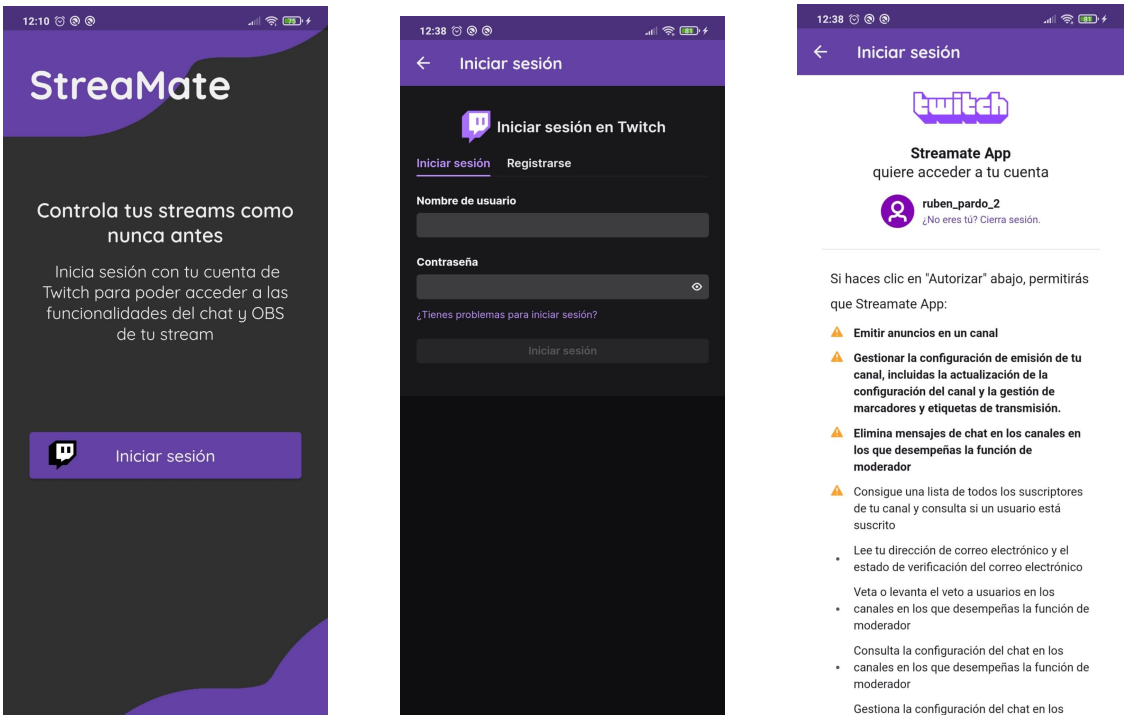

Figura 22. Diseño de la pantalla de inicio de sesión de la aplicación

5.2 Moderación del chat

En esta sección se explica cómo se han implementado los casos de uso del 3 al 7 y el 14, que son los que engloban la integración del chat y su moderación.

Utilizaremos un cliente de **IRC** (Internet Relay Chat) (Oikarinen, 1993) para conectarnos al chat de Twitch. Una vez que establecemos la conexión con el servidor de IRC de Twitch, recibiremos mensajes de chat en formato IRC, estos mensajes se mapean a los objetos del modelo (ver figura 13) y posteriormente a widgets de Flutter para representarlos visualmente en la interfaz de usuario del chat. A través de este cliente también llegan mensajes de estado del chat, que se usan para **actualizar de manera dinámica los valores de los ajustes del chat.**

Cada mensaje del chat se representa como un widget que mostrará el nombre de usuario, el contenido del mensaje y otros detalles relevantes.

Para **pausar el chat**, se proporciona un botón o una acción en la interfaz de usuario que detiene la actualización de nuevos mensajes. Al hacer clic en un widget que representa un mensaje del chat, aparece un botón para **eliminar ese mensaje** específico.

Al hacer **clic en el nombre de usuario** en un mensaje del chat, se muestra un menú inferior que contiene la información relevante del usuario. Dentro de este menú, se incluye un sub chat que muestra solo los mensajes enviados por ese usuario en particular. Además se muestran dos botones para **vetar o expulsar** temporalmente al usuario del chat.

Para **filtrar el chat** y mostrar solo los mensajes de suscripciones, se indica un botón en el menú superior que active o desactive este filtro. Al activar el filtro de suscripciones, sólo se mostrarán los mensajes enviados por usuarios suscritos.

En la parte superior del chat, se muestran los botones que permiten al usuario cambiar los **ajustes del chat**, el modo de suscripción, modo emoticonos, modo lento y modo seguidor. Al hacer clic en el modo lento y modo seguidor, aparecerá una ventana emergente que permite al usuario seleccionar la configuración deseada.

Respecto al rendimiento, se ha implementado un sistema para mostrar solo una cantidad limitada de mensajes para no saturar la aplicación.

El resultado final se muestra en la Figura 23.

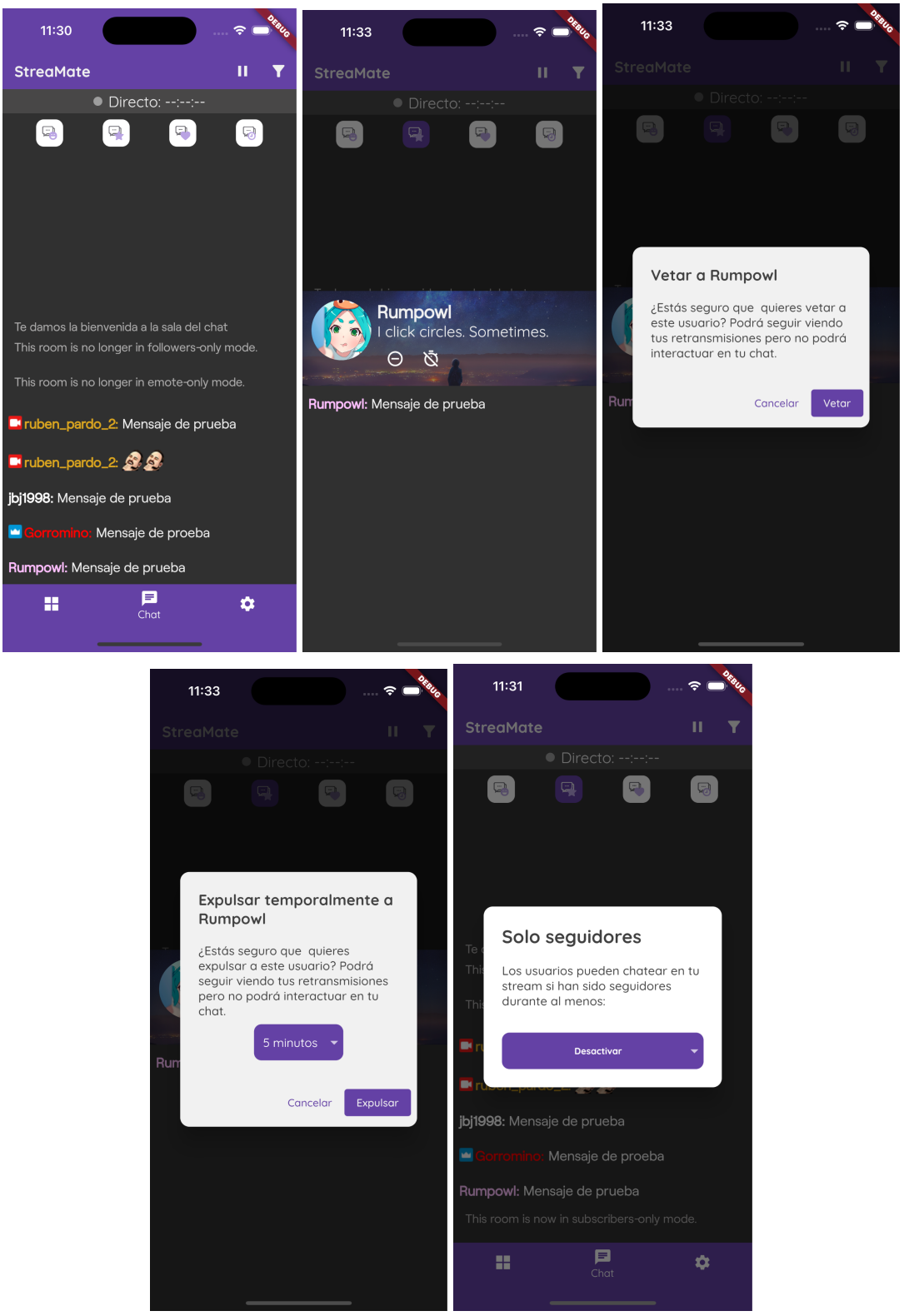

Figura 23. Diseño de las pantalla de la integración del chat

5.3 Control del OBS

En esta sección se explica cómo se han implementado los casos de uso del 9, 10, 11 y 15, que son los relacionados con la integración del OBS.

Utilizaremos una biblioteca compatible con Flutter, como obs_websocket (faithoflifedev, 2022), para establecer una conexión con OBS Studio.

El usuario ingresa la dirección IP, el puerto y la contraseña de OBS a través de un diálogo de configuración. Después de **establecer una conexión** exitosa con OBS, se realizan solicitudes a través del WebSocket para obtener la lista de escenas disponibles y las pistas de audio.

Mantendremos una conexión activa con OBS y estaremos atentos a los eventos de **cambio en tiempo real,** como cambios de escenas o ajustes de audio, mediante los "streams" proporcionados por la biblioteca obs_websocket para recibir notificaciones de cambios y actualizar dinámicamente la interfaz de usuario de Flutter en respuesta a ellos.

Una vez vinculado se muestran la **lista de escenas** obtenidas de OBS en forma de botones o elementos interactivos en la interfaz de usuario. Al hacer clic en una escena, se envía una solicitud a través del WebSocket para cambiar a esa escena específica en OBS.

El mismo proceso se repite con las **pistas de audio** obtenidas de OBS. Se muestran en dos grupos separados, "Audio General" y "Audio de Escena", pero en vez de botones, se utilizan controles deslizantes para ajustar los niveles de volumen y un botón para silenciar la pista. Los cambios envían solicitudes a través del WebSocket para actualizar los valores.

Cuando se detecta el **inicio de la transmisión**, se actualiza el contador en la parte superior de la interfaz del control del OBS y lo resaltamos en rojo para indicar que se está en directo.

El resultado se muestra en la Figura 24.

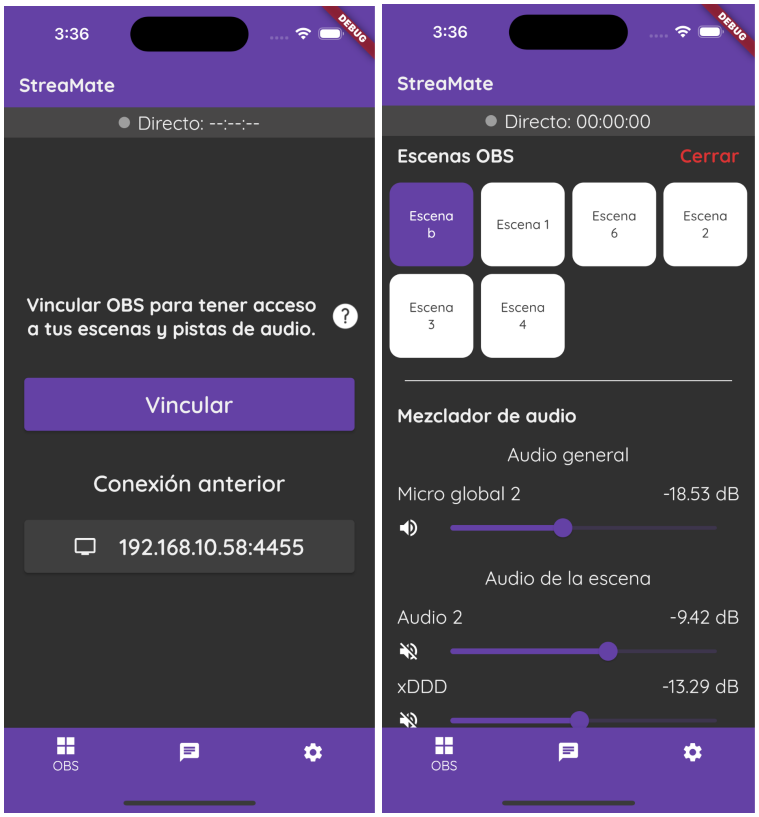

Figura 24. Diseño de las pantalla de la integración del OBS

5.4 Ajustes

En esta sección se explica cómo se han implementado los casos de uso 12 y 13, que son los relacionados con los ajustes del directo y salir de la aplicación.
Se ha realizado una pantalla con un campo de texto para el **título del directo** y otro para la **categoría del directo**. Este último funciona como un botón que redirige al usuario a otra pantalla con un buscador en la parte superior y el listado resultado debajo. Al seleccionar una categoría volveremos a la pantalla de ajustes. Cuando la aplicación detecta que la categoría o el título ha cambiado aparecerá un botón para **guardar los cambios.**

En la parte inferior de esta pantalla habrá siempre un botón para **cerrar sesión** de la cuenta de Twitch el cual redirigirá al usuario a la pantalla del Inicio de Sesión.

El resultado se muestra en la Figura 25.

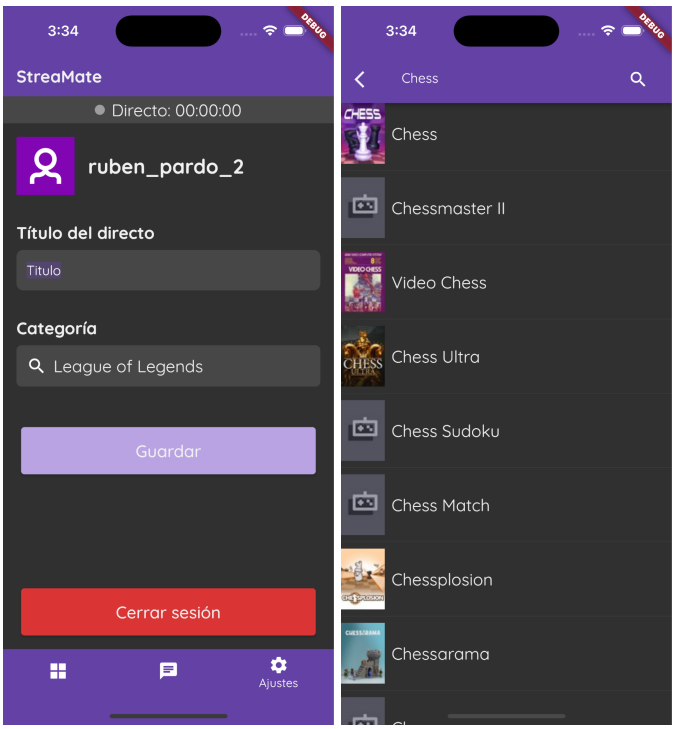

Figura 25. Diseño de las pantalla de los ajustes

Capítulo 6. Manuales

En este capítulo se explican los manuales para poder poner en marcha el proyecto para su mantenimiento y desarrollo. Para ello debemos configurar la consola de Twitch para la API y compilar la aplicación en un dispositivo Android o iPhone.

6.1 Compilación de la aplicación en dispositivos Android

Los siguientes pasos explican cómo poner en marcha el proyecto y compilarlo en un dispositivo Android:

1. Instalar y configurar la librería de Flutter en el equipo.
2. Conectar un dispositivo por USB o lanzar un emulador.
3. Ejecutar el comando “flutter run”.

6.2 Compilación de la aplicación en dispositivos iPhone

Los siguientes pasos explican cómo poner en marcha el proyecto y compilarlo en un dispositivo Iphone:

1. Instalar y configurar la librería de Flutter en el equipo.
2. Tener el xCode configurado con una cuenta de desarrolladores de Apple.
3. Abrir el proyecto en la terminal y navegar hasta la carpeta iOS y ejecutar el comando “pod install”.
4. Conectar un dispositivo por USB o lanzar un emulador.
5. Ejecutar el comando “flutter run”.

6.3 Configuración de la Consola de Twitch para la API

Para configurar la consola de Twitch y utilizar su API, hay que seguir siguientes pasos:

1. Acceder al sitio web de desarrolladores de Twitch en https://dev.twitch.tv e inicie sesión en la consola de desarrolladores de Twitch.
2. Crear una nueva aplicación en la consola de desarrolladores de Twitch. Escriba un nombre descriptivo para la aplicación, seleccione el tipo de aplicación correspondiente y proporcione una URL de redireccionamiento (en nuestro caso http://localhost, porque no hace falta)
3. Una vez creada la aplicación, se obtienen el Client ID y un Client Secret. Estos son los identificadores necesarios para autenticarse y realizar llamadas a la API de Twitch desde la aplicación.

Capítulo 7. Evaluación

7.1 Pruebas de usabilidad

Las pruebas de usabilidad son una parte muy importante en el desarrollo del software para la evaluación de la experiencia de usuario de la aplicación. Estas pruebas están diseñadas para analizar cómo los usuarios interaccionan con el sistema, identificar posibles fallos y evaluar la eficiencia y satisfacción del producto.

La metodología usada para estas pruebas consistió en seleccionar a un grupo de personas objetivo de esta aplicación, explicarle el contexto de la aplicación y los diferentes posibles usos de ella para que al final rellenarán un Google Form para recabar sus opiniones.

Dado que el ámbito de este trabajo es un TFG no fue posible conseguir una muestra de población más grande, en nuestro caso la población encuestada consta de solo 5 personas. No obstante, en trabajos futuros se podría ampliar esta muestra para obtener resultados más completos.

El formulario tiene como objetivo obtener opiniones sobre la usabilidad y diseño del inicio de sesión, el control del OBS y los ajustes de usuario

Con respecto a la usabilidad y diseño del inicio de sesión el resultado obtenido fue el siguiente:

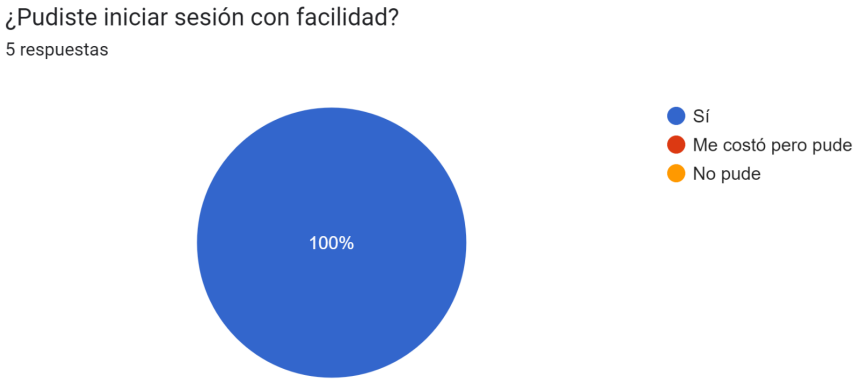

Figura 26. Gráfica de los resultados de "¿Pudiste iniciar sesión con facilidad?

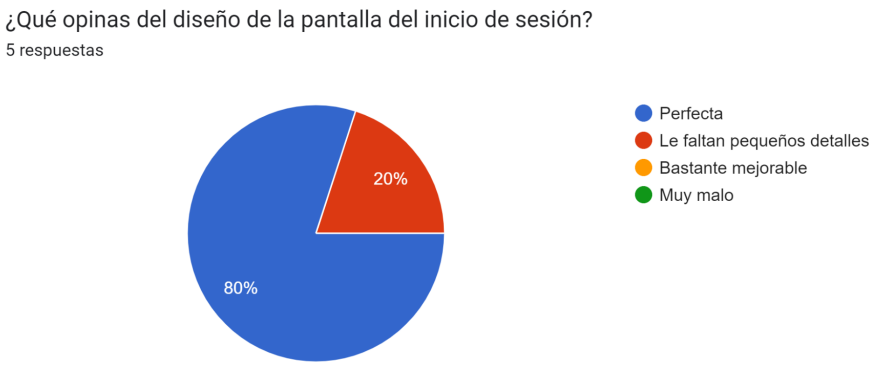

Figura 27. Gráfica de los resultados sobre la opinión del diseño del inicio de sesión

Como se puede observar en la Figura 26 el 100% de los encuestados encontraron fácil iniciar sesión. Casi la totalidad de los usuarios encuestados mostraron su conformidad con el diseño de la pantalla de inicio de sesión (Figura 27). Sin embargo, un 20% indicaron que le faltan algunos pequeños detalles como la mejora de diseño de las cargas de los “webviews”.

Para la usabilidad del componente de OBS se preguntó la facilidad para vincular la aplicación móvil a OBS y la usabilidad de los controles, las respuestas fueron las siguientes:

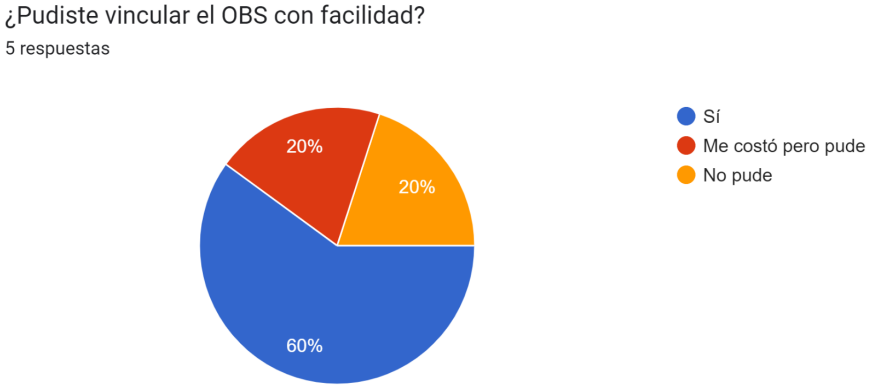

Figura 28. Gráfica de los resultados sobre la facilidad de vincular el OBS

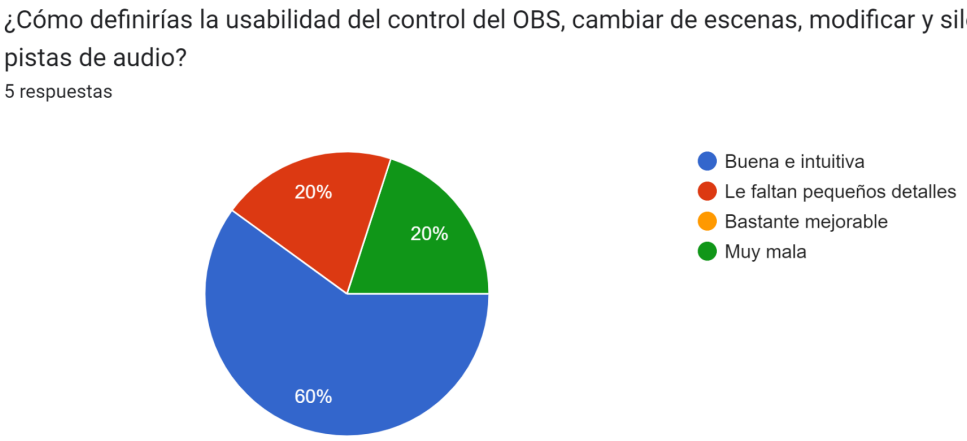

Figura 29. Gráfica de los resultados sobre la usabilidad del control del OBS

Como se ve en la Figura 28 el 20% de usuarios no pudieron vincular el OBS debido a algunas especificaciones de algunos dispositivos. El otro 20% de usuarios que tuvieron problemas fueron debido a la complejidad de buscar el host de tu OBS (ip:puerto). El 60% restante encontró sencillo el proceso de vincular el OBS. Sobre la respuesta de la usabilidad (Figura 29) corresponde con los datos anteriores (Figura 28), los que respondieron “Muy mala” fueron los que no pudieron vincular y los que respondieron que “Le faltan pequeños detalles” comentaron que se debería hacer más sencillo el proceso. En futuras versiones se prevé simplificar este proceso.

Gracias a este feedback se descubrió que en ciertos dispositivos la vinculación al OBS no funciona correctamente. Al recibir información detallada sobre los dispositivos específicos afectados, se ha podido investigar y realizar ajustes necesarios en el software para garantizar una compatibilidad más amplia y una funcionalidad óptima.

Respecto al componente del chat de Twitch preguntamos por la usabilidad de todas sus funcionalidades:

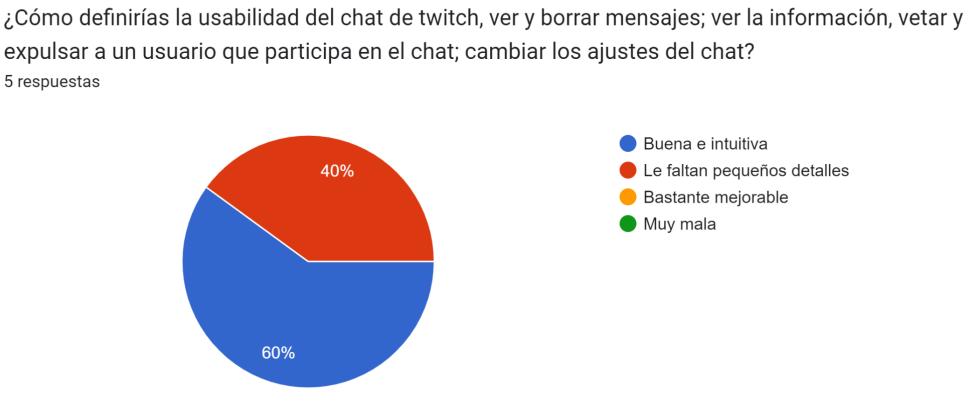

Figura 30. Gráfica de los resultados sobre la opinión de la usabilidad del chat de Twitch

Como se puede ver en la Figura 30 el 60% de los encuestados encontraron una buena usabilidad. De los 40% restantes que respondieron que le faltaban detalles añadieron comentarios adicionales que hacían referencia a la mejora de los iconos tanto de los filtros como de los ajustes del chat con el fin de que resulten más intuitivos. Por ejemplo, cambiar el icono del embudo del filtro por la estrella de suscripciones propia de Twitch. También sugirieron mejorar la calidad de los ajustes del chat.

Y para finalizar, esto fue lo que respondieron sobre los ajustes de la aplicación:

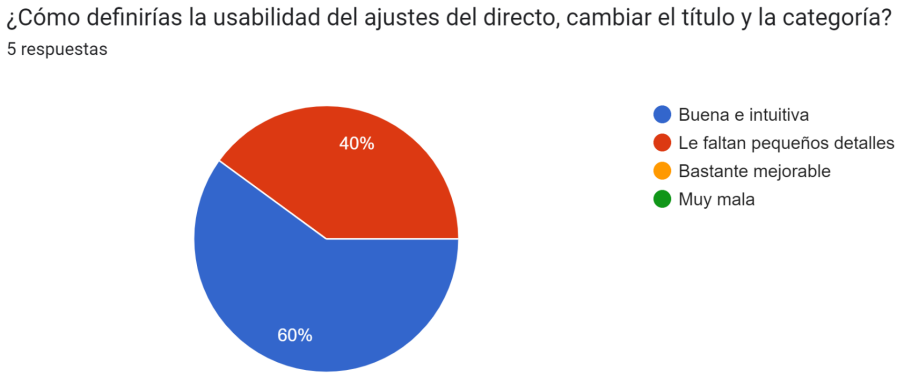

Figura 31. Gráfica de los resultados sobre la opinión de la usabilidad de los ajustes

Como se puede ver en la Figura 31 el 60% de los encuestados encontraron intuitivo la usabilidad de los ajustes. Sobre el 40% que respondieron que le faltan detalles sugirieron una sección de categorías recientes o favoritas para cambiar más rápido la categoría del directo.

Como conclusión podemos afirmar que, en general, la aplicación tiene una buena usabilidad pero que tiene ciertos fallos o mejoras que se le pueden aplicar. Gracias a esta encuesta se han podido arreglar ciertos errores y tomar nota para funcionalidades futuras como añadir las categorías favoritas o cambiar ciertos iconos para un mejor diseño.

7.2 Pruebas unitarias

Las pruebas unitarias son una parte fundamental en el desarrollo de una aplicación. Son una forma de verificar el correcto funcionamiento de cada componente o partes del código y asegurarse que funcionen de forma aislada antes de integrarlo en el sistema. El propósito principal de las pruebas unitarias es garantizar la calidad del código y detectar posibles fallos.

Para verificar la correcta funcionalidad de nuestro código se ha diseñado una completa batería de tests que prueben las funcionalidades individuales de la aplicación así como los casos de uso definidos. La suite de tests completa finalmente ha consistido en un conjunto de 36 tests de los cuales el 100% pasan con éxito.

Se ha dado prioridad a testear los casos de uso, ya que son la parte más importante de la aplicación ya que define la lógica de negocio de esta.

Se ha procurado en todo momento que los tests diseñados cumplan con las propiedades elementales de todo test unitario: independiente, automatizable, rápido, completo y reutilizable. Para ello se ha utilizado, cuando así lo ha requerido el test, herramientas de apoyo en el diseño de tests como los mocks.

A continuación se incluye un ejemplo de test para el caso de uso del login:

```
29      test("Login - nos devuelven un token valido",() async {
30
31        String authCode = "1234";
32        String url = "https://dummyurl?code=$authCode";
33
34          when(() => serviceLocator<TwitchAuthRepository>().getTokenDataRemote(
35                any(),
36              )).thenAnswer((invocation) => Future.value(TokenData.dummyValido()));
37          when(() => serviceLocator<TwitchAuthRepository>().saveTokenDataLocal(any()),).thenAnswer((invocation) => Future.value());
38
39          var result = (await loginUseCase.call(url)).fold((error) => error,(loged)=>loged); // result puede ser left de tipo MyError o right de tipo Bool
40
41          // si va bien tiene que devolver un valor right y true
42          expect(result, true);
43          // se llamo al metodo del repositorio de obtener el token del server
44          verify(() => serviceLocator<TwitchAuthRepository>().getTokenDataRemote(authCode)).called(1);
45          // guarda el token en local
46          verify(() => serviceLocator<TwitchAuthRepository>().saveTokenDataLocal(any())).called(1);
47
48
49      });
```

Figura 32. Código ejemplo test unitario Inicio sesión.

Como se puede ver en la Figura 32, las líneas 34 y 37 sirven para mockear las respuestas que se realizan dentro del caso de uso “loginUseCase”, es decir falseamos las respuestas de estas funciones para poder probar el caso de uso con mayor facilidad. Y por último, a partir de la línea 42 hacemos las comprobaciones para ver si todo ha ido correcto.

En conclusión, en este capítulo hemos llevado a cabo una evaluación del proyecto, centrándonos en las pruebas de usabilidad y los tests unitarios. A través de las pruebas de usabilidad, hemos obtenido la opinión de los usuarios sobre la interacción con el sistema y hemos identificado áreas de mejora para proporcionar una mejor experiencia. Por otro lado, los tests unitarios han sido clave para garantizar la calidad y funcionalidad del código en las diferentes unidades y componentes del proyecto.

En general, la combinación de pruebas de usabilidad y tests unitarios ha sido esencial para lograr un proyecto sólido y satisfactorio tanto para los usuarios como para los desarrolladores.

Capítulo 8. Conclusiones

En este capítulo, resumiremos los principales resultados y contribuciones de nuestro trabajo, destacando la relevancia e impacto de la aplicación desarrollada. Discutiremos la eficacia y utilidad de la aplicación, evaluaremos el cumplimiento de los objetivos planteados inicialmente y reflexionaremos sobre las limitaciones encontradas durante el proceso. Además, señalando las posibles aplicaciones futuras y las áreas de mejora identificadas.

A lo largo del proyecto, se han abordado objetivos generales y específicos para lograr una solución funcional y satisfactoria. A continuación, se presentan las conclusiones obtenidas:

En primer lugar, se ha logrado analizar el **entorno y el estado del arte**, cumpliendo con el objetivo de examinar otras posibles soluciones existentes en el campo. Esto ha permitido identificar las necesidades de los usuarios y establecer los fundamentos para el desarrollo de la aplicación.

El objetivo de la **integración de Twitch** ha sido logrado satisfactoriamente. Los usuarios pueden acceder a la aplicación con los credenciales de Twitch para visualizar los mensajes del chat de su directo, además de contar con herramientas para gestionar y moderar los mensajes de forma efectiva a través de la API pública de Twitch.

Otro de los objetivos ha sido el **control remoto del OBS** desde la aplicación. La aplicación desarrollada permite a los usuarios vincular el OBS y controlar sus escenas y ajustes de audio de manera remota. Esto proporciona una mayor flexibilidad y comodidad para los streamers al realizar los directos.

La **compatibilidad con los dispositivos Android e iOS**, se ha logrado exitosamente. Los usuarios pueden acceder a la aplicación desde móviles que hagan uso de uno de estos sistemas operativos, asegurando una amplia cobertura de usuarios.

Realizar una amplia batería de **pruebas exhaustivas** ha sido clave para garantizar la estabilidad, seguridad y funcionalidad de la aplicación. Se realizaron pruebas de usabilidad y de integración para detectar posibles errores y asegurar la calidad de la aplicación

Todo el proceso se ha documentado, desde el análisis inicial hasta la implementación y las pruebas, además de proporcionar guías que facilitan la instalación y configuración de la aplicación.

Nuestro objetivo principal era ofrecer una herramienta a aquellos “streamers” que no tienen los suficientes recursos para poder gestionar y moderar las retransmisiones en vivo de Twitch. Muchas de las soluciones actuales (vistas en la Tabla 1) ofrecen estas gestiones pero en productos separados o suelen ser costosas. En este trabajo se ha conseguido realizar una aplicación que unifique las funcionalidades más importantes de la moderación del chat de Twitch, como vetar a usuarios, borrar mensajes o cambiar los ajustes del chat y las funcionalidades básicas de la gestión del OBS.

En definitiva, este trabajo ha cumplido los objetivos generales establecidos. Se ha demostrado la viabilidad de la solución descrita y se han obtenido resultados satisfactorios en términos de funcionalidad y usabilidad.

8.1 Trabajo a futuro

Además de los objetivos conseguidos, este proyecto abre camino a **trabajos futuros** para mejoras.
A continuación se listan algunas de las propuestas de mejora a aplicar:

- Una mejora en la interfaz gráfica. Aunque los resultados en el capítulo anterior eran positivos respecto a este tema, siempre hay espacio de mejora para la experiencia de usuario. Además de añadir los cambios también comentados en el capítulo 7, cambio de iconos y hacer más fácil vincular el dispositivo al OBS.
- Realizar más pruebas de usabilidad, de integración y añadir pruebas de rendimiento para optimizar la aplicación y ofrecer un mejor producto.
- Integrar otras plataformas de directos además de Twitch, como Youtube Live o Facebook Gaming y así ofrecer un solución más amplia y versátil a los usuarios.
- Publicar en los principales repositorios, Apple Store y Google Play para dar acceso al público a la aplicación.

8.2 Relación del trabajo desarrollado con las asignaturas con el grado

Programación 1 y Programación 2: Estas asignaturas dan los fundamentos de la programación, enseñando los conceptos básicos y avanzados de la programación estructurada y orientada a objetos, además de aprender a analizar y diseñar software. Los conocimientos adquiridos en estas asignaturas fueron fundamentales para desarrollar la lógica y la estructura del código de la aplicación.

Diseño de interfaces y experiencia de usuario: Esta asignatura proporciona conocimientos sobre los principios de diseño de interfaces y la experiencia del usuario. Estos conocimientos fueron aplicados al desarrollar la interfaz de la aplicación, como los Wireframes, asegurándose de que fuera intuitiva, fácil de usar y estéticamente agradable para los usuarios.

Aplicaciones para dispositivos móviles: En esta asignatura se enseñan conceptos básicos del desarrollo de aplicaciones móviles y de la librería Android. Aunque se haya desarrollado el proyecto con una librería diferente como lo es Flutter, una librería para desarrollo de aplicaciones multiplataforma, ha sido realmente relevante para las configuraciones necesarias para la compilación del proyecto de Android además de ciertos conceptos básicos del desarrollo de aplicaciones móviles como arquitecturas, ciclo de vida o navegación entre otras.

Redes y servicios telemáticos, Redes de área local e Integración de redes : Estas asignaturas proporcionan conocimientos sobre la comunicación en red y los servicios telemáticos. Estos conocimientos fueron relevantes para comprender la comunicación REST con la API de Twitch, la comunicación UDP con el servicio IRC del chat de Twitch y la conexión WebSocket con el OBS

En la asignatura de **Diseño de Interfaces**, se adquieren conocimientos sobre los principios de diseño de interfaces de usuario y la experiencia del usuario. Estos conocimientos son fundamentales para crear una interfaz intuitiva, atractiva y fácil de usar en la aplicación.

Por otro lado, la **metodología CDIO** se enfoca en el desarrollo de proyectos de ingeniería desde una perspectiva integral y orientada a la resolución de problemas reales. Esta metodología se basa en los siguientes pasos: Concebir, Diseñar, Implementar y Operar. Esta asignatura ha ayudado a saber cómo aplicar actividades como la definición de los requisitos del proyecto, el diseño de la arquitectura de software, la implementación de las funcionalidades requeridas y la operación del software resultante.

Bibliografía

Cervera, I. (2021, April 9). *Stream Deck, el panel de control que facilita la vida de los streamers*. Geekno. Consultada February 1, 2023, URL: https://www.geekno.com/stream-deck-el-panel-de-control-que-facilita-la-vida-de-los-streamers.html

faithoflifedev. (2022, Noviembre 18). *obs_websocket | Dart Package*. Pub.dev. Consultada June 1, 2023, URL: https://pub.dev/packages/obs_websocket

felangel.dev. (2023). *mocktail | Dart Package*. Pub.dev. Consultada July 1, 2023, URL: https://pub.dev/packages/mocktail

Flutter. (2023, March 27). Flutter - Build apps for any screen. Consultada March 27, 2023, URL: https://flutter.dev/

Flutter Community. (2021, Julio 13). *get_it | Dart Package*. Pub.dev. Consultada February 9, 2023, URL: https://pub.dev/packages/get_it

Jourdan, M. (2022, August 10). *StreamCompanion - Apps en Google Play*. Google Play. Consultada February 1, 2023, URL: https://play.google.com/store/apps/details?id=com.streamhelper.streamhelper

Kounex. (2023, Enero 20). *OBS Blade*. OBS Blade | Home. Consultada February 1, 2023, URL: https://obs-blade.kounex.com

Logitech Services S.A. (2023, March 17). *Software gratuito de streaming en tiempo real y grabación*. Streamlabs. Consultada March 17, 2023, URL: https://streamlabs.com/es-es

Martin, R. C. (2012, August 13). *Clean Coder Blog*. Clean Coder Blog. Consultada March 27, 2023, URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Meta Platforms, Inc. (2023, March 27). *React Native*. React Native. Learn once, write anywhere. https://reactnative.dev/

Moobot. (2023). *Moobot*. Moobot, your Twitch bot for 2023 [FREE]. Consultada March 17, 2023, URL: https://moo.bot

NightDev, LLC. (2023, 03 17). *Nightbot*. Nightbot. Consultada 03 17, 2023, URL: https://nightbot.tv

Oikarinen, J. (1993, Mayo). *RFC 1459: Internet Relay Chat Protocol*. RFC Editor. Consultada June 1, 2023, URL: https://www.rfc-editor.org/rfc/rfc1459

Open Broadcaster Software. (2023, Enero 7). *OBS Studio* (29). OBS Studio. Consultada Febrero 1, 2023, URL: https://obsproject.com/es

Parecki, A. (2023, March 27). *OAuth 2.0 — OAuth*. OAuth. Consultada March 27, 2023, URL: https://oauth.net/2/

Sánchez, J. (2020, Marzo 26). *Introducción al patrón BLoC*. XurxoDev. Consultada February 9, 2023, URL: https://xurxodev.com/introduccion-al-patron-bloc/

Touch Portal. (2023, February 1). Touch Portal - Remote macro control deck for PC and Mac OS for streamers, content creators and other professionals. Consultada February 1, 2023, URL: https://www.touch-portal.com

Twitch Interactive. (2023, 03 27). *Twitch*. URL: https://www.twitch.tv/

Twitch Interactive, Inc. (2023, February 9). *Authentication*. Twitch Developers. Consultada February 9, 2023, URL: https://dev.twitch.tv/docs/authentication/

Twitch Interactive, Inc. (2023, Enero 23). *Twitch - Apps en Google Play*. Google Play. Consultada February 1, 2023, URL: https://play.google.com/store/apps/details?id=tv.twitch.android.app

Twitch Statistics 2023 - How many People use Twitch? (2023, March 9). Abdalslam. Consultada March 27, 2023, URL: https://abdalslam.com/twitch-statistics#google_vignette

Wikipedia. (2023, February 9). *Inyección de dependencias*. Wikipedia. Consultada February 9, 2023, URL: https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias

Wikipedia. (2023, February 9). *Observer (patrón de diseño)*. Wikipedia. Consultada February 9, 2023, URL: https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)

Xamarin.Forms Documentación - Xamarin. (2023, March 27). Microsoft Learn. Consultada March 27, 2023, URL: https://learn.microsoft.com/es-es/xamarin/xamarin-forms/

Scrum: conceptos clave y cómo se aplica en la gestión de proyectos [2023] • Asana. (2023, June 19). Asana. Retrieved July 15, 2023, URL: https://asana.com/es/resources/what-is-scrum