

# **Taller de Programación II - 75.52**

## **Chotuve**

Aplicación Android

# Índice

<b>1. Manual de Usuario</b>	<b>2</b>
1.1. Login	2
1.2. Registro	3
1.3. Recupero de contraseña	3
1.4. Pantallas principales	4
1.4.1. Home	4
1.4.2. Subir Video	5
1.4.3. Perfil propio	6
1.4.4. Lista de amigos	7
1.4.5. Notificaciones	7
1.5. Pantallas secundarias	8
1.5.1. Vista de video	8
1.5.2. Preferencias de usuario	9
1.5.3. Chat	10
1.5.4. Búsqueda de usuarios	10
1.5.5. Edición de video	11
1.5.6. Perfil de otro usuario	11
1.6. Códigos de error	12
<b>2. Arquitectura</b>	<b>13</b>
2.1. Navegación	13
2.1.1. Bottom Tabs	13
2.1.2. Stack	13
2.1.3. Jerarquía de pantallas	14
2.2. Flujo de autenticación	14
2.2.1. Pantallas	15
2.2.2. Google y Facebook	15
2.2.3. Jerarquía de componentes incluyendo Login	16
2.3. Estado	16
2.4. Comunicación con servidores	16
2.5. Colores y estilos	17
2.6. Notificaciones y Chat	17
<b>3. Detalles de implementación</b>	<b>17</b>
3.1. Cache	17
3.2. Previsualización de videos	18

## 1. Manual de Usuario

### 1.1. Login

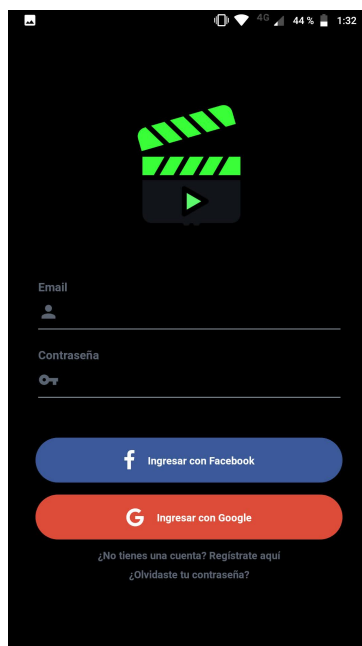


Figura 1: Pantalla principal de Log-In

La primera pantalla que se va a encontrar el usuario al ingresar en la aplicación es la pantalla de login. Desde la misma tenemos diferentes opciones para autenticarnos, la estándar es mediante un e-mail y una contraseña, los cuales debemos introducir en los campos que se ven en pantalla. Luego de introducir la contraseña y presionar la tecla Enter, se realizará el ingreso automáticamente, debido a eso no es necesario un botón específicamente para esa tarea.

También tenemos la posibilidad de ingresar una cuenta de Google o de Facebook, para eso simplemente se debe presionar el botón correspondiente y eso nos llevará a el ingreso de cada aplicación en caso de ser necesario.

Para registrar un nuevo usuario o recuperar la contraseña de nuestra cuenta se debe presionar el texto correspondiente que se encuentra debajo, lo cual nos llevará a la pantalla donde podremos realizar esas acciones.

## 1.2. Registro

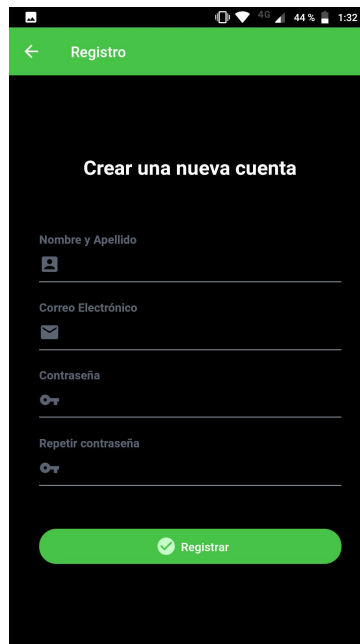
The screenshot shows a mobile application interface for user registration. At the top, there is a green header bar with a back arrow and the word "Registro". Below the header, the title "Crear una nueva cuenta" is centered. The form consists of four input fields: "Nombre y Apellido" with a person icon, "Correo Electrónico" with an envelope icon, "Contraseña" with a key icon, and "Repetir contraseña" with a key icon. At the bottom, there is a green button with a checkmark and the text "Registrar". The status bar at the top shows 4G, 44% battery, and 1:32.

Figura 2: Formulario de registro de un usuario nuevo

Para registrar un nuevo usuario simplemente debemos llenar el formulario con los datos que se nos solicitan, la única restricción es que la contraseña debe tener como mínimo seis caracteres, y el e-mail debe ser válido. Una vez que presionemos el botón que se encuentra en la parte inferior, se intentará registrar el usuario y en caso de éxito se ingresará a la aplicación, no es necesario verificar la dirección de e-mail.

## 1.3. Recupero de contraseña

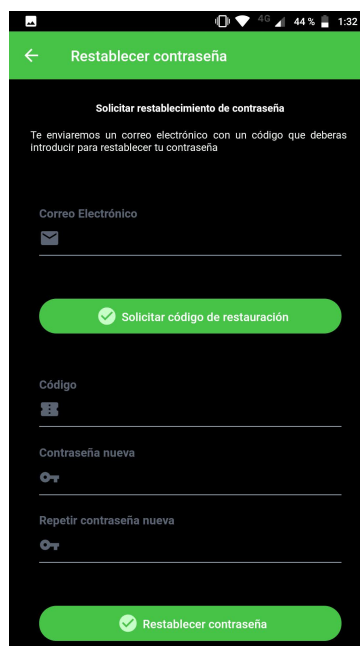
The screenshot shows a mobile application interface for password recovery. At the top, there is a green header bar with a back arrow and the text "Restablecer contraseña". Below the header, the title "Solicitar restablecimiento de contraseña" is centered. A message states: "Te enviaremos un correo electrónico con un código que deberas introducir para restablecer tu contraseña". Below this, there is an input field for "Correo Electrónico" with an envelope icon. A green button with a checkmark and the text "Solicitar código de restauración" is positioned below the email field. Further down, there are three more input fields: "Código" with a person icon, "Contraseña nueva" with a key icon, and "Repetir contraseña nueva" with a key icon. At the bottom, there is a green button with a checkmark and the text "Restablecer contraseña". The status bar at the top shows 4G, 44% battery, and 1:32.

Figura 3: Formulario de recupero de contraseña

El proceso de recuperación de contraseña consiste en dos pasos, primero debemos introducir la dirección de e-mail correspondiente a la cuenta que queremos recuperar. Al presionar el primer botón, en caso de que exista la cuenta, se enviará un e-mail el cual contiene un código, debemos introducir el mismo en el formulario inferior junto con la nueva contraseña. Al presionar el segundo botón se realizará el cambio de contraseña, luego de eso regresaremos a la pantalla de Log-In donde podremos ingresar con la cuenta.

## 1.4. Pantallas principales

Lo primero que vamos a encontrarnos luego de ingresar con un usuario es con la pantalla principal, la cual contiene cinco pestañas, las cuales detallamos a continuación:

### 1.4.1. Home



Figura 4: Pantalla principal

La pantalla principal de la aplicación, en la misma se le presenta al usuario la lista de videos que estan disponibles para ver en la aplicación. Los videos se presentan en forma de una imagen miniatura acompañado por el título, el autor y la fecha de subida. Además, si el usuario se mantiene por cierto tiempo mirando una miniatura, se comienza a reproducir una previsualización del video, aunque con el audio silenciado para que no resulte invasivo. Cuando el usuario desee mirar el video completo puede presionar la imagen para navegar hacia la pantalla de video, la cual explicaremos más adelante.

### 1.4.2. Subir Video

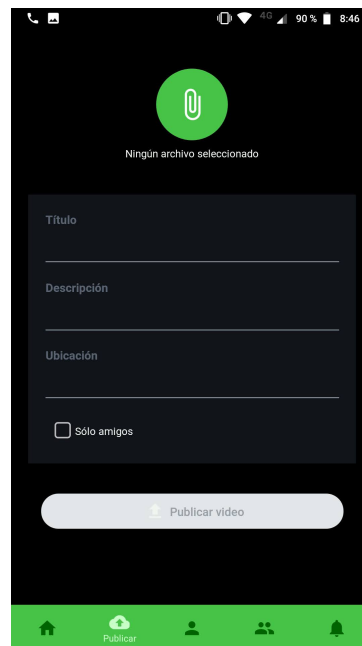
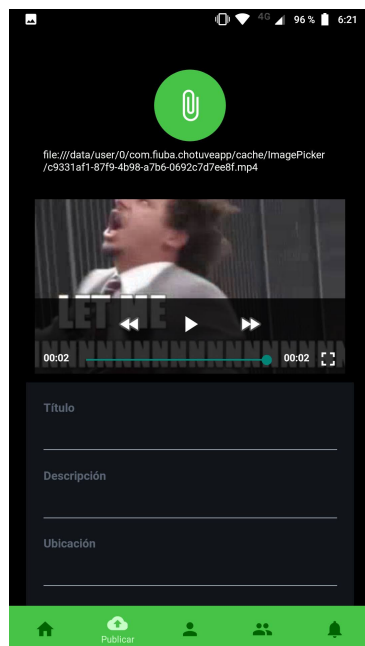


Figura 5: Subir video

En esta pantalla se presenta la interfaz a través de la cual el usuario puede subir un video a la plataforma. Contiene un botón principal el cual al ser presionado se lanza una pantalla de selección de archivos. Una vez que se selecciona el mismo aparecerá una previsualización del video con la cual el usuario puede verificar el video antes de subirlo, luego se deben rellenar los campos que se encuentran debajo para que se habilite el botón inferior. Al presionar este botón comenzará la subida del video y se mostrará el porcentaje del progreso de subida.



(a) Visualización del video luego de seleccionarlo



(b) Porcentaje de subida luego de comenzar a subir el video

### 1.4.3. Perfil propio

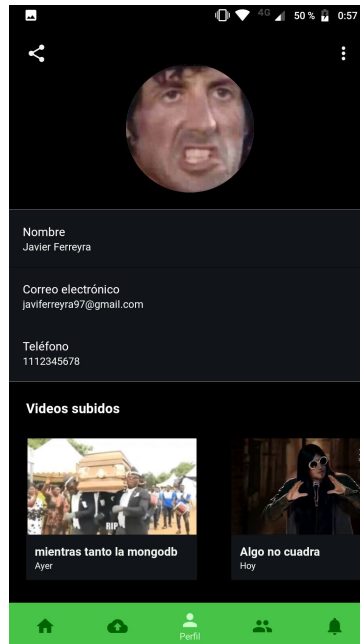


Figura 6: Perfil del usuario

La pantalla del perfil del usuario contiene toda la información del usuario local. En la parte superior se muestra la foto de perfil la cual al presionarla se lanza una pantalla de selección de archivos donde podemos elegir una nueva imagen de perfil para nuestro usuario. Luego se muestra la información y por último los videos subidos.

También se puede presionar el botón de la esquina superior izquierda el cual nos permite compartir externamente un link al perfil, el cual al ser abierto se abrirá la aplicación y llevará al perfil.

Si presionamos el botón de menú que se encuentra en la esquina superior derecha podremos ver un menú con las opciones de ir a la pantalla de preferencias o salir (log-out).

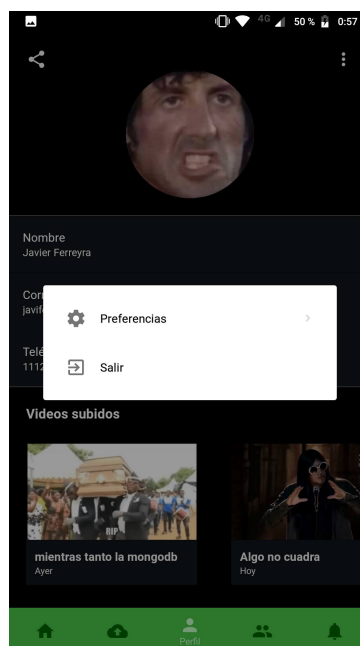


Figura 7: Opciones dentro del perfil de usuario propio

#### 1.4.4. Lista de amigos

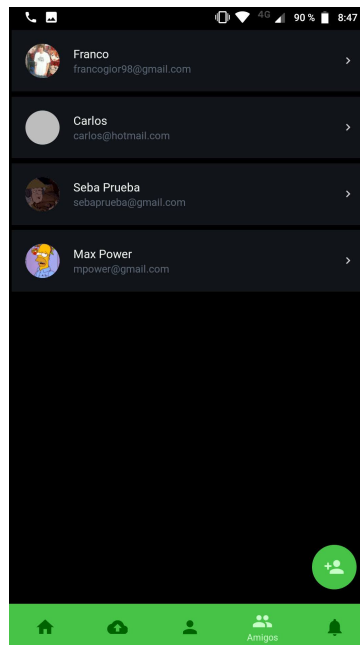


Figura 8: Lista de amigos

En esta pantalla se muestra la lista de amigos, si presionamos en uno de ellos se navega hacia la pantalla de chat, la cual vamos a explicar más adelante. En la esquina inferior derecha tenemos un botón con el cual navegamos hacia la pantalla de búsqueda de usuarios, también explicada más adelante.

#### 1.4.5. Notificaciones

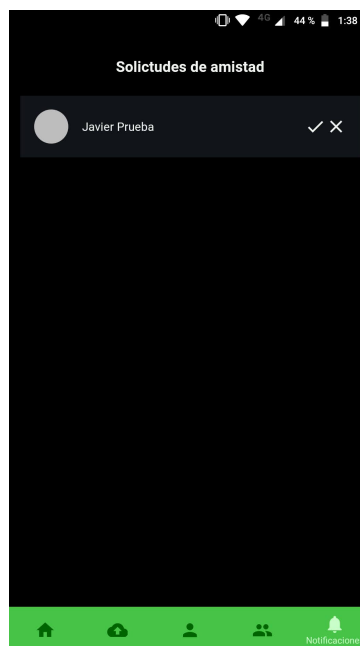


Figura 9: Pantalla de notificaciones

En esta pantalla nos saldrán los avisos que requieran una acción del usuario, por ahora la única acción disponible es aceptar solicitudes de amistad, pero existe la posibilidad de agregar notificaciones



de otros tipos en el futuro.

## 1.5. Pantallas secundarias

Luego tenemos diferentes sub-pantallas a las cuales podemos ingresar desde las anteriores, las mismas se explican a continuación:

### 1.5.1. Vista de video



Figura 10: Visualización de video

A esta pantalla el usuario accede desde el Home al presionar un video. Contiene la visualización del video, la cual se mantiene fija y por debajo se encuentra la información del video y los comentarios. Todo lo que se encuentra debajo del video se encuentra dentro de un scroll por lo que en caso de que los comentarios ocupen más que lo que entra en la pantalla se puede deslizar, ocultando la información del video y la caja de comentarios.

Desde aquí podemos acceder al perfil de un usuario presionando su nombre, ya sea desde la lista de comentarios o desde la caja de información del video.

Además, tenemos un botón en la esquina superior derecha con el cual podremos compartir un link al video en alguna aplicación externa.

En caso de que el video sea propio, en la figura 11 podemos ver un segundo botón en la esquina superior derecha, con el cual se puede acceder a un menú desde el cual se puede editar las preferencias del video o bien eliminarlo.

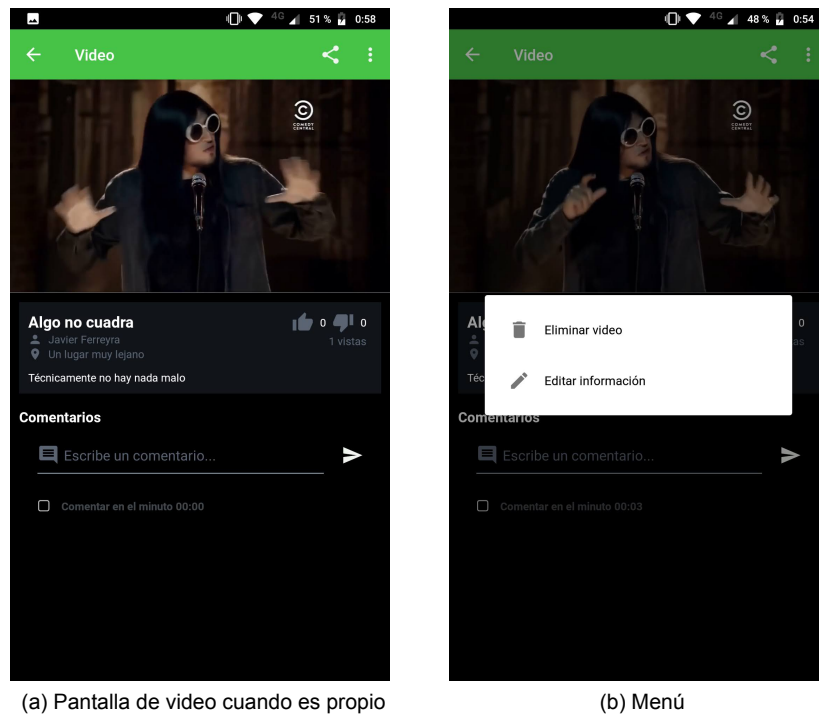


Figura 11: Vista cuando el video es propio

### 1.5.2. Preferencias de usuario

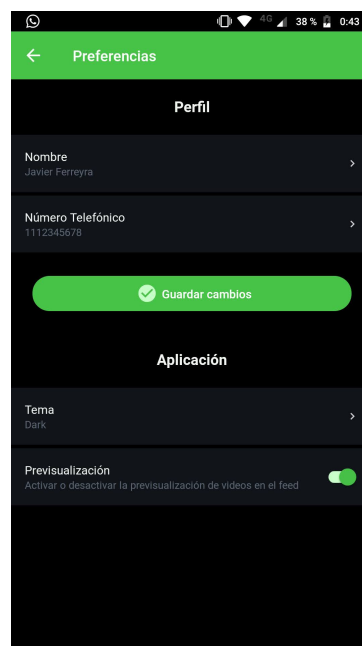


Figura 12: Pantalla de preferencias

A esta pantalla se accede desde el menú del perfil propio. Desde esta pantalla el usuario puede editar su perfil y otras preferencias de la aplicación.

### 1.5.3. Chat

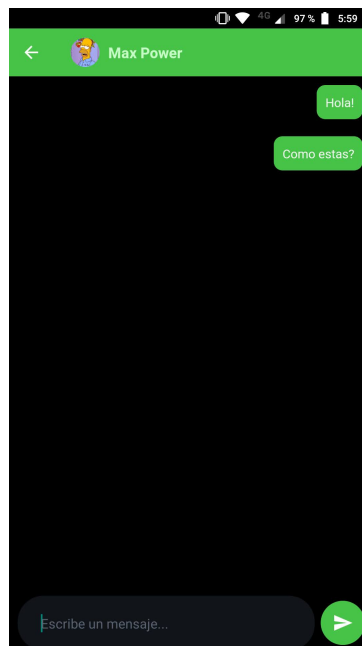


Figura 13: Pantalla de chat

A esta pantalla se accede desde la lista de amigos, desde aquí el usuario puede intercambiar mensajes con otro usuario, la interfaz es similar a otras aplicaciones de mensajería. Si se presiona el nombre de usuario en la parte superior se navegará al perfil del mismo.

### 1.5.4. Búsqueda de usuarios

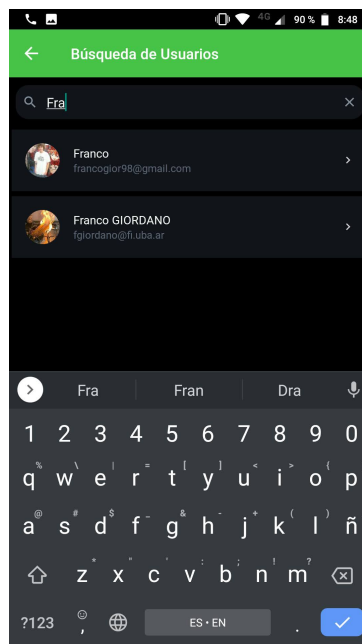


Figura 14: Pantalla de búsqueda

Aquí se accede desde la lista de amigos presionando el boton inferior. Para buscar un usuario se debe comenzar a escribir un nombre en la barra de búsqueda y la misma se realizará de forma

automática. Luego cuando se presente la lista de resultados, se podrá presionar en uno de ellos para navegar hacia el perfil del usuario.

#### 1.5.5. Edición de video

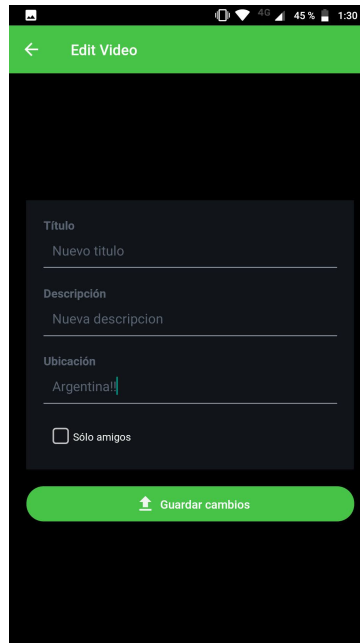
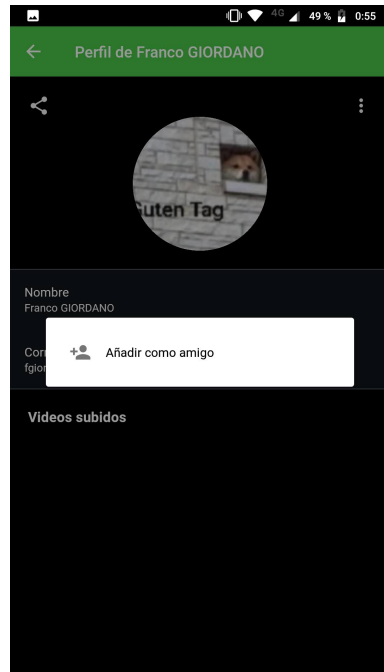


Figura 15: Pantalla para editar la información del video

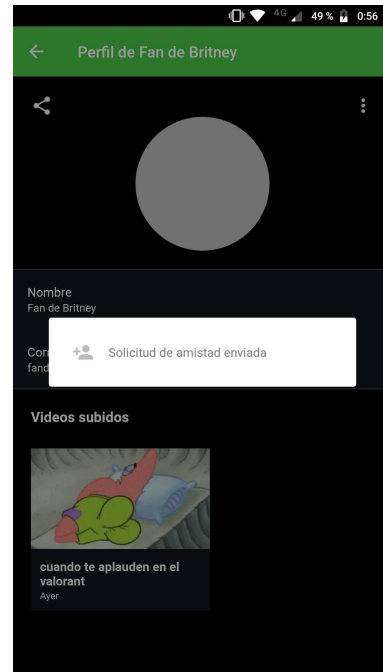
Aquí accederá el usuario desde el menú que se encuentra en los videos propios, en esta pantalla podrá editar la información del video, al presionar el botón se enviará la información al servidor.

#### 1.5.6. Perfil de otro usuario

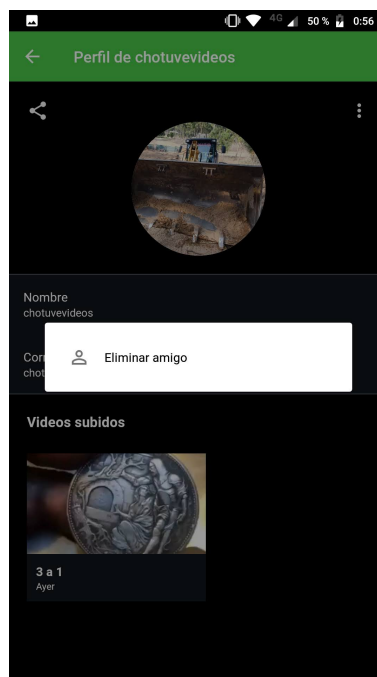
Cuando el usuario se encuentre en el perfil de otro, la interfaz que verá es similar a la del perfil propio, con la diferencia de que el menú tendrá las opciones para añadirlo como amigo, o eliminarlo de su lista de amigos, la opción que se muestra depende del estado actual del vínculo entre ambos.



(a) Añadir amigo



(b) Solicitud ya enviada



(c) Eliminar amigo

## 1.6. Códigos de error

Ante un error al realizar alguna acción se muestra un mensaje del tipo *Toast*, el cual indica qué acción originó el error junto a un código numérico. Los códigos en su mayoría corresponden a códigos HTTP, aunque se agregaron otros para indicar errores internos de la aplicación, a continuación se detallan las posibles causas de error.

- **404:** Se está intentando acceder a un recurso que no se encontró en el servidor, la causa mas común de este error es que se intente acceder a un video que fue eliminado, esto es posible si otro usuario elimina un video después de que obtuvimos la lista y no actualizamos.

- **400:** Este error puede surgir porque se está haciendo un llamado a la API de forma incorrecta, puede deberse a que hubo un cambio en la API y el cliente no se actualizó. Actualizar el cliente a la última versión podría solucionar este error.
- **401:** Se está intentando acceder a un recurso para el cual no se tiene permiso, este error no debería salir en la aplicación ya que la misma no tiene la funcionalidad para acceder a recursos de administrador y estos son solo accesibles desde la web.
- **500 y 502:** Errores internos del servidor, de encontrarse un error de este tipo debe informarse a los administradores.
- **503:** El servicio no está disponible, este error surge frecuentemente debido a que Heroku apaga los servidores si no recibe pedidos en un período de tiempo. Para solucionar este error simplemente se debe esperar a que los servidores se inicien y reintentar la acción.
- **1000:** Error de autenticación. Este error puede surgir porque no se pudo obtener un token de autenticación de Firebase. Para solucionarlo se debe salir de la cuenta de usuario y volver a loguearse.
- **1001:** Error de fetch. Este error surge normalmente cuando no se tiene conexión a internet.

## 2. Arquitectura

### 2.1. Navegación

La navegación entre las distintas pantallas se realizó mediante la combinación de dos patrones conocidos los cuales son Stack y Bottom Tabs, los cuales se explican a continuación.

#### 2.1.1. Bottom Tabs

Este patrón de navegación consiste en que las distintas pantallas se presentan como un conjunto de pestañas, y para navegar a través de ellas se presenta una barra situada en la parte inferior. Este patrón permite darle al usuario un panorama general desde el comienzo y que conozca en solo un vistazo las acciones principales que puede realizar en la aplicación. Las pantallas que se presentan de esta manera son cinco:

- Home
- Subir video
- Perfil
- Amigos
- Notificaciones

Para la implementación del mismo simplemente se utilizó el componente `MaterialBottomTabNavigator`, el mismo es una variante del original `BottomTabNavigator`, pero que presenta una interfaz mucho más acorde a un sistema Android, ya que utiliza el diseño *Material*, característico del mismo.

#### 2.1.2. Stack

La navegación por stack consiste en que las pantallas a las que vamos accediendo sucesivamente se renderizan por encima de la anterior, es decir que se van apilando, de esta manera si el usuario quiere regresar hacia atrás simplemente se elimina la pantalla superior y automáticamente se encuentra en la pantalla anterior.

Este patrón se utilizó en la presente aplicación para todas las pantallas secundarias a las cuales se accede a través de las principales que se encuentran en forma de pestañas, mencionadas anteriormente.

Para evitar ciclos en la pila, y que no sea posible navegar infinitamente entre pantallas llenando la memoria, las pantallas del mismo tipo no pueden estar en el stack al mismo tiempo. Un ejemplo de

esto es cuando un usuario ingresa a un video a través del perfil del usuario. Luego desde la pantalla de video puede ingresar nuevamente a un perfil del mismo usuario u otro, y desde el perfil nuevamente a un video, generando un ciclo. Con la implementación utilizada esto no es posible, ya que si una de estas pantallas ya se encuentra en el stack, en lugar de apilar una nueva pantalla, se desapilarán las necesarias para llegar a la deseada.

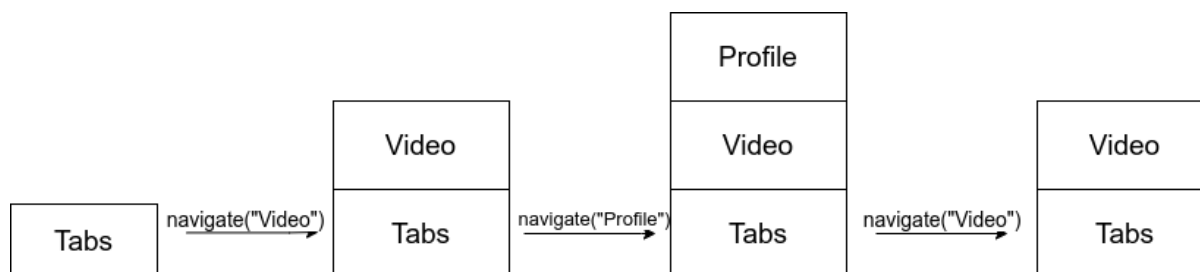


Figura 16: Stack de pantallas al intentar realizar un ciclo

Este comportamiento de las pantallas del stack se realiza de manera automática gracias a la librería React Navigation, ya que se cuenta con dos métodos posibles para navegar, uno es `push()` el cual siempre apila una nueva pantalla, pero si utilizamos `navigate()`, el stack se maneja de manera inteligente buscando la pantalla entre las ya presentes, en lugar de apilar una nueva.

### 2.1.3. Jerarquía de pantallas

La jerarquía de componentes que nos queda es la siguiente:

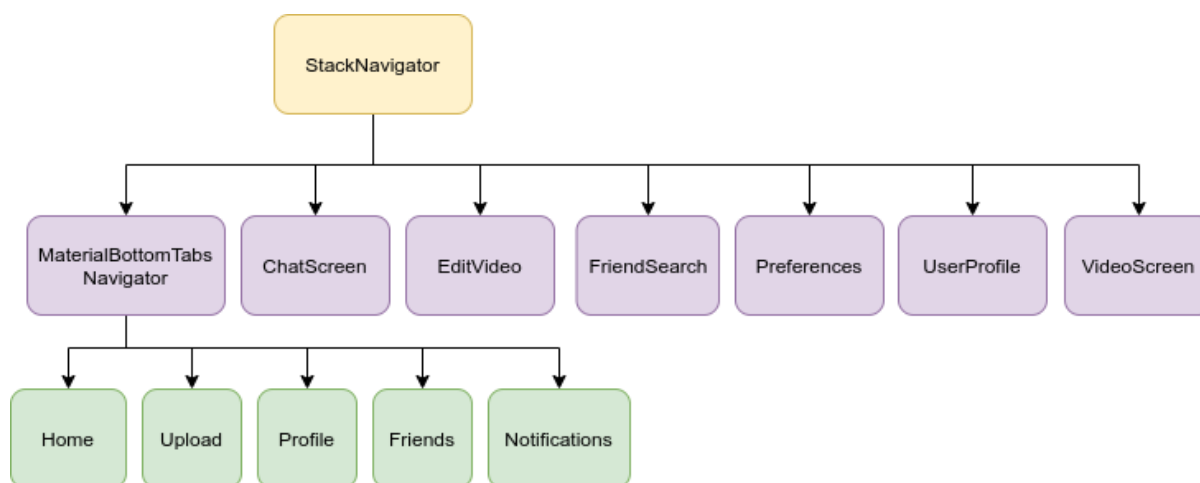


Figura 17: Jerarquía de componentes de las pantallas de la aplicación

## 2.2. Flujo de autenticación

Para la autenticación del usuario se utilizó la plataforma Firebase de Google. El proceso funciona de la siguiente manera:

1. El usuario ingresa usando o bien usuario y contraseña, o bien Google o Facebook.
2. Se envía a Firebase las credenciales <sup>1</sup>
3. Firebase responde con toda la información del usuario, incluido token, o si es un usuario nuevo o fue un log-in, todo esto se guarda.
4. En caso de usuario nuevo se envía cierta información al AppServer para que se lo registre.

<sup>1</sup>Se envía usuario y contraseña, o bien las credenciales que devuelve la librería de Expo al realizar login con Google o Facebook.

5. Si todo es válido se actualiza el estado global de la aplicación con el usuario logueado.

Para evitar el acceso del usuario a la aplicación sin autenticación se utiliza el mecanismo de React. El componente principal de la aplicación, al momento de renderizarse, utiliza el estado global mencionado antes para saber si hay un usuario ingresado o no. En caso de que no lo haya se renderiza el sistema de pantallas de login, explicado más adelante. Si lo hay se renderiza el Stack principal de la aplicación explicado en la sección anterior.

Cuando un usuario ingresa, al actualizarse el estado global, se produce un re-renderizado del componente principal, ese re-renderizado se produce ya que el componente consume al estado global, entonces React lo renderiza nuevamente.

### 2.2.1. Pantallas

Las pantallas de login se manejan también con un patrón Stack, en este caso se cuenta solamente con tres pantallas:

- Log-in
- Registro
- Recuperación de contraseña

### 2.2.2. Google y Facebook

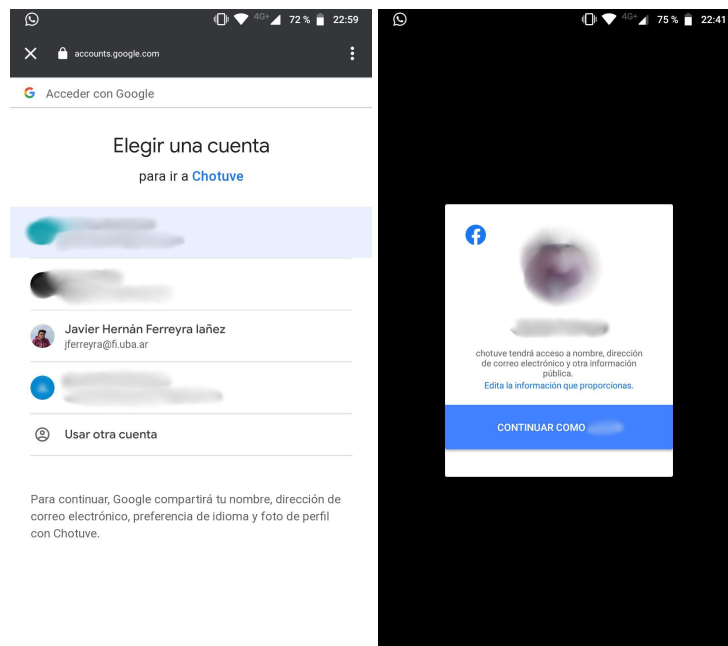


Figura 18: Pantalla de login con Google y con Facebook

Para el ingreso con Google o con Facebook, se llama a un método de la librería de Expo que muestra la pantalla correspondiente. Cuando el usuario ingresa a su cuenta, el método devuelve las credenciales de la cuenta que la aplicación debe reenviar a Firebase, que luego responde con la información de la cuenta.



### 2.2.3. Jerarquía de componentes incluyendo Login

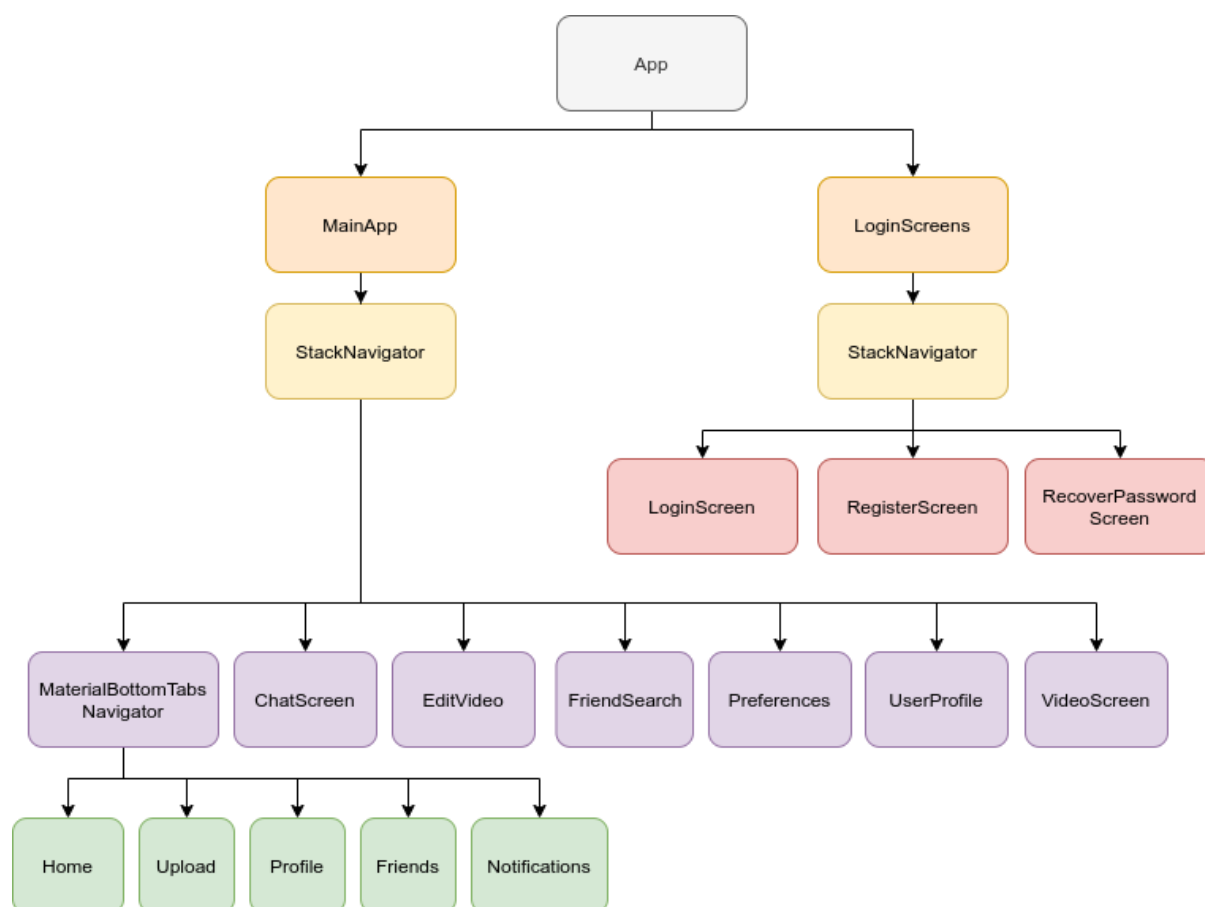


Figura 19: Jerarquía completa: Si el estado global indica que el usuario está autenticado, App renderiza MainApp, sino renderiza LoginScreens

## 2.3. Estado

Para el manejo del estado global de la aplicación se utilizó la API de hooks de React. Los hooks `useContext` y `useState` permiten esto. Se define un componente proveedor que contiene el estado global mediante `useState`, y el *contexto* instanciado con `createContext`. Luego se encierra el componente principal de la aplicación en este componente. Luego, todos los componentes hijos que deseen suscribirse al estado global, importan el *contexto* y utilizan el hook `useContext(contexto)` para tener acceso y re-renderizarse cuando se actualice el mismo.

## 2.4. Comunicación con servidores

La comunicación con los servidores tanto de Firebase como el AppServer se implementó en la clase `ServerProxy`. Esta clase contiene métodos que se corresponden con los endpoints de la API del servidor, así como también los métodos de Firebase que se utilizan para los servicios que utilizamos en la app, los cuales son:

- Login
- Storage

Se guarda una instancia de esta clase en el estado global de la aplicación, entonces cuando un componente necesita comunicarse con el servidor simplemente se suscribe para tener acceso y la utiliza para obtener la información necesaria.

## 2.5. Colores y estilos

Para tener un estilo y colores consistentes a lo largo de la app, también se utilizó un estado global de la forma descrita anteriormente, de esta forma se pueden reutilizar los mismos estilos y colores. Se eligió un estado global porque de esta manera resulta más sencillo implementar la función de cambiar el tema de colores, ya que así se re-renderizan los componentes con el nuevo color instantáneamente al cambiar la opción en la pantalla de preferencias.

## 2.6. Notificaciones y Chat

Las notificaciones se implementaron utilizando la librería *Notifications* de Expo. Al iniciarse la aplicación, al momento de renderizarse el componente principal luego del log-in (es decir, MainApp) la aplicación utiliza la librería para obtener un Token de notificaciones, este token identifica al dispositivo en el servidor de Expo, entonces debe ser enviado al AppServer para que lo guarde y lo utilice para enviar notificaciones cuando sea necesario.

La librería *Notifications* también provee un método para suscribirse a las notificaciones y definir un Callback, el cual será llamado en el momento en que ocurra algún evento relacionado a las notificaciones, esto puede ser tanto cuando llega una notificación, como también cuando el usuario la toca en la barra de notificaciones. Cuando el AppServer envía una notificación, ésta contiene información sobre el tipo de notificación que es, con esta información, en el Callback se define que hacer cuando el usuario toca la notificación:

- Cuando es un mensaje de Chat, navegar hacia el chat con ese usuario.
- Cuando es una solicitud de amistad, navegar hacia la pestaña de Notificaciones, desde donde se pueden aceptar o rechazar las solicitudes.
- Cuando es un comentario en un video, navegar hacia la pantalla de ese video.

Los mensajes de chat también funcionan con este mecanismo. Cuando un usuario ingresa a un chat, se define un nuevo callback utilizando el mismo método de la librería mencionado antes. Este callback atrapa las notificaciones cuando llegan y si se trata de un mensaje del usuario del chat actual, se descarta la notificación y se muestra el nuevo mensaje en tiempo real, caso contrario se muestra la notificación normalmente en la barra. Cuando el usuario abandona la pantalla de chat, desuscribe a este callback de las notificaciones.

# 3. Detalles de implementación

## 3.1. Cache

Para agilizar la carga y evitar el fetch de los mismos datos del servidor de forma repetida se implementó una cache de datos en la clase ServerProxy. Esta caché es simplemente un conjunto de diccionarios, cada diccionario corresponde a una clase de información que se puede traer del servidor (por ejemplo, cache de datos de usuarios, o cache de datos de videos), aunque no hay caché para todos los tipos posibles ya que no es necesario. Solamente se consideraron tipos que no suelen cambiar frecuentemente.

En este caso, las caches se implementaron de forma que no sean utilizadas siempre, y que se permita en ciertos casos que los datos se descarguen del servidor ignorando si los mismos se encuentran en la caché local o no. Se hizo de esa forma para mejorar la experiencia de usuario, ya que en ciertos casos el usuario desea explícitamente que los datos sean descargados desde el servidor. Por eso el uso de la caché se implementó siguiendo estas reglas:

- **Los datos se buscan en cache si los pide algún componente de manera automática**, esto es, cuando un componente se renderiza y lo que muestra necesita obtenerlo del servidor. Un ejemplo de esto son los comentarios en los videos: cada comentario se muestra en un componente separado, este componente la única información que tiene es el ID del usuario y el texto del comentario, por lo que para mostrar el Username debe hacer un fetch. Puede ocurrir que un usuario haya hecho varios comentarios, por lo que cada componente estaría haciendo un fetch del mismo dato de manera automática. Para evitar esto, se usa la cache, entonces si un comentario

anterior ya pidió el nombre de algún usuario, esto se guarda en la cache, y el próximo comentario utiliza ese dato sin traerlo del servidor.

- **Se ignora la cache si el usuario actualiza explícitamente una pantalla**, esto es, arrastrando con el dedo desde la parte superior de la pantalla. Cuando el usuario hace esto, se considera que lo que espera es que los datos se descarguen del servidor, entonces, siempre que el usuario actualice manualmente la pantalla las caches se van a ignorar y se van a pedir los datos al servidor.

Los tipos de dato que se decidieron cachear son los siguientes, ya que se consideró como hipótesis que son datos que no cambian frecuentemente:

- **Datos de usuario:** La clave del diccionario son los UUIDs de los usuarios, contiene toda la información del usuario que provee el servidor.
- **Videos:** Las claves del diccionario son los ID de los videos, contiene toda la información del video a excepción de los comentarios.
- **URLs:** El servidor no guarda URLs directas a los videos y a las imágenes, sino que guarda la ruta del archivo en el Storage de Firebase, por esa razón el cliente debe pedirle a Firebase la URL del archivo en cuestión de manera separada, por lo tanto existe también una cache para estas direcciones, la clave del diccionario es la ruta del archivo.

### 3.2. Previsualización de videos

El mecanismo que se implementó para la previsualización de videos es el siguiente: Se tiene un estado en el componente principal del feed de videos (Home), el cual indica el ID del video cuya miniatura se encuentra actualmente en la pantalla (si hay más de uno se utiliza el primero). Luego este estado se pasa como prop a todos los componentes que corresponden a la miniatura de cada video. En caso de que el ID del video coincida con el de la miniatura, se renderiza una preview en forma de video en lugar de una imagen.

Este mecanismo es posible gracias al componente FlatList de React Native. Este componente es una lista scrolleable, la cual se utiliza para mostrar la lista de videos. Provee un prop a través del cual se le puede pasar un callback que será invocado cuando cambien los componentes que se están mostrando actualmente en pantalla. Este callback al ser invocado, recibe una sublista de la lista original, que contiene solo los elementos que son visibles en ese momento, entonces se obtiene el ID del primer video de la lista y se setea el estado mencionado anteriormente.