UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

Año 2020 - 1er Cuatrimestre

Taller de Programación II (75.52)

ChoTuve

ClienteMobile

Documento de arquitectura / diseño

Fecha de entrega: 30/07/2020

Grupo 8

79979 – Gonzalez, Juan Manuel (juanmg0511@gmail.com)
82449 – Laghi, Guido (guido321@gmail.com)
86429 – Casal, Romina (casal.romina@gmail.com)
96453 – Ripari, Sebastian (sebastiandripari@gmail.com)
97839 – Daneri, Alejandro (alejandrodaneri07@gmail.com)

Contenido

1. Objetivo

El objetivo de este documento es presentar la arquitectura utilizada, así como también comentar los aspectos de diseño más importantes del **ClienteMobile** utilizado en la aplicación *ChoTuve*.

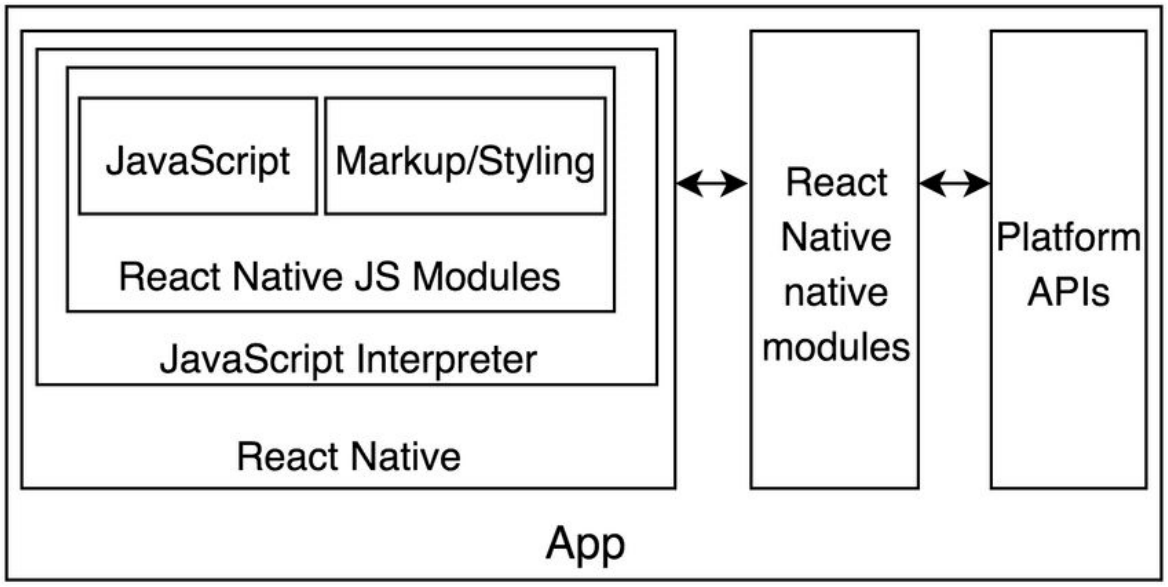
2. Arquitectura

En esta sección se describe la arquitectura del **ClienteMobile**.

2.1 Consideraciones generales

Para la implementación del **ClienteMobile** se decidió utilizar React Native. Basamos nuestra elección teniendo en consideración la experiencia previa de los miembros del grupo con React, y la posibilidad de poder entregar una solución que soporte tanto Android como iOS.

A grandes rasgos, la arquitectura del ClienteMobile puede resumirse mediante el siguiente esquema:



2.1.1 Ambiente de desarrollo

El ambiente de desarrollo se configuró de manera tradicional en las máquinas de cada desarrollador. Esto se realizó siguiendo las instrucciones provistas en:

https://reactnative.dev/docs/environment-setup

Se utilizaron las siguientes combinaciones de sistema operativo/dispositivos:

OS	Target
Linux	Android
macOS	iOS, Android

2.2.2 Soporte multiplataforma

Luego del tercer checkpoint, se acordó con el docente asignado al grupo un cambio de alcance al enunciado original: teniendo como objetivo generar una App de mayor calidad final y más terminada en lo que refiere a los detalles, se propuso no implementar la funcionalidad de *chat* pedida en el enunciado, visto el tiempo de proyecto restante, para concentrar el esfuerzo en pulir la funcionalidad ya implementada.

Asimismo, también se propuso brindar soporte para dispositivos iOS, algo no contemplado en los requerimientos, pero que a criterio del grupo valía la pena implementar, dado el uso de React Native como tecnología para el **ClienteMobile**.

Cabe aclarar que para poder lograr esto último se necesita de cierto esfuerzo, ya que hay que adaptar partes de código para que la solución funcione adecuadamente en iOS (sign in with Google, permisos, push notifications).

2.2.3 Testing

El esfuerzo de testing del **ClienteMobile** se distribuyó a medida que avanzaba su desarrollo, que fue prácticamente continuo durante el tiempo que duró el trabajo práctico.

Dado que no existían requerimientos de testing para este componente de la solución, se decidió optar por una estrategia que fuera efectiva pero no que comprometiese las horas disponibles de los integrantes del grupo:

- De esta manera, durante todo el proceso se realizaron pruebas unitarias en los ambientes locales, con metodología de caja blanca.
- Un tiempo prudencial antes de cada checkpoint (variable, por lo general unos 2 días), se generaba un build de la aplicación, en forma de APK, para instalar en los equipos físicos de los integrantes que no estuviesen afectados al desarrollo del **ClienteMobile**.
- Cada integrante luego realizaba pruebas sobre el total la aplicación, con metodología de caja negra. Por restricciones de tiempo no se utilizaron pruebas de especificación (casos de test) durante estos casos.

La coordinación de las pruebas se llevó a cabo mediante una planilla de cálculo, donde se registró en forma de tabla los errores o defectos encontrados más información adicional para su seguimiento.

Como se decidió dar soporte a iOS luego del tercer checkpoint, solo se realizaron pruebas unitarias locales con esta versión por restricciones de tiempo.

Para más información sobre el esfuerzo de testing completo, se puede consultar el manual de proyecto.

2.2.4 Estructura del proyecto

El proyecto se encuentra hosteado en GitHub. Se cuenta con dos *branches*, una para staging (develop) y otra para producción (master). Durante la fase de desarrollo, se *pushean* los updates al branch de desarrollo, y luego de cada checkpoint se actualiza master con la entrega realizada.

Los cambios menores propuestos por algún integrante del equipo, producto del uso de la API o testing es codificada en un *branch* nuevo e integrada a develop mediante *pull requests*.

La home del repositorio es:

https://github.com/encubos/fiuba-taller-2-cliente-mobile

Su estructura es:

```
.
├── __tests__
├── android
├── components
├── images
├── ios
├── models
├── networking
├── node_modules
└── utils

10537 directories, 76342 files
```

3. Diseño

En esta sección se expone el diseño del **ClienteMobile**. Se utilizará una sub-sección para hacer comentarios sobre cada componente relevante de la solución.

3.1 Paquetes utilizados

En la siguiente tabla se presenta una lista de los paquetes más importantes utilizados por el **ClienteMobile**:

Nombre	Versión utilizada	Propósito
react	16.11.0	Framework base.
react-native	0.62.2	Framework base.
react-native-document-picker	3.3.3	Permite elegir archivos en el dispositivo.
react-native-elements	2.0.3	Biblioteca de UI, sólo utilizada para mostrar el avatar del usuario.
react-native-ffmpeg	0.4.4	Utilizado para determinar la longitud de los videos.
react-native-firebase	5.6.0	Disponibiliza los servicios de Firebase utilizado.
react-native-fs	2.16.6	Permite acceder a elementos del sistema de archivos.
react-native-gesture-handler	1.6.1	Manejo de multi-touch.
react-native-google-signin	2.1.1	Disponibiliza el botón de “Sign in with Google” y toda la funcionalidad asociada.
react-native-paper	3.8.0	Biblioteca utilizada para el diseño de UI, por su adhesión a Material Desgin.
react-native-permissions	2.1.5	Permite manejar permisos sobre los dispositivos, cross-platform.

react-native-reanimated	1.8.0	Posibilita animaciones de UI.
react-native-safe-area-context	0.7.3	Brinda configuraciones para las áreas de pantalla reservadas de los dispositivos.
react-native-screens	2.5.0	Provee navigation containers.
react-native-vector-icons	6.7.0	Provee los íconos utilizados en la aplicación.
react-native-video	4.4.5	Permite mostrar los videos en pantalla.
react-native-video-controls	2.6.0	Muestra los controles de reproducción al mostrar un video en pantalla.

3.1.1 Android

Para el caso específico de Android, se cuenta con un proyecto en el directorio ‘android’ del repositorio, donde se pueden configurar los distintos parámetros propios de este sistema. Se utiliza Gradle para las tareas de build y deploy al emulador, mediante el uso de un wrapper. La generación del archivo instalable apk se genera mediante esta tool.

Los íconos de sistema fueron creados mediante la aplicación de macOS Image2Icon.

3.1.2 iOS

A fin de brindar soporte para la plataforma iOS, el proyecto cuenta con un directorio ‘ios’ en el repositorio, donde se encuentra un proyecto de Xcode y todos los archivos de configuración específicos de la plataforma. En particular, se destaca el uso de cocoapods, para manejar las dependencias del proyecto. La generación de los builds se realiza mediante el uso de comandos de react native.

Los íconos de sistema fueron creados mediante la aplicación de macOS Image2Icon.

3.2 Funcionalidad

Se presenta en esta sección los aspectos de funcionalidad más relevantes del **ClienteMobile**, con una sub-sección por cada una de ellas.

3.2.1 UI / UX

Siguiendo las pautas del enunciado, se trabajó con la idea de respetar al máximo los lineamientos de Material Design para la construcción de la UI. En este sentido, se optó por utilizar la biblioteca

react-native-paper, ya que es una biblioteca que sigue dicha premisa. Sólo cuando no se pudo resolver mostrar el avatar y nombre del usuario en forma conveniente, se utilizaron componentes de otra librería, react-native-elements, que posee un componente de avatar mucho más completo y funcional.

Respecto a la UX, dentro de las limitaciones del proyecto, se hizo el ejercicio de ponerse en el lugar del usuario, y realizar algunas pruebas de usabilidad, a fin de lograr la mejor experiencia posible.

El objetivo final fue diseñar una aplicación atractiva, simple e intuitiva de utilizar, que no requiera documentación de usuario.

3.2.2 Componentes

Se presenta a continuación una tabla con los principales componentes desarrollados para el **ClienteMobile**, más un resumen de las funciones que cumplen:

Nombre	Descripción
EditProfileScreen	Permite editar los datos de perfil del usuario logueado, y cambiar su contraseña (si no usa Sign in with Google).
EditVideoScreen	Permite editar la metadata de un video propio.
FriendsScreen	Pantalla que presenta los pedidos de amistad pendientes (y acciones disponibles) y el listado de amigos.
LoginScreen	Permite iniciar sesión en la aplicación, o acceder a las funcionalidades de registro y recupero de contraseña.
MuroScreen	Pantalla principal de la aplicación, muestra una serie de videos (según un orden de importancia definido con un motor de reglas) y provee acceso al resto de las funcionalidades de la aplicación.
PasswordRecoveryScreen	Permite recuperar la contraseña.
ProfileScreen	Pantalla que muestra los datos de perfil, estadísticas

	generales y listado de videos de un usuario.
SignUpScreen	Permite registrarse al sistema, mediante una combinación de usuario y contraseña.
SplashScreen	Pantalla de presentación y carga del sistema.
UploadVideoScreen	Permite subir un video en la aplicación. Contiene adicionalmente la lógica de solicitud de permisos, y manejo de archivos.
VideoScreen	Pantalla que muestra un video, sus datos generales, y una lista de comentarios de usuarios.

3.2.3 Permisos sobre el OS

El *request* de permisos sobre el sistema operativo es uno de los puntos donde debió utilizarse, en primer lugar, una biblioteca cross-platform, así como también código diferenciado por plataforma, por ejemplo para especificar los permisos a solicitar.

Para los permisos sobre videos e imágenes, se utilizó la biblioteca react-native-permissions, mientras que para las notificaciones *push*, esto es resuelto por la biblioteca de FCM en forma directa.

3.2.4 Manejo de archivos físicos

El manejo de archivos físicos fue otro punto donde fue necesario el uso de código diferenciado por plataforma. Si bien se utilizó la biblioteca react-native-fs, que es *cross-platform*, tuvimos que hacer un manejo especial de los path físicos de los archivos, dado que en Android e iOS son reportados de forma distinta.

Para obtener la duración de los videos, se utilizó la biblioteca react-native-ffmpeg. Una vez que es seleccionado el archivo, se procede a obtener su metadata antes de subirlo a Firebase.

Finalmente, sobre la estrategia para subir los videos, se resolvió de la siguiente forma:

- La aplicación inicia la subida del archivo a Firebase directamente.
- La aplicación contacta y postea la metadata del video en el AppServer.
- El AppServer contacta al Media Server y obtiene el path y el thumbnail del video.

3.2.5 Sign in with Google

Esta funcionalidad se resolvió mediante el uso de react-native-google-signin, que simplifica en gran forma el proceso. Este paquete provee el componente que muestra el botón en la pantalla, y toda la funcionalidad asociada. Lo único que debió hacerse fue sacar un key para Android y otro para iOS, e incorporarlos a los archivos de configuración correspondientes en cada proyecto.
Se tomó la decisión de registrar automáticamente a un usuario de Google inicia sesión en la aplicación por primera vez, tomando los datos de su perfil de Google (luego los podrá editar en la pantalla correspondiente).

Para más información sobre el funcionamiento de esta capacidad de la aplicación, se recomienda consultar el manual de Diseño/Arquitectura del AuthServer.

3.2.6 Push notifications

Las notificaciones *push* que recibe el usuario fueron implementando FCM. Esencialmente la aplicación se encarga de obtener un *token* de FCM, y luego lo registra en el App server. Hecho esto, el dispositivo se encuentra habilitado para recibir notificaciones. Las acciones de logout y cierre de cuenta, además de cerrar la sesión, se encargan de des-registrar el token de FCM tanto en el AppServer como en Firebase.

Al recibir una notificación:

- En caso de estar cerrada la aplicación: se muestra la notificación y al interactuar con ella se abre la aplicación.
- En caso de estar abierta y en primer plano la aplicación: se muestra un “Alert” con opción de “OK”.

Eventos que disparan una notificación:

- Solicitud de amistad (al destinatario).
- Aprobación de una solicitud de amistad (al que hizo la solicitud).
- Nuevo comentario en video (al dueño del video).
- Nuevo like (al dueño del video).