Práctica

Visión General

A través de los ejemplos hemos aprendido gran parte de los fundamentos de Android. Es hora de hacer crecer nuestra aplicación de Chat y aplicar tus conocimientos dentro de la plataforma.

Objetivos

1. **Aplicación de los conocimientos de la plataforma:** Uso de los componentes principales de Android

2. **Aplicación de los conocimientos de Kotlin:** Uso de los conceptos de programación orientada a objetos y programación funcional

3. **Diseño de la aplicación:** Seguir las guidelines de Material Design para la construcción de la interfaz

Especificaciones

Lo que haremos será añadir una serie de funcionalidades a la aplicación de Chat, para ello se proveerá una historia de usuario y una propuesta para su desarrollo

1. Manejo de errores

Historia de usuario

En la aplicación sólo se ha tenido en cuenta el Happy-Path, ignorando el caso en el que se produzca un error, ya sea por la conexión de red u otro caso (En nuestra aplicación de Chat estos errores suceden de manera random por medio de los mocks configurados). El usuario debe ser notificado de cualquier tipo de error que suceda en la aplicación que sea relevante para él

Detalles de implementación

-. Los errores deben ser mostrados al usuario sin bloquear su interacción a menos que sea requerida una acción por parte del usuario

2. Loading/Empty states

Historia de usuario

El usuario debe tener constancia en todo momento de qué está pasando en la aplicación y que no sienta que la aplicación se ha quedado colgada si una operación en segundo plano de carga de datos se demora demasiado por diversas causas. Si por algún caso el listado solicitado por el usuario es vacío, el usuario debe visualizar un empty state indicando tal resultado

Detalles de Implementación

-. Durante la carga de datos se deberá mostrar un ProgressBar indicando al usuario que se está realizando una carga de datos.
-. En el caso de no recibir ningún contenido, se deberá mostrar al usuario que no se han recibido ningún dato

3. Pull-To-Refresh

Historia de Usuario

El usuario debe ser capaz de refrescar el listado por medio del gesto Pull-To-Refresh

Detalles de implementación

-. Uso de Swipe-To-Refresh en el RecyclerView (Puedes consultar la documentación de Android https://developer.android.com/)

4. Mejorar la UI del listado de conversaciones

Historia de usuario

El listado de conversaciones debe mostrar más información al usuario sin necesidad de acceder dentro de la conversación. Para ello se modificará la UI de cada fila incluyendo los siguientes datos:
-. Día y hora del último mensaje
-. Imagen asociada a la conversación

Detalles de implementación

-. El modelo de datos y el ViewHolder debe ser actualizado con los nuevos datos

5. Mejorar la UI de la conversación

Historia de usuario

El usuario, una vez accede a la conversación, ve el listado de mensajes enviados por cada usuario. Este listado podría estar vacío porque la conversación todavía no tiene ningún mensaje. El usuario debería saber que esa conversación no tiene ningún mensaje mediante un “Empty State”. Además, el listado de mensajes debe mostrar más información acerca del mensaje que ha sido mandado.

Detalles de implementación

-. En el caso de que la conversación esté vacía, mostrar una vista que indique que no hay mensajes
-. Cada mensaje debe mostrar el avatar del usuario que mandó el mensaje, el nombre del usuario y el mensaje enviado
-. Sería interesante tener diferentes ViewHolders por cada tipo de fila para poder mostrar diferente información en el caso de que el mensaje haya sido enviado por el propio usuario o por otra persona.

6. Controlar comportamiento de envío de mensajes

Historia de usuario

El usuario no debería de ser capaz de enviar mensajes con texto vacío. Para ello deberíamos habilitar/deshabilitar el botón de enviar mensaje

Detalles de implementación

-. Controlar el texto que el usuario ha ido introduciendo y habilitar o deshabilitar el botón de envío dependiendo de este texto.

Evaluación

Se deberán entregar las 6 tareas anteriores implementadas sobre la aplicación base que se provee al alumno. El alumno puede añadir cualquier otra mejora que considere oportuna. Se valorará la calidad del código fuente así cómo el uso de Kotlin y la aplicación del patrón de diseño MVVM. Se valorará positivamente la creación de tests que comprueban el correcto funcionamiento de las partes más delicadas de la aplicación

