Universidad Politécnica de Chiapas

Ingeniería en Tecnologías de la Información e Innovación Digital

[Programación para Móviles]

[C1-A6-Login con API y

manejo de pantalla]

[Nomenclatura del nombre de archivo: C1-A6-Login con API y manejo de pantalla
-223216-DanielPeregrinoPerez.pdf]

[Alumno – Peregrino Pérez Daniel] - [223216]

Docente: [José Alonso Macias Montoya]

Fecha de entrega: [26/05/2025]

1. DESCRIPCIÓN DE LA ACTIVIDAD

1.1. Enunciado del problema

El problema consistió en desarrollar una aplicación cliente para Android capaz de interactuar con una API RESTful existente para gestionar la autenticación de usuarios. La aplicación deb´ıa conectarse a un servicio backend (API REST) previamente desplegado, el cual gestiona la autenticación y los datos de los usuarios. La aplicación cliente deb´ıa consumir los endpoints de la API para el registro y el login, manejar las respuestas (éxito y error), y persistir el estado de la sesión del usuario localmente. Las funcionalidades principales requeridas fueron el registro de nuevos usuarios y el inicio de sesión de usuarios existentes, permitiendo la navegación a una pantalla principal tras una autenticación exitosa.

1.2. Objetivos de aprendizaje

- Comprender e implementar el patrón de arquitectura MVVM (Modelo-Vista-ViewModel) en una aplicación Android.
- Utilizar Kotlin como lenguaje de programación principal para el desarrollo Android.
- Integrar y consumir una API RESTful utilizando la librer´ıa Retrofit para la comunicación de red.
- Manejar respuestas utilizando la librer´ıa Retrofit.
- Manejar la asincron´ıa en operaciones de red mediante Coroutines de Kotlin.
- Convertir respuestas JSON a objetos Kotlin utilizando Gson.
- Implementar la persistencia de datos básicos (como tokens JWT y datos de usuario) usando SharedPreferences.
- Implementar la navegación entre pantallas (Activities).
- Gestionar el estado de la interfaz de usuario de autenticación.
- Diseñar interfaces de usuario funcionales y responsivas utilizando XML layouts, LiveData y ViewBinding.
- Gestionar el ciclo de vida de las actividades y la navegación entre ellas.
- Depurar y solucionar problemas comunes en el desarrollo Android utilizando Logcat y otras herramientas de Android Studio.
- Entender los conceptos básicos de autenticación basada en tokens (JWT).

2. FUNDAMENTOS TEÓRICOS

Para el desarrollo de esta actividad, se aplicaron los siguientes fundamentos teóricos y tecnolog´ıas:

- **Android SDK:** Conjunto de herramientas de desarrollo para crear aplicaciones para el sistema operativo Android.
- **Kotlin:** Lenguaje de programación moderno, conciso y seguro, oficialmente soportado para el desarrollo Android.
- **Arquitectura MVVM (Modelo-Vista-ViewModel):** Patrón de diseño que separa la lógica de la interfaz de usuario (View) de la lógica de negocio (ViewModel) y los datos (Model/Repository). Facilita la separación de responsabilidades, mejorando la testeabilidad y mantenibilidad del código.
- **Retrofit:** Cliente HTTP type-safe para Android y Java, utilizado para consumir APIs REST. Simplifica la definición de endpoints y la serialización/deserialización de datos.
- **Gson:** Librer´ıa de Google para convertir objetos Java/Kotlin a su representación JSON y viceversa.
- **Coroutines de Kotlin:** Facilidad para escribir código as´ıncrono de manera secuencial y legible, esencial para operaciones de red sin bloquear el hilo principal.
- **LiveData:** Clase observable del ciclo de vida de Android Jetpack. Se utiliza para notificar a la UI sobre cambios en los datos desde el ViewModel.
- **ViewModel:** Clase de Android Jetpack diseñada para almacenar y gestionar datos relacionados con la UI de forma consciente del ciclo de vida.
- **ViewBinding:** Caracter´ıstica que facilita la interacción con las vistas definidas en archivos XML de forma segura y concisa, reemplazando ‘findViewById‘.
- **SharedPreferences:** Mecanismo para almacenar pares clave-valor de datos primitivos de forma persistente en el dispositivo. Utilizado para guardar el token de autenticación y otros datos de sesión.
- **JWT (JSON Web Tokens):** Estándar abierto (RFC 7519) para crear tokens de acceso que permiten la propagación segura de identidad.
- **Material Design Components:** Biblioteca de componentes de UI que implementan las gu´ıas de Material Design, facilitando la creación de interfaces visualmente atractivas y consistentes.

3. DESARROLLO DE LA ACTIVIDAD

3.1. Configuración del Proyecto

El desarrollo de la aplicación se estructuró siguiendo el patrón MVVM y se dividió en las siguientes etapas:

- Creación de un nuevo proyecto en Android Studio.
- Adición de las dependencias necesarias en ‘build.gradle (Module :app)‘ para Retrofit, Gson, ViewModel, LiveData, Coroutines, Material Components y ConstraintLayout.

- Habilitación de ViewBinding en ‘build.gradle‘.
- Adición del permiso de Internet en ‘AndroidManifest.xml‘.

3.2. Implementación de la Lógica de Autenticación

Capa de Datos (Data Layer)

- **Modelado de Datos (data/model):** Creación de data classes en Kotlin (‘LoginRequest‘, ‘LoginResponse‘, ‘RegisterRequest‘, ‘RegisterResponse‘, ‘User‘, ‘ErrorResponse‘) para mapear los JSON de la API.
- **Servicio API (data/remote/ApiService):** Definición de la interfaz con los endpoints de la API (‘/auth/login‘, ‘/auth/register‘) usando anotaciones de Retrofit.
- **Cliente Retrofit (data/remote/RetrofitClient):** Configuración de la instancia singleton de Retrofit, incluyendo la URL base de la API, el convertidor Gson y un interceptor de logging para depuración.
- **Repositorio (data/repository/AuthRepository):** Clase responsable de abstraer el origen de datos. Contiene funciones suspendidas que llaman a los métodos del ‘ApiService‘ para realizar las peticiones de login y registro.

Capa de ViewModel (ui/login/LoginViewModel)

- Se creó ‘LoginViewModel‘ que interactúa con ‘AuthRepository‘.
- Expone ‘LiveData‘ (‘loginResult‘, ‘registerResult‘) de tipo ‘Resource¡T¿‘ (una clase sellada para manejar estados de Éxito, Error y Cargando) para comunicar los resultados de las operaciones a la Vista.
- Lanza las operaciones de red dentro de ‘viewModelScope‘ usando coroutines.
- Implementa lógica para manejar las respuestas de la API, incluyendo el parseo de errores HTTP.

Capa de Vista (View - ui/login/LoginActivity, ui/main/MainActivity)

- **Diseño de Layouts (XML):** Creación de ‘activity*ogin.xml*‘(*camposparaemail/usuario/contrase*
- ▪ Utiliza ViewBinding para acceder a las vistas.
  - Observa los ‘LiveData‘ del ‘LoginViewModel‘ para actualizar la UI (mostrar progreso, mensajes de error/éxito, navegar).
  - Configura listeners para los botones de login y registro, recopilando los datos de los ‘EditText‘ y llamando a los métodos correspondientes del ‘LoginViewModel‘.
  - Verifica si ya existe un token al inicio para navegar directamente a la pantalla principal.

MainActivity (posteriormente FeaturesDemoActivity):

- Muestra un mensaje de bienvenida (obteniendo el nombre de usuario de ‘SessionManager‘).

- Implementa un botón de logout que borra los datos de ‘SessionManager‘ y navega de vuelta a ‘LoginActivity‘.

Gestión de Sesión (util/SessionManager): Clase para guardar y recuperar el token JWT y datos del usuario de ‘SharedPreferences‘.

Utilidades (util/):

- ‘Constants.kt‘: Para la URL base de la API.
- ‘Resource.kt‘: Clase sellada para manejar los estados de las respuestas de la API.

Configuración de Temas y Estilos

Se ajustó el tema de la aplicación en ‘themes.xml‘ para asegurar que heredara de un tema de Material Components (‘Theme.Material3.DayNight.NoActionBar‘) para la correcta visualización de los widgets y el uso de atributos de tema, evitando así errores de ‘InflateException‘ con componentes Material.

3.3. Implementación (opcional) - Funcionalidades Adicionales

Para enriquecer la aplicación y demostrar diversas capacidades de Android, se implementaron las siguientes funcionalidades adicionales después del login exitoso, todas ellas accesibles desde una nueva actividad llamada ‘FeaturesDemoActivity‘:

1. **Visualización de Imágenes:**
   - *Imagen desde Recurso:* Se muestra una imagen almacenada localmente en la carpeta res/drawable utilizando un ImageView y el atributo android:src.
   - *Imagen desde Internet:* Se carga y muestra una imagen desde una URL pública utilizando la librería Glide. Se incluyen placeholders para estados de carga y error.
     - *Dependencia añadida:* com.github.bumptech.glide:glide
   - *Imagen desde Recurso Escalada:* Se demuestra el uso del atributo android:scaleType (ej. centerCrop) en un ImageView para controlar cómo se ajusta una imagen local a las dimensiones del ImageView.
2. **Manejo de Orientación de Pantalla:**
   - La FeaturesDemoActivity permite la rotación de pantalla por defecto.
   - Se explicó cómo se podría fijar la orientación (ej. android:screenOrientation="portrait") en AndroidManifest.xml si fuese necesario.
   - Se destacó la importancia de que los ViewModel sobrevivan a los cambios de configuración para retener los datos.
3. **Implementación de Pestañas (Tabs):**
   - Se utilizaron TabLayout y ViewPager2 para crear una interfaz con cuatro pestañas.
   - Cada pestaña es un Fragment independiente (TabOneFragment, TabTwoFragment, etc.).

- Se creó un TabsPagerAdapter (que extiende FragmentStateAdapter) para gestionar los fragmentos del ViewPager2.
- La configuración se realizó en FeaturesDemoActivity, vinculando el TabLayout con el ViewPager2 mediante TabLayoutMediator.

4. **Obtención de Posición GPS:**

- Se solicitan los permisos ACCESS FINE LOCATION y ACCESS COARSE LOCATION en tiempo de ejecución.
- Se utiliza FusedLocationProviderClient de Google Play Services para obtener actualizaciones de la ubicación.
  - *Dependencia añadida:* com.google.android.gms:play-services-location
- Se creó un LocationRepository para abstraer la lógica de obtención de la ubicación y exponer los datos mediante LiveData.
- El FeaturesDemoViewModel consume este repositorio y expone la ubicación a FeaturesDemoActivity.
- La ubicación (latitud y longitud) se muestra en una MaterialCardView.
- Se maneja el inicio y detención de las actualizaciones de ubicación para optimizar el uso de bater´ıa y evitar fugas de memoria.

5. **Obtención de Información de la Pantalla:**

- Se obtiene y muestra información sobre la pantalla del dispositivo: Ancho y alto en p´ıxeles, Densidad de p´ıxeles (DPI), Factor de densidad, Tasa de refresco (Hz).
- Esta información se obtiene en FeaturesDemoActivity utilizando WindowManager y DisplayMetrics.
- Los datos se pasan al FeaturesDemoViewModel que los expone mediante LiveData para ser observados y mostrados en una MaterialCardView.

Estructura de Archivos Modificada/Añadida para Nuevas Funcionalidades:

```
com.example.login_validado/
        data/
            model/
                DisplayInfo.kt
                LocationData.kt
            repository/
                LocationRepository.kt
        ui/
            featuresdemo/
                FeaturesDemoActivity.kt
                FeaturesDemoViewModel.kt
                adapter/
                    TabsPagerAdapter.kt
                fragments/
                    TabOneFragment.kt
                    TabTwoFragment.kt
                    TabThreeFragment.kt
                    TabFourFragment.kt
        res/
            drawable/
                local_image_example.png
                placeholder_image.png
```

```
                error_image.png
            layout/
                activity_features_demo.xml
                fragment_tab_one.xml
                fragment_tab_two.xml
                fragment_tab_three.xml
                fragment_tab_four.xml
```

Listing 1: Estructura de Archivos Relevante

(Nota: Otros archivos como LoginActivity.kt, LoginViewModel.kt, AndroidManifest.xml, build.gradle y themes.xml fueron modificados para soportar estas nuevas funcionalidades y corregir errores).

4. RESULTADOS

4.1. Resultados obtenidos

Se obtuvo una aplicación Android funcional que cumple con los siguientes requisitos:

- Permite a un nuevo usuario registrarse proporcionando nombre de usuario, email y contraseña. La API devuelve un mensaje de éxito (y un token, aunque esto se simuló temporalmente en el cliente).
- Permite a un usuario existente iniciar sesión con su email y contraseña. La API devuelve un mensaje de éxito (y un token/datos de usuario, simulado temporalmente).
- Muestra Toast con mensajes de la API o de error.
- En caso de login/registro exitoso, guarda el token de autenticación y navega a una pantalla de demostración de funcionalidades (‘FeaturesDemoActivity‘).
- La pantalla de demostración (‘FeaturesDemoActivity‘) muestra:
  - Imágenes cargadas desde recursos locales y desde internet.
  - Una interfaz con 4 pestañas funcionales.
  - La posición GPS actual del dispositivo.
  - Información detallada de la pantalla del dispositivo.
- La persistencia del token mediante ‘SessionManager‘ permite que el usuario no tenga que iniciar sesión cada vez que abre la aplicación (si el token sigue siendo válido y está guardado).
- La estructura MVVM facilitó la separación de responsabilidades y la organización del código.


2:08
Imagen desde Recurso
Imagen desde Internet
Imagen desde Recurso Escalada (centerCrop)
Posición GPS
Latitud: 16.6168768
Longitud: -93.0969072
Información de Pantalla
Ancho: 1080 px
Alto: 2400 px
Densidad (DPI): 420
Densidad (Factor): 2.625
Tasa de Refresco: 60.0 Hz
Info Tab 1
Data Tab 2
Config Tab 3
Extra Tab 4
Contenido del Tab Uno

4.2. Análisis de resultados

La aplicación se comporta según lo esperado. Las solicitudes a la API (simulando la respuesta completa en el cliente cuando fue necesario) se realizan y las respuestas son procesadas para actualizar la UI. El flujo de navegación entre la pantalla de login y la pantalla de funcionalidades es coherente. El manejo de errores básicos (como credenciales incorrectas o campos vac´ıos) funciona como se diseñó, mostrando mensajes informativos al usuario. Las nuevas funcionalidades implementadas en 'FeaturesDemoActivity' demuestran la integración de diversas capacidades de Android.

5. CONCLUSIONES

- **Aprendizaje:**
  - Se consolidó el conocimiento sobre el patrón MVVM y su aplicación práctica en Android.
  - Se adquirió experiencia en el consumo de APIs REST con Retrofit y el manejo de JSON con Gson.
  - Se mejoraron las habilidades en el uso de ViewBinding, Coroutines para programación as´ıncrona y LiveData para la comunicación reactiva entre ViewModel y UI.
  - Se reforzó la importancia de una buena estructura de proyecto y la separación de concerns.
  - Se aprendió a integrar y utilizar componentes de Material Design y a configurar correctamente los temas de la aplicación.
  - Se obtuvo experiencia en la implementación de funcionalidades comunes como tabs, carga de imágenes, y uso de servicios de localización.
- **Objetivos:**
  - Se cumplieron los objetivos de la actividad, logrando una aplicación funcional de autenticación con funcionalidades adicionales.
  - La evaluación de los problemas encontrados y sus soluciones (como la configuración de temas y el manejo de respuestas de API incompletas) fue positiva, aplicando los conceptos teóricos fundamentales.
- **Aplicación:**
  - Los conocimientos adquiridos son directamente aplicables a una amplia gama de proyectos Android que requieran interacción con servicios web, gestión de datos, y una UI moderna y responsiva.
  - La estructura MVVM y las librer´ıas utilizadas (Retrofit, Glide, Material Components) son estándares en la industria, lo que prepara para desarrollos más complejos.

6. DIFICULTADES Y SOLUCIONES

Identificación del problema	Análisis de causas	Implementación de solución	Verificación
La aplicación no redirig´ıa después de un login exitoso, aunque la API respond´ıa con un mensaje de éxito.	El Logcat reveló que la API (en su estado actual) no devolv´ıa el campo 'token' ni 'user' en la respuesta JSON, solo un mensaje. La lógica en 'LoginActivity' esperaba un token para proceder.	Se implementó una **solución temporal** en 'LoginViewModel': si la respuesta era exitosa (código 200 y mensaje de éxito) pero el token era nulo, se generaba un token y usuario falsos.	Con la solución temporal, la app ahora navega a 'FeaturesDemoActivity' tras ingresar credenciales válidas.
Crash al iniciar 'FeaturesDemoActivity' con 'InflateException' para 'MaterialCardView'.	El Logcat indicó: 'IllegalArgumentException: The style on this component requires your app theme to be Theme.MaterialComponents (or a descendant)'. El tema de la app no heredaba de Material Components.	Se modificó 'res/values/themes.xml' (y su versión nocturna) para que el tema principal ('Theme.Login*V alidado*')*her*	*correctamente. Material3.D*
Crash al iniciar 'FeaturesDemoActivity' con 'UnsupportedOperationException' al intentar obtener 'display' desde 'ViewModel'.	El Logcat indicó: 'Tried to obtain display from a Context not associated with one'. Se intentaba acceder a 'context.display' usando el 'applicationContext' dentro del 'ViewModel'.	Se refactorizó: 'FeaturesDemoActivity' ahora obtiene 'DisplayMetrics' y tasa de refresco (usando el contexto de la Activity) y la pasa al 'ViewModel' v´ıa 'setDisplayInfo()'.	La app dejó de crashear y la info de pantalla se mostró correctamente.
Errores Ünresolved reference"para clases de ViewBinding de los Fragmentos de las pestañas (ej. 'FragmentTabThreeBinding').	Clases de ViewBinding no generadas/reconocidas. Posibles causas: nombres XML incorrectos, errores XML, o necesidad de sincronizar/reconstruir.	Se verificaron nombres y contenido XML de los layouts de fragmentos. Se realizaron Çlean Project", Rebuild Projectz "Sync Project with Gradle Files".	Los errores desaparecieron y las clases ViewBinding fueron reconocidas.

7. REFERENCIAS

- Documentación oficial de Android Developers: https://developer.android.com
- Gu´ıa de Arquitectura de Aplicaciones Android (MVVM): https://developer.android.com/jetpack/guide
- Documentación de Kotlin: https://kotlinlang.org/docs/home.html
- Documentación de Retrofit: https://square.github.io/retrofit/
- Documentación de Glide: https://bumptech.github.io/glide/
- Material Components for Android: https://material.io/components/android
- API de Render (Despliegue de backend): https://render.com
- Repositorio de la API backend utilizada (ejemplo): https://github.com/rauw15/Login-API-android-studio (o la URL real de tu API).