Universidad Polit´ecnica de Chiapas

Ingenier´ıa en Tecnolog´ıas de la Informacio´n e Innovacio´n Digital

# [Programacio´n para M´oviles] [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 P´erez Daniel] - [223216]



Docente: [Jos´e Alonso Macias Montoya] Fecha de entrega: [26/05/2025]

# DESCRIPCIO´N DE LA ACTIVIDAD

## Enunciado del problema

El problema consistio´ en desarrollar una aplicacio´n cliente para Android capaz de inter- actuar con una API RESTful existente para gestionar la autenticaci´on de usuarios. La aplicacio´n deb´ıa conectarse a un servicio backend (API REST) previamente desplegado, el cual gestiona la autenticacio´n y los datos de los usuarios. La aplicacio´n cliente deb´ıa consumir los endpoints de la API para el registro y el login, manejar las respuestas (´exi- to y error), y persistir el estado de la sesi´on del usuario localmente. Las funcionalidades principales requeridas fueron el registro de nuevos usuarios y el inicio de sesi´on de usua- rios existentes, permitiendo la navegacio´n a una pantalla principal tras una autentica- cio´n exitosa.

## Objetivos de aprendizaje

 Comprender e implementar el patro´n de arquitectura MVVM (Modelo-Vista-ViewModel) en una aplicaci´on Android.

Utilizar Kotlin como lenguaje de programacio´n principal para el desarrollo Android.

Integrar y consumir una API RESTful utilizando la librer´ıa Retrofit para la comuni- cacio´n 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 b´asicos (como tokens JWT y datos de usuario) usando SharedPreferences.

Implementar la navegacio´n entre pantallas (Activities).

Gestionar el estado de la interfaz de usuario de autenticaci´on.

Disen˜ar interfaces de usuario funcionales y responsivas utilizando XML layouts, Live- Data y ViewBinding.

Gestionar el ciclo de vida de las actividades y la navegacio´n entre ellas.

Depurar y solucionar problemas comunes en el desarrollo Android utilizando Logcat y otras herramientas de Android Studio.

 Entender los conceptos ba´sicos de autenticaci´on basada en tokens (JWT).

# FUNDAMENTOS TEO´RICOS

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

**Android SDK:** Conjunto de herramientas de desarrollo para crear aplicaciones para el sistema operativo Android.

**Kotlin:** Lenguaje de programacio´n moderno, conciso y seguro, oficialmente soporta- do para el desarrollo Android.

**Arquitectura MVVM (Modelo-Vista-ViewModel):** Patro´n de disen˜o que sepa- ra la lo´gica de la interfaz de usuario (View) de la lo´gica de negocio (ViewModel) y los datos (Model/Repository). Facilita la separacio´n de responsabilidades, mejorando la testeabilidad y mantenibilidad del co´digo.

**Retrofit:** Cliente HTTP type-safe para Android y Java, utilizado para consumir APIs REST. Simplifica la definici´on de endpoints y la serializacio´n/deserializaci´on de datos.

**Gson:** Librer´ıa de Google para convertir objetos Java/Kotlin a su representaci´on JSON y viceversa.

**Coroutines de Kotlin:** Facilidad para escribir c´odigo as´ıncrono de manera secuen- cial 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 no- tificar a la UI sobre cambios en los datos desde el ViewModel.

**ViewModel:** Clase de Android Jetpack disen˜ada para almacenar y gestionar datos relacionados con la UI de forma consciente del ciclo de vida.

**ViewBinding:** Caracter´ıstica que facilita la interaccio´n con las vistas definidas en archivos XML de forma segura y concisa, reemplazando ‘findViewById‘.

**SharedPreferences:** Mecanismo para almacenar pares clave-valor de datos primiti- vos de forma persistente en el dispositivo. Utilizado para guardar el token de autenti- cacio´n y otros datos de sesio´n.

**JWT (JSON Web Tokens):** Esta´ndar abierto (RFC 7519) para crear tokens de acceso que permiten la propagaci´on segura de identidad.

 **Material Design Components:** Biblioteca de componentes de UI que implementan las gu´ıas de Material Design, facilitando la creaci´on de interfaces visualmente atracti- vas y consistentes.

# DESARROLLO DE LA ACTIVIDAD

## Configuracio´n del Proyecto

El desarrollo de la aplicaci´on se estructur´o siguiendo el patr´on MVVM y se dividi´o en las siguientes etapas:

Creacio´n de un nuevo proyecto en Android Studio.

Adicio´n de las dependencias necesarias en ‘build.gradle (Module :app)‘ para Retrofit, Gson, ViewModel, LiveData, Coroutines, Material Components y ConstraintLayout.

Habilitacio´n de ViewBinding en ‘build.gradle‘.

 Adicio´n del permiso de Internet en ‘AndroidManifest.xml‘.

## Implementacio´n de la L´ogica de Autenticaci´on

### Capa de Datos (Data Layer)

 **Modelado de Datos (data/model):** Creacio´n de data classes en Kotlin (‘LoginRe- quest‘, ‘LoginResponse‘, ‘RegisterRequest‘, ‘RegisterResponse‘, ‘User‘, ‘ErrorRespon- se‘) para mapear los JSON de la API.

**Servicio API (data/remote/ApiService):** Definicio´n de la interfaz con los end- points de la API (‘/auth/login‘, ‘/auth/register‘) usando anotaciones de Retrofit.

**Cliente Retrofit (data/remote/RetrofitClient):** Configuracio´n de la instancia singleton de Retrofit, incluyendo la URL base de la API, el convertidor Gson y un interceptor de logging para depuracio´n.

 **Repositorio (data/repository/AuthRepository):** Clase responsable de abstraer el origen de datos. Contiene funciones suspendidas que llaman a los m´etodos del ‘Api- Service‘ para realizar las peticiones de login y registro.

### Capa de ViewModel (ui/login/LoginViewModel)

 Se cre´o ‘LoginViewModel‘ que interactu´a con ‘AuthRepository‘.

Expone ‘LiveData‘ (‘loginResult‘, ‘registerResult‘) de tipo ‘Resource¡T¿‘ (una clase sellada para manejar estados de E´xito, Error y Cargando) para comunicar los resulta- dos de las operaciones a la Vista.

Lanza las operaciones de red dentro de ‘viewModelScope‘ usando coroutines.

Implementa lo´gica para manejar las respuestas de la API, incluyendo el parseo de errores HTTP.

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

 **Disen˜o de Layouts (XML):** Creacio´n de ‘activity*login.xml*‘(*camposparaemail/usuario/contrase *  Utiliza ViewBinding para acceder a las vistas.

Observa los ‘LiveData‘ del ‘LoginViewModel‘ para actualizar la UI (mostrar pro- greso, mensajes de error/´exito, navegar).

Configura listeners para los botones de login y registro, recopilando los datos de los ‘EditText‘ y llamando a los m´etodos correspondientes del ‘LoginViewModel‘.

 Verifica si ya existe un token al inicio para navegar directamente a la pantalla prin- cipal.

### MainActivity (posteriormente FeaturesDemoActivity):

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

 Implementa un bot´on de logout que borra los datos de ‘SessionManager‘ y navega de vuelta a ‘LoginActivity‘.

**Gesti´on de Sesio´n (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.

### Configuracio´n de Temas y Estilos

Se ajusto´ el tema de la aplicacio´n en ‘themes.xml‘ para asegurar que heredara de un tema de Material Components (‘Theme.Material3.DayNight.NoActionBar‘) para la co- rrecta visualizaci´on de los widgets y el uso de atributos de tema, evitando as´ı errores de ‘InflateException‘ con componentes Material.

## Implementacio´n (opcional) - Funcionalidades Adicionales

Para enriquecer la aplicacio´n y demostrar diversas capacidades de Android, se imple- mentaron las siguientes funcionalidades adicionales despu´es del login exitoso, todas ellas accesibles desde una nueva actividad llamada ‘FeaturesDemoActivity‘:

### Visualizaci´on de Im´agenes:

 *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 pu´blica utilizando la librer´ıa Glide. Se incluyen placeholders para estados de carga y error.

* + *Dependencia an˜adida:* com.github.bumptech.glide:glide

 *Imagen desde Recurso Escalada:* Se demuestra el uso del atributo android:scaleType

(ej. centerCrop) en un ImageView para controlar c´omo se ajusta una imagen local a las dimensiones del ImageView.

### Manejo de Orientacio´n de Pantalla:

 La FeaturesDemoActivity permite la rotacio´n de pantalla por defecto.

 Se explic´o co´mo se podr´ıa fijar la orientaci´on (ej. android:screenOrientation="portrait") en AndroidManifest.xml si fuese necesario.

 Se destac´o la importancia de que los ViewModel sobrevivan a los cambios de configuracio´n para retener los datos.

### Implementacio´n de Pestan˜as (Tabs):

Se utilizaron TabLayout y ViewPager2 para crear una interfaz con cuatro pes- tan˜as.

Cada pestan˜a es un Fragment independiente (TabOneFragment, TabTwoFragment, etc.).

 Se cre´o un TabsPagerAdapter (que extiende FragmentStateAdapter) para ges- tionar los fragmentos del ViewPager2.

 La configuraci´on se realiz´o en FeaturesDemoActivity, vinculando el TabLayout

con el ViewPager2 mediante TabLayoutMediator.

### Obtencio´n de Posici´on GPS:

 Se solicitan los permisos ACCESS FINE LOCATION y ACCESS COARSE LOCATION en tiempo de ejecucio´n.

 Se utiliza FusedLocationProviderClient de Google Play Services para obte- ner actualizaciones de la ubicacio´n.

* + *Dependencia an˜adida:* com.google.android.gms:play-services-location

 Se cre´o un LocationRepository para abstraer la lo´gica de obtencio´n de la ubi- cacio´n y exponer los datos mediante LiveData.

 El FeaturesDemoViewModel consume este repositorio y expone la ubicacio´n a

FeaturesDemoActivity.

 La ubicaci´on (latitud y longitud) se muestra en una MaterialCardView.

 Se maneja el inicio y detenci´on de las actualizaciones de ubicaci´on para optimi- zar el uso de bater´ıa y evitar fugas de memoria.

### Obtencio´n de Informaci´on de la Pantalla:

 Se obtiene y muestra informacio´n sobre la pantalla del dispositivo: Ancho y al- to en p´ıxeles, Densidad de p´ıxeles (DPI), Factor de densidad, Tasa de refresco (Hz).

 Esta informaci´on 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/An˜adida para Nuevas Funcionalidades:

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | com . example . login\_validado /  data /  model/  DisplayInfo . kt  Location Data . kt | |
| 6 |  | repository / |
| 7 |  | Location Repository . kt |
| 8 | ui/ |  |
| 9 |  | featuresdemo / |
| 10 |  | FeaturesDemo Activity . kt |
| 11 |  | FeaturesDemo View Model . kt |
| 12 |  | adapter/ |
| 13 |  | TabsPagerAdapter . kt |
| 14 |  | fragments/ |
| 15 |  | Tab One Fragment . kt |
| 16 |  | Tab Two Fragment . kt |
| 17 |  | Tab Three Fragment . kt |
| 18 |  | Tab FourFragment . kt |
| 19 | res/ |  |
| 20 |  | drawable / |
| 21 |  | local\_image\_example . png |
| 22 |  | placeholder\_image . png |

23

error\_image . png

layout/

activity\_features\_demo . xml fragment\_tab\_one . xml fragment\_tab\_two . xml fragment\_tab\_three . xml fragment\_tab\_four . xml

24

25

26

27

28

29

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).*

# RESULTADOS

## Resultados obtenidos

Se obtuvo una aplicacio´n Android funcional que cumple con los siguientes requisitos:

 Permite a un nuevo usuario registrarse proporcionando nombre de usuario, email y contrasen˜a. La API devuelve un mensaje de ´exito (y un token, aunque esto se simulo´ temporalmente en el cliente).

 Permite a un usuario existente iniciar sesi´on con su email y contrasen˜a. La API de- vuelve un mensaje de ´exito (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 autenticacio´n y navega a una pantalla de demostraci´on de funcionalidades (‘FeaturesDemoActivity‘).

 La pantalla de demostracio´n (‘FeaturesDemoActivity‘) muestra:

* Ima´genes cargadas desde recursos locales y desde internet.
* Una interfaz con 4 pestan˜as funcionales.
* La posici´on GPS actual del dispositivo.
* Informacio´n detallada de la pantalla del dispositivo.

 La persistencia del token mediante ‘SessionManager‘ permite que el usuario no tenga que iniciar sesio´n cada vez que abre la aplicacio´n (si el token sigue siendo va´lido y esta´ guardado).

 La estructura MVVM facilito´ la separacio´n de responsabilidades y la organizacio´n del co´digo.



## An´alisis de resultados

La aplicaci´on se comporta segu´n 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 navegaci´on entre la pantalla de login y la pantalla de funcionalidades es coherente. El manejo de errores ba´sicos (como credencia- les incorrectas o campos vac´ıos) funciona como se disen˜o´, mostrando mensajes informa- tivos al usuario. Las nuevas funcionalidades implementadas en ‘FeaturesDemoActivity‘ demuestran la integracio´n de diversas capacidades de Android.

# CONCLUSIONES

### Aprendizaje:

* Se consolid´o el conocimiento sobre el patro´n MVVM y su aplicacio´n pra´ctica en Android.
* Se adquiri´o 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 progra- macio´n as´ıncrona y LiveData para la comunicacio´n reactiva entre ViewModel y UI.
* Se reforz´o la importancia de una buena estructura de proyecto y la separacio´n de concerns.
* Se aprendi´o a integrar y utilizar componentes de Material Design y a configurar correctamente los temas de la aplicaci´on.
* Se obtuvo experiencia en la implementacio´n de funcionalidades comunes como tabs, carga de ima´genes, y uso de servicios de localizaci´on.

### Objetivos:

* Se cumplieron los objetivos de la actividad, logrando una aplicacio´n funcional de autenticacio´n con funcionalidades adicionales.
* La evaluacio´n de los problemas encontrados y sus soluciones (como la configura- cio´n de temas y el manejo de respuestas de API incompletas) fue positiva, apli- cando los conceptos teo´ricos fundamentales.

### Aplicaci´on:

* Los conocimientos adquiridos son directamente aplicables a una amplia gama de proyectos Android que requieran interacci´on con servicios web, gestio´n de datos, y una UI moderna y responsiva.
* La estructura MVVM y las librer´ıas utilizadas (Retrofit, Glide, Material Compo- nents) son est´andares en la industria, lo que prepara para desarrollos m´as com- plejos.

# DIFICULTADES Y SOLUCIONES

2

|  |  |  |  |
| --- | --- | --- | --- |
| **Identificacio´n**  **del problema** | **An´alisis de cau-**  **sas** | **Implementacio´n de**  **soluci´on** | **Verificaci´on** |
| La aplicaci´on no  redirig´ıa despu´es de un login exito- so, aunque la API respond´ıa con un mensaje de ´exito. | El Logcat revel´o  que la API (en su estado actual) no devolv´ıa el campo ‘token‘ ni ‘user‘ en la respuesta JSON, solo un  mensaje. La lo´gica en ‘LoginActivity‘ esperaba un token  para proceder. | Se implemento´ una **so-**  **lucio´n temporal** en ‘LoginViewModel‘: si la respuesta era exitosa (co´digo 200 y mensaje de ´exito) pero el token  era nulo, se generaba un token y usuario falsos. | Con la solucio´n  temporal, la app ahora navega a ‘FeaturesDemoAc- tivity‘ tras ingresar credenciales v´ali- das. |
| Crash al iniciar  ‘FeaturesDe- moActivity‘ con ‘InflateException‘ para ‘Material- CardView‘. | El Logcat in-  dico´: ‘IllegalArgu- mentException: The style on this component re- quires your app theme to be The-  me.MaterialCompone (or a descendant)‘.  El tema de la app no heredaba de Material Compo-  nents. | Se modific´o ‘res/value-  s/themes.xml‘ (y su versio´n nocturna) pa- ra que el tema principal  (‘Theme.Login*V alidado*‘)*her* nts | La app dejo´ de  crashear y los com- ponentes Material se renderizaron  *e*c*d*o*a*r*r*re*a*c*d*t*e*a‘m*T h*en*e*t*m*e.*e.M ate* |
| Crash al iniciar  ‘FeaturesDe- moActivity‘ con ‘UnsupportedOpe- rationException‘ al intentar obtener ‘display‘ desde ‘ViewModel‘. | El Logcat indico´:  ‘Tried to obtain display from a Context not as- sociated with one‘. Se inten- taba acceder a ‘context.display‘ usando el ‘appli- cationContext‘  dentro del ‘View- Model‘. | Se refactoriz´o: ‘Featu-  resDemoActivity‘ ahora obtiene ‘DisplayMetrics‘ y tasa de refresco (usando el contexto de la Activity) y la pasa al ‘ViewModel‘ v´ıa ‘setDisplayInfo()‘. | La app dejo´ de  crashear y la in- fo de pantalla se mostro´ correcta- mente. |
| Errores U¨ nresolved  reference”para clases de View- Binding de los Fragmentos de las pestan˜as (ej. ‘FragmentTabTh- reeBinding‘). | Clases de View- Binding no gene- radas/reconocidas. Posibles causas: nombres XML incorrectos, errores XML, o necesidad de sincronizar/re-  construir. | Se verificaron nombres y contenido XML de los  layouts de fragmentos. Se realizaron C¸ lean Project”, Rebuild Project ”Sync Project with Gradle Fi- les”. | Los errores des- aparecieron y las clases ViewBinding fueron reconocidas. |

*rial*3*.D*

# REFERENCIAS

Documentacio´n oficial de Android Developers: [https://developer.android.com](https://developer.android.com/)

Gu´ıa de Arquitectura de Aplicaciones Android (MVVM): [https://developer.android.](https://developer.android.com/jetpack/guide) [com/jetpack/guide](https://developer.android.com/jetpack/guide)

Documentacio´n de Kotlin: <https://kotlinlang.org/docs/home.html>

Documentacio´n de Retrofit: <https://square.github.io/retrofit/>

Documentacio´n 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](https://render.com/)

Repositorio de la API backend utilizada (ejemplo): [https://github.com/rauw15/](https://github.com/rauw15/Login-API-android-studio) [Login-API-android-studio](https://github.com/rauw15/Login-API-android-studio) (o la URL real de tu API).