Instituto Politécnico Nacional

Escuela Superior de Cómputo

Programa académico / Plan de estudios
Ingeniería en Sistemas Computacionales / 2020

Unidad de aprendizaje
Desarrollo de aplicaciones móviles nativas

Tarea 3: "Desarrollo de una Aplicación Móvil Nativa con Consumo de API REST"

Objetivo: El propósito de esta actividad es desarrollar habilidades prácticas en la creación de aplicaciones móviles nativas en Android que consuman servicios REST. Aprenderá a configurar correctamente los permisos de red, realizar peticiones HTTP, procesar respuestas JSON y mostrar la información recibida en la interfaz de usuario.

Ejercicio 1: Creación de un Backend Básico y Conexión con Android

Descripción de la actividad: En esta primera parte, implementará un servicio REST básico y conectará su aplicación Android para consumir dicho servicio.

1. **Desarrollo del Backend**
   - Implemente un servicio REST básico utilizando Spring Boot o el framework de su preferencia.
   - El servicio debe exponer al menos un endpoint que retorne un mensaje simple en formato JSON (por ejemplo, un "Hola Mundo").
   - Configure el servidor para que escuche en un puerto accesible (por ejemplo, el puerto 8080).
   - Documente la URL y la estructura de la respuesta JSON.

2. **Configuración de la Aplicación Android**
   - Cree un nuevo proyecto en Android Studio.
   - Configure los permisos necesarios en el archivo AndroidManifest.xml para permitir la conexión a internet:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

   - Para Android 9 (API 28) o superior, cree un archivo de configuración de seguridad de red network_security_config.xml en la carpeta res/xml/ que permita conexiones a localhost (10.0.2.2):

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">10.0.2.2</domain>
    </domain-config>
</network-security-config>
```

- Referencie la configuración de seguridad en su AndroidManifest.xml:

```
<application
    ...
    android:networkSecurityConfig="@xml/network_security_config"
    ... >
```

3. **Consumo del Servicio REST**
   - Implemente el código necesario para realizar una petición HTTP al servicio REST creado.
   - Utilice la biblioteca Retrofit, Volley o HttpURLConnection para realizar la petición.
   - Procese la respuesta JSON y muestre el contenido en la interfaz de usuario.
   - Implemente un manejo adecuado de errores y estados de carga.
4. **Evidencias:**
   - Capturas de pantalla del backend ejecutándose y mostrando la respuesta JSON.
   - Capturas de pantalla de la aplicación Android mostrando la información recibida del backend.
   - Código fuente relevante comentado (clases de conexión, manejadores de respuesta, etc.).

Ejercicio 2: Consumo de una API Pública

Descripción de la actividad: En esta segunda parte, extenderá sus conocimientos para consumir una API pública y mostrar información más compleja en su aplicación Android.

1. **Selección de una API Pública**
   - Elija una API pública para consumir. Algunas opciones recomendadas son:
     - **Open Library API**: Para búsqueda de libros y información bibliográfica.
     - **TV Maze API**: Para obtener información sobre series y programas de televisión.
     - **The Movie Database (TMDB)**: Para información sobre películas y series.
     - **OpenWeatherMap**: Para datos meteorológicos.
   - Documente la API seleccionada, incluyendo la base URL y los endpoints que planea utilizar.
2. **Diseño de la Interfaz de Usuario**
   - Diseñe una interfaz de usuario que permita:
     - Realizar búsquedas o seleccionar categorías según la API elegida.
     - Mostrar los resultados en una lista o grid utilizando RecyclerView.
     - Ver detalles al seleccionar un elemento de la lista.
   - Incluya elementos visuales como imágenes (si la API proporciona URLs de imágenes).
   - Implemente un diseño responsivo que funcione correctamente en diferentes tamaños de pantalla.
3. **Implementación del Consumo de API**
   - Cree las clases de modelo necesarias para mapear las respuestas JSON de la API.
   - Implemente el servicio de conexión utilizando Retrofit o la biblioteca de su elección.
   - Configure adecuadamente los interceptores o headers necesarios (si la API requiere autenticación).
   - Implemente la lógica de paginación si la API lo soporta y es necesario.
4. **Mejoras de Experiencia de Usuario**
   - Añada elementos de carga (progress bars o skeletons) durante las peticiones.
   - Implemente cacheo de respuestas para mejorar el rendimiento.
   - Configure manejo de errores con mensajes descriptivos y opciones de reintentar.
   - Añada la funcionalidad de "pull to refresh" para actualizar el contenido.
5. **Evidencias:**
   - Capturas de pantalla mostrando la navegación y funcionalidad completa de la aplicación.
   - Capturas de pantalla de diferentes estados (carga, error, sin resultados, etc.).
   - Video breve demostrando la interacción con la aplicación.
   - Código fuente relevante con comentarios explicativos.

Evidencias a entregar:

Suba las siguientes evidencias a Classroom:

1. Código fuente completo de ambos ejercicios, organizado en directorios separados.
2. Capturas de pantalla y videos mostrando la funcionalidad de cada ejercicio.
3. Incluya un archivo README.md en su repositorio, que contenga:
    - Descripción detallada del proyecto.
    - Instrucciones paso a paso para configurar y ejecutar tanto el backend como la aplicación Android.
    - Diagrama o explicación de la arquitectura de la aplicación.
    - Capturas de pantalla relevantes mostrando la funcionalidad.
    - Explicación de los desafíos encontrados y cómo fueron resueltos.
    - Lista de las dependencias utilizadas y su propósito.

Recursos útiles:

1. **Configuración de Retrofit en Android:**
    - Guía oficial de Retrofit
    - Uso de Retrofit con Kotlin y Corrutinas
2. **APIs públicas recomendadas:**
    - Open Library API
    - TV Maze API
    - The Movie Database API
3. **Mejores prácticas para el manejo de JSON en Android:**
    - Uso de Gson o Moshi para procesamiento de JSON
    - Jackson para Android

Fecha de entrega:

- La fecha límite para la entrega de esta práctica es el **martes 11 de marzo de 2025**. No se aceptarán entregas fuera de tiempo y forma.