Resolución de Ejercicio de Altran

Generalidades del desarrollo

Se realizó utilizando AndroidStudio 3.1, compilando con API 27 (Android 8.0), con un target mínimo de API 19 (Android 4.4). Se eligió esa versión de mínimo, por la disponibilidad de dispositivos. Para evitar perdida de información de datos se deshabilito el cambio de orientación del dispositivo.

Se priorizo una arquitectura robusta y escalable a una interfaz atractiva.

El código y un APK está disponible en github:

https://github.com/CorradiSebastian/Gnomes

Arquitectura

Para la resolución del ejercicio se utilizó la arquitectura MVP, dejando el control de la navegación del lado de las vistas.

Librerías

Se utilizaron las siguientes librerías:

- Gson para el mapeo de objetos JSON a Java.
- Volley para manejar los requests.
- Glide Para la carga de imágenes

Alcance

Se implementa la solución al ejercicio pedido: Una aplicación para la visualización, búsqueda y filtro de gnomos.

Se consideran las siguientes situaciones:

- Que las llamadas a los servicios no funcionen bien.
- Que no haya internet (Funcionamiento offline)
- Que los dispositivos estén configurados en ingles o castellano.

Fuera de Alcance

La solución de la aplicación NO va a incluir:

- Selección de skins o estilos de UI.
- Registración y logeo de usuarios.

Supuestos

Durante el desarrollo de la aplicación, se implementaron ciertas funcionalidades basadas en supuestos descriptos a continuación:

- No hay gnomos repetidos.
- En la llamada al servicio que devuelve los gnomos, estos vienen ordenados alfabéticamente por nombre.
- Los colores disponibles de pelo son: pink, red, gray, black y green.
- No se trabajó la UI en forma minuciosa, ya que se supone el trabajo con un equipo de diseño.

Se evitó la implementación de múltiples búsquedas, filtros y ordenamientos suponiendo que la realización de uno ya demuestra el conocimiento del tema.

Desarrollo

Para la búsqueda se utiliza un EditText que realiza una búsqueda incremental de la lista (ya sea filtrada o no).

Para la realización del filtro (y la verificación de su correcto funcionamiento) se incluyó un spinner con los distintos colores de pelo que funciona en forma conjunta con la búsqueda incremental.

Para el almacenamiento de los gnomos se utiliza una base de datos de SQlite, la cual es consultada cuando no hay internet, y sobre la cual se realiza la búsqueda filtrada por color de pelo.

Para evitar el congelamiento de la pantalla durante los procesos largos (como el almacenamiento de gnomos en la BD) se realizan operaciones en un thread nuevo, Se podría haber usado un AsyncTask, pero en la complejidad de la aplicación no lo ameritaba (hubiera sido necesario si, por ejemplo, se hubiera querido mostrar el progreso de los “inserts” en la BD).

Para el ordenamiento de los resultados, se usa una imagen, que se invierte según el ordenamiento haya sido ascendiente o descendiente.

Agregados

Para lograr una rápida visualización se usó una lista expandible donde se visualiza el nombre y la edad de un gnomo por cada fila. También se agregó un dibujo genérico de un gnomo, ajustando el ancho y alto de la imagen a la altura y peso del gnomo como así también el color del pelo.

Posibles Mejoras

Las siguientes son mejoras que se pueden incluir al desarrollo, pero que no se pudieron implementar por falta de tiempo o definición:

- Estilos de visualización de pantalla.
- Actualización periódica de la BD donde se almacenan los gnomos.
- Permitir el cambio de orientación de la pantalla.
- Búsqueda por la totalidad de los campos.
- Selección de lenguaje (la estructura lo soporta, pero no se puede elegir)
- Incluir una pantalla de descripción de errores más amigable.
- Permitir el cambio de orientación de pantalla.
- Incluir Unit Tests

Sebastián Corradi

SebastianCorradi@gmail.com

Licenciado en Análisis de Sistemas (U.B.A.)