Fabián Zamora Ramírez
Jeff Schmidt Peralta

Día 5 (Volley)

1. Agregar dependencias al proyecto

a. En el archivo build.gradle del módulo app se debe agregar la dependencia hacia la biblioteca de soporte de diseño:
   i. *compile 'com.android.volley:volley:1.0.0'*

2. Agregar permisos para Internet

a. Solo se requiere agregar el permiso "android.permission.INTERNET" en el AndroidManifest.xml.

3. Crear VolleyRequestSingleton

a. Crear una clase **VolleyRequestSingleton**. Esta clase debe ser *singleton*. No debe heredar de nada en específico.
b. Esta clase contiene un Queue de volley al cual se le agregan Requests.
c. Esta clase contiene un *Map* que contiene los *Headers* que se deben enviar con cada Request.
d. Agregar método genérico para agregar requests al Queue: **JsonObjectRequest** y **JsonArrayRequest**.
e. Esta clase funciona por medio de callbacks asíncronos. El request se ejecuta en hilos aparte y se notifica su finalización por medio de listeners.

4. Crear BackendAccess para Login

a. Crear clase **BackendAccess.** No debe heredar de nada en específico. Esta es la clase que se va a encargar de crear URL y body para los requests que se van a agregar al Queue de volley a partir de los modelos (información ingresada por el usuario).
b. Esta clase funciona por medio de callbacks también, pero se debe crear un callback propio.
c. Agregar un método para hacer **login()**.

5. Agregar llamadas al Backend para Login

a. Dentro del fragment **LoginFragment** se debe agregar una instancia de **BackendAccess**. La instancia se inicializa dentro del método onCreate(Bundle) del ciclo de vida.

b. Cuando se intenta hacer login (se da click en el botón) se hacen las llamadas correspondientes de **BackendAccess** y se inician las nuevas ventanas o se muestra el error dentro del **callback**.
c. Si hubo éxito se debe pasar a la ventana MainActivity y además se **deben guardar los headers** en la clase **VolleyRequestSingleton** que se van a necesitar para autenticar al usuario en las próximas llamadas.

6. Mantener el la sesion iniciada después de cerrar el app

a. Se debe guardar los datos del usuario localmente para poder iniciar sesión después de haber cerrado el app. En este caso para verificar al usuario basta con **email** y el **authentication token.**
b. Crear una clase **Preferences.** No debe heredar de nada en específico. Se va a encargar de escribir/borrar datos de los **SharedPreferences**.
c. Agregar metodos para guardar / borrar / obtener el **email** y el **authentication token.** Estos valores van a ser guardados y persisten durante todas las sesiones de la aplicación.
d. En el callback donde se guardan los headers (método *login()*, clase **BackendAccess**) ahora también se debe guardar la información del usuario.
e. En el LandingActivity, se debe verificar si ya existe un usuario guardado en los **SharedPreferences.** De ser así, se debe pasar directo a la **MainActivity** y guardar los headers para los request de volley.

7. Cargar las tareas de un usuario

a. Agregar método para obtener las tareas del backend

i. Agregar un método *indexTasks(callback)* a la clase **BackendAccess**. Como ya agregamos los headers para autenticación dentro de *login()*, el backend sabe de cual usuario estamos pidiendo las tareas.

b. Modificar **TasksAdapter**

i. Hay que eliminar los datos dummy que se habían agregado y hacer que el adapter tome la lista a mostrar como un parámetro. **Importante:** en el método *getItemCount()* revisar que la lista de tareas no sea nula.
ii. Dentro del fragment **TasksFragment** se debe guardar la instancia del **Adapter** como un atributo de clase, para indicarle cada vez que se carga la lista de tareas.

iii. Dentro del fragment **TasksFragment** se debe agregar una instancia de **BackendAccess**. La instancia se inicializa dentro del método onCreate(Bundle) del ciclo de vida.

iv. Dentro del fragment **TasksFragment** se debe crear un método *indexTasks()* que utilice la clase **BackendAccess** para obtener la lista de tareas. Dentro del callback, al **adapter** se le pasan las tareas cargadas y se llama el método *Adapter#notifyDataSetChanged()* para que actualice la UI.

v. El nuevo método *indexTasks()* se debe llamar dentro del *onResume()* del fragment **TasksFragment.**

c. Cargar la imagen de la tarea

i. Agregar un ImageView al **layout/row_tasks.xml.**

ii. Agregar una referencia al nuevo ImageView dentro del **TaskHolder**.

iii. En el archivo build.gradle del módulo app se debe agregar la dependencia hacia la biblioteca de soporte de diseño:

1. compile 'com.squareup.picasso:picasso:2.5.2'

iv. Dentro de *TasksAdapter#onBindViewHolder()*, se puede tomar el URL del task y cargar dentro del ImageView utilizando la biblioteca picasso.

8. Crear una nueva tarea en el backend

a. Agregar método para guardar la tareas en el backend

i. Agregar un método *createTask(task, callback)* a la clase **BackendAccess**. Como ya agregamos los headers para autenticación dentro de *login()*, el backend sabe a cual usuario guardarle la tarea.

ii. Dentro del fragment **CreateTaskFragment** se debe agregar una instancia de **BackendAccess**. La instancia se inicializa dentro del método onCreate(Bundle) del ciclo de vida.

iii. Dentro del fragment **CreateTaskFragment** se debe crear un método *attemptSave()* que utilice la clase **BackendAccess** para obtener guardar la tarea.

iv. Para enviar una imagen , en este caso, se manda la imagen como un String codificando los bytes en base 64.

Fabián Zamora Ramírez
Jeff Schmidt Peralta

9. Repetir pasos para Logout y Register