

Talento Digital Android 2021-07-14 (Jue) Módulo 3 - Clase 18

Libreta: GTD

Creado: 29-07-2021 10:42

Actualizado: 29-07-2021 19:50

Autor: Sebastián

Talento Digital Android 2021-07-14 (Jue) Módulo 3 - Clase 18

Día 4, semana 12

Clase:

Android Life Cycle

Cuando las activity se crean, se usan, y luego pierden el control pasan por diferentes estados. Android permite manejar lo que pasa en esos estados/cambios de estado. Permite que la app:

- No falle si el usuario recibe una llamada telefónica o cambia a otra app.
- No consuma recursos valiosos del sistema cuando el usuario no la use de forma activa.
- No pierda el progreso del usuario si este abandona la app y regresa a ella posteriormente.
- No falle ni pierda el progreso del usuario cuando se gire la pantalla entre la orientación horizontal y la vertical.

Android Life Cycle

Los métodos del ciclo de vida:

- **OnCreate:** cuando se crea el activity
- **OnStart:** cuando se entra en estado started, luego de create
- **OnResume:** cuando la actividad entra en el estado Resume, pasa al primer plano
- **OnPause:** cuando se está abandonando la activity
- **OnStop:** cuando el usuario ya no puede ver la actividad
- **OnDestroy:** cuando se va a finalizar la actividad, pero antes de que finalice

Patrón MVP

Componentes de MVP

- **Modelo:** Esta capa de acceso a datos, tales como bases de datos.
- **Vista:** Se encarga de mostrar los datos. Aquí se encuentran nuestras Vistas.
- **Presentador:** es la capa que provee datos desde la vista hasta el modelo. También maneja las tareas en segundo plano.

Callbacks

Función "A" que se usa como argumento de otra función "B"

- Cuando se llama a "B", ésta ejecuta "A"

```
boton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        presionar();  
    }  
});
```

Android y Testing

Android incluye algunas herramientas especiales para facilitar el testing

- EJ: BuildVariants
- Permite crear diferentes versiones de la App usando el mismo código fuente
- Varían en su configuración
- Capacidad de definir código de testing para versiones específicas de la app
- Podemos crear y configurar tipos de compilación en el archivo build.gradle del nivel del módulo dentro del bloque Android

Android y Testing

Permite la muy necesaria tarea de testear la app en diferentes ambientes

- Emuladores
- Dispositivos físicos
- Manteniendo diferentes configuraciones.

JUnit

- Conjunto de bibliotecas para hacer pruebas unitarias de aplicaciones Java
- Pruebas unitarias
 - Aislar una parte del código y comprobar que funcionan según la especificación
 - Son pequeños tests que validan el comportamiento de una pieza de código

JUnit

Ventajas

- Las pruebas unitarias demuestran que la lógica del código está en buen estado y que funcionará en todos los casos.
- Aumentan la legibilidad del código y ayudan a los desarrolladores a entender el código base, lo que facilita hacer cambios más rápidamente.
- Los test unitarios bien realizados sirven como documentación del proyecto.
- Se realizan en pocos milisegundos, por lo que podrás realizar cientos de ellas en muy poco tiempo

Mockito

Mocking permite:

- Probar piezas de software simulando sus dependencias externas
- Permite verificar el comportamiento de la pieza independiente de sus dependencias
- Nos aseguramos que cualquier defecto se deba a un error en el código específico de la pieza

Espresso

Espresso

- Framework para testing de UI
- Permite escribir pruebas de la IU de Android concisas, eficaces y confiables
- Las pruebas de Espresso se ejecutan con una rapidez óptima
- La API de Espresso alienta a los autores de pruebas a pensar en términos de lo que un usuario podría hacer mientras interactúa con la app
- Cuenta con funcionalidad que permite ubicar y aislar elementos de la UI convenientemente

Otras alternativas de testeo

Existen otras alternativas de frameworks y librerías muy útiles para la realización de testing en Android, tales como:

- Appium
- Detox
- Calabash: