UNIVERSIDAD CATÓLICA DE TEMUCO

Informe Evaluación N°5
Paquetización Android (APK) + Implementación de Métricas en Kivy

Nombre: Juan Cornejo

Asignatura: Desarrollo Móvil

Fecha: 20 de noviembre de 2025

UNIVERSIDAD CATÓLICA DE TEMUCO

1. Introducción

El proyecto consiste en una aplicación móvil desarrollada con Python y el framework Kivy, cuyo objetivo fue el de implementar un contador interactivo capaz de registrar métricas de uso del usuario, Además, la aplicación fue paquetizada a formato APK utilizando Buildozer dentro de un entorno WSL/Ubuntu, permitiendo su ejecución en dispositivos Android reales.

Se desarrollaron y documentaron dos componentes principales:

1. **Paquetización de la aplicación Kivy a APK**
2. **Implementación de métricas de uso** (sesiones, pulsaciones, eventos)

El proyecto demuestra el dominio de Kivy, Buildozer, métricas básicas y buenas prácticas de documentación.

A continuación vamos con los pasos realizados.

2. Pasos Realizados para la Paquetización (APK)

A continuación, se detalla el proceso seguido para generar el APK de la aplicación utilizando Buildozer:

2.1 Preparación del entorno (WSL + Ubuntu + dependencias)

Antes de iniciar el proceso de paquetización, fue necesario configurar un entorno Linux usando **WSL (Windows Subsystem for Linux)**, ya que Buildozer requiere herramientas nativas de Linux para compilar aplicaciones Android.

a) Instalación de WSL desde PowerShell

Se abrió *PowerShell como administrador* y se ejecutó el siguiente comando: **wsl --install**

Esto instaló automáticamente los componentes necesarios para ejecutar Linux dentro de Windows, después de la instalación, el sistema solicitó reiniciar la máquina.

b) Verificación de distribuciones disponibles

Luego del reinicio, se utilizó el comando: **wsl.exe --list --online**

Este comando muestra todas las distribuciones Linux disponibles. Entre ellas se seleccionó *Ubuntu versión 24.04*, que es la distribución recomendada para trabajar con Buildozer.

c) Instalación y configuración de Ubuntu en WSL

Se procedió a instalar *Ubuntu* con el comando **wsl --install -d Ubuntu 24.04,** se configura el usuario de Ubuntu con los datos de usuario = user y contraseña = user y se reinicia el sistema.

d) Ingresar a la terminal de Ubuntu

Una vez iniciado el sistema se ingresa a la terminal de *Ubuntu*, desde aquí se ejecutaron todos los comandos necesarios para la paquetización.

2.2 Instalación de Dependencias en Ubuntu

En la terminal de Ubuntu se ejecutaron los siguientes comandos:

- **sudo apt update**
- **sudo apt upgrade -y**
- **sudo apt install python3 python3-pip -y**
- **sudo apt install zip unzip -y**
- **sudo apt install openjdk-17-jdk -y**

2.3 Instalación de Buildozer + Entorno Virtual

a) Crear entorno virtual

Ingresando los siguientes comandos:

- **python3 -m venv venv**
- **source venv/bin/activate**

b) Instalar dependencias adicionales

- **pip install --upgrade pip**
- **pip install cython==0.29.37**
- **pip install buildozer**

c) Instalar paquetes del sistema necesarios para Buildozer

Luego se ingresaron estos comandos todos juntos:

- **sudo apt install -y git python3-setuptools python3-pkg-resources python3-jinja2 libffi-dev libssl-dev libsqlite3-dev liblzma-dev libgstreamer1.0-dev libmtdev-dev zlib1g-dev autoconf automake libtool pkg-config**

2.4 Inicialización y Configuración del Proyecto

Dentro de la carpeta del proyecto, ejecutar: **buildozer init**

Esto genera el archivo **buildozer.spec**.

Dentro del archivo buildozer.spec se editaron estos 3 parámetros:

- **package.name = evaluacion5**
- **package.domain = org.evaluacion**
- **source.dir = app**

2.5 Compilación del APK

Con el entorno ya configurado, se ejecutó: **buildozer -v android debug**

Después de varios minutos el archivo APK generado quedó en:

bin/evaluacion5-0.1-arm64-v8a_armeabi-v7a-debug.apk

Este APK fue probada en un dispositivo Android real y funciona correctamente.

3. Métricas Implementadas y Explicación Técnica

La app incluye un módulo especializado llamado **metrics.py** que está encargado del registro de métricas persistentes.

3.1 Tipos de métricas implementadas

Las métricas que fueron implementadas son las siguientes:

✔ **Número total de presiones de botón**

✔ **Registro de reinicios del contador**

✔ **Duración total de cada sesión de uso**

✔ **Registro cronológico en archivo .log**

✔ **Persistencia en metrics.json**

3.2 Explicación Técnica

Se usa **RotatingFileHandler** para generar logs legibles y seguros.

Cada vez que se presiona el botón, se ejecuta:

- **self.metrics.register_button_press()**

Al iniciar la app:

- **self.metrics.start_session()**

Al cerrarla:

- **self.metrics.end_session()**

metrics.json guarda:

- **total_presses**
- **historial de sesiones**
- **última sesión**

metrics.log guarda todos los eventos importantes con fecha y hora.

3.3 Capturas de metrics.json y metrics.log

metrics.json:


```
{
  "total_presses": 17,
  "sessions": [
    {
      "start": "2025-11-18T11:58:34.373746Z",
      "end": "2025-11-18T11:58:44.650270Z",
      "duration_seconds": 10.28
    },
    {
      "start": "2025-11-18T11:58:56.974059Z",
      "end": "2025-11-18T11:59:09.386759Z",
      "duration_seconds": 12.41
    }
  ],
  "last_session": {
    "start": "2025-11-18T11:58:56.974059Z",
    "end": "2025-11-18T11:59:09.386759Z",
    "duration_seconds": 12.41
  }
}
```


metrics.log


```
2025-11-18 08:58:34,373 - INFO - session_start
2025-11-18 08:58:39,009 - INFO - button_press total_presses=1
2025-11-18 08:58:39,267 - INFO - button_press total_presses=2
2025-11-18 08:58:39,552 - INFO - button_press total_presses=3
2025-11-18 08:58:39,850 - INFO - button_press total_presses=4
2025-11-18 08:58:40,096 - INFO - button_press total_presses=5
2025-11-18 08:58:40,416 - INFO - button_press total_presses=6
2025-11-18 08:58:40,737 - INFO - button_press total_presses=7
2025-11-18 08:58:44,651 - INFO - session_end duration=10.28s
2025-11-18 08:58:56,974 - INFO - session_start
2025-11-18 08:58:58,072 - INFO - button_press total_presses=8
2025-11-18 08:58:58,284 - INFO - button_press total_presses=9
2025-11-18 08:58:58,458 - INFO - button_press total_presses=10
2025-11-18 08:58:58,657 - INFO - button_press total_presses=11
2025-11-18 08:58:58,832 - INFO - button_press total_presses=12
2025-11-18 08:58:59,006 - INFO - button_press total_presses=13
2025-11-18 08:58:59,030 - INFO - button_press total_presses=14
2025-11-18 08:58:59,218 - INFO - button_press total_presses=15
2025-11-18 08:59:03,805 - INFO - button_press total_presses=16
2025-11-18 08:59:04,081 - INFO - button_press total_presses=17
2025-11-18 08:59:09,387 - INFO - session_end duration=12.41s
```


:

4. Captura de Pantalla de la aplicación

5. Resultados Obtenidos

Los resultados finales fueron plenamente satisfactorios:

- ✔ La aplicación Kivy funciona de manera estable y fluida.
- ✔ El sistema de métricas registra múltiples tipos de eventos sin errores.
- ✔ El APK fue compilado exitosamente con Buildozer.
- ✔ La aplicación funciona correctamente en un dispositivo Android real.
- ✔ El repositorio GitHub contiene todos los elementos exigidos:
    - Código ordenado
    - Logs
    - APK
    - README
    - Capturas
    - Informe PDF

6. Reflexión sobre el Uso de Inteligencia Artificial

La Inteligencia Artificial se utilizó como apoyo en:

- Corrección de errores en **Buildozer.**
- Configuración del entorno **WSL.**
- Redacción técnica del **README.**
- Mejora del manejo de **métricas** (estructura modular y persistencia JSON).
- Redacción y estructuración del **informe.**

Se realizaron ajustes basados en las respuestas de la IA, como:

- Separar **metrics.py** en un módulo independiente.
- Implementar **RotatingFileHandler** en lugar de un logger simple.
- Ajustar **buildozer.spec** para que use **source.dir = app.**
- Mejorar la estructura del **README** e incluir descripción técnica.

La IA fue una herramienta de apoyo, pero la implementación final fue validada, corregida y probada manualmente.

7. Enlace al Repositorio GitHub

https://github.com/Juan-Cornejo/Evaluacion-5

8. Conclusión

Este proyecto representó una oportunidad valiosa para comprender a fondo el proceso de desarrollo móvil con Python, desde la creación de una interfaz funcional con Kivy hasta la paquetización real de una aplicación en Android mediante Buildozer. Además, la implementación de métricas permitió explorar cómo registrar y analizar el comportamiento del usuario dentro de una app, integrando conceptos prácticos como logs, persistencia de datos y sesiones. En conjunto, la experiencia fortaleció mi capacidad para resolver problemas, adaptar herramientas nuevas y comprender de forma más integral cómo se construyen aplicaciones móviles completas y medibles en un entorno profesional.