PLANTILLA DE EVALUACIÓN


**EVALUACIÓN**

**HERRAMIENTAS DE PROGRAMACIÓN MOVIL**

**Semana 3**

Michel Brevis

03-11-2024

**Técnico en Análisis y Programación Computacional**

DESARROLLO:

Eres parte de un equipo de desarrollo del Ministerio de Cultura, el cual está en proceso de creación de una nueva aplicación móvil en Android, para poder gestionar las reservas de entradas a los distintos museos del país. Dado lo anterior se han fijado los siguientes requerimientos:

* **El nombre del proyecto debe ser su nombre\_apellido\_fecha\_seccioncurso (la fecha en formato yyyymmdd).**
* **La aplicación debe ser desarrollada en lenguaje Kotlin (aplicativo en lenguaje Java no será considerado).**
* **Cada vez que la aplicación sea abierta o cerrada debe quedar un registro de debug en la consola de desarrollador.**
* **Debe considerar elementos básicos para el ingreso de la reserva (persona responsable, cantidad de personas y fecha).**
* **Debe contener un botón que, al pincharlo, muestre la información de la reserva solicitada (persona responsable, cantidad de personas y fecha) como mensaje de alerta y en la consola del desarrollador.**
* **Debe utilizar layout para el ordenamiento de los controles en pantalla.**

A continuación, responde las siguientes preguntas:

1. **¿Qué elementos de la anatomía de una aplicación móvil son parte de la aplicación desarrollada? Explica.**

Para desarrollar la aplicación en Android, se utilizan varios elementos de la anatomía de una aplicación móvil que ayudan a estructurar y manejar su funcionalidad. Los que destaco en el desarrollo de la aplicación son:

* **Activity**:

Una Activity representa una pantalla en la aplicación. En este caso, podríamos tener una Activity principal que gestione la pantalla de reservas. Cada vez que la aplicación se abra o cierre, se puede registrar un mensaje de debug en la consola de desarrollador desde los métodos onCreate() y onDestroy() de esta Activity.

* **Layouts:**

Para organizar visualmente los controles en la pantalla, se utiliza un layout. Android proporciona varios tipos de layouts como LinearLayout, RelativeLayout, ConstraintLayout, etc. Estos layouts permiten estructurar elementos como el campo para el nombre de la persona responsable, la cantidad de personas, la fecha, y el botón para mostrar la información de reserva.

* **Views:**

Los Views son los elementos visuales de la interfaz de usuario (UI) en Android, como botones, campos de texto, y etiquetas. Aquí, por ejemplo, utilizarás un EditText para la entrada de texto (persona responsable), otro para la cantidad de personas, y probablemente un selector de fecha (DatePicker). El botón de confirmación de reserva sería otro View importante, que al ser presionado muestra la información de reserva.

* **Intents:**

En este proyecto no se menciona específicamente el uso de múltiples pantallas o actividades, pero los Intents se podrían usar para abrir otras partes de la aplicación si fuera necesario, o para enviar datos entre componentes de la misma aplicación en futuras versiones.

* **Logcat:**

La consola de debug en Android se llama Logcat. Esta herramienta permite ver mensajes de registro en tiempo real. En Kotlin, se pueden utilizar las funciones Log.d(), Log.i(), Log.e(), etc., para crear estos registros al abrir y cerrar la aplicación, como lo especifica el requerimiento.

1. **¿Qué layout utilizaste para el ordenamiento de los controles? Justifica tu elección indicando por qué se utilizó este layout sobre los otros disponibles.**

**Para esta aplicación, utilizaría ConstraintLayout como el layout principal para el ordenamiento de los controles en la pantalla, ya que esta herramienta permite:**

* **Flexibilidad en la disposición:**

ConstraintLayout permite alinear y posicionar los elementos en relación a otros controles o al borde de la pantalla, lo que permite una gran flexibilidad y precisión en el diseño. Esto es útil para crear una interfaz limpia y ordenada, independientemente del tamaño de pantalla, ya que se puede hacer que los controles se ajusten automáticamente.

* **Optimización de rendimiento:**

A diferencia de layouts anidados como LinearLayout y RelativeLayout, ConstraintLayout permite agregar y posicionar todos los elementos en una única jerarquía. Esto reduce la profundidad de los elementos de la interfaz de usuario, lo que resulta en un mejor rendimiento en comparación con otros layouts anidados.

* **Facilidad de ajuste responsivo:**

Con ConstraintLayout, es sencillo adaptar la interfaz de usuario para que funcione bien en diferentes tamaños de pantalla y orientaciones. Esto es particularmente importante en una aplicación móvil que puede ser utilizada en dispositivos de distintas resoluciones. Se puede definir el comportamiento de cada control en caso de redimensionamientos, como expandir el campo de texto o centrar el botón al ajustar la pantalla.

* **Posicionamiento preciso y controlado:**

Para los elementos básicos requeridos (nombre de la persona responsable, cantidad de personas, fecha y botón), ConstraintLayout permite que cada uno tenga una ubicación específica en la pantalla, manteniendo una alineación adecuada y sin requerir demasiado código adicional para ajustar posiciones.

1. **¿Qué elementos básicos adecuados consideraste para construir la aplicación sobre la base de los requerimientos? Explica.**

**Para construir la aplicación de reservas de entradas de museos sobre la base de los requerimientos, consideré los siguientes elementos básicos, asegurando que cumplan con cada uno de los puntos mencionados:**

* **Activity principal:**

Creé una MainActivity en Kotlin como la pantalla principal de la aplicación. Esta Activity representa la interfaz donde los usuarios ingresarán los detalles de la reserva. Utilicé los métodos onCreate() y onDestroy() para registrar en la consola cada vez que la aplicación se abre o cierra, usando Log.d() en cada evento, lo cual ayuda a llevar un registro en la consola de desarrollo.

* **EditTexts para capturar información:**

Para que el usuario ingrese los datos de la reserva, se usaron varios campos de texto (EditText):

**-Persona Responsable**: Un EditText para el nombre de la persona responsable de la reserva, permitiendo la entrada de texto.

**-Cantidad de personas**: Un EditText configurado para aceptar solo números, facilitando la entrada de la cantidad de personas que asistirán.

* **DatePicker para la selección de fecha:**

Para simplificar la entrada de fecha de reserva y reducir errores, utilicé un DatePicker. Esto ofrece una forma intuitiva para que el usuario seleccione la fecha deseada sin necesidad de ingresar manualmente el formato, garantizando que sea consistente.

* **Button para mostrar la información de la reserva:**

Incluí un botón (Button) que, al ser presionado, toma los datos de los campos anteriores y muestra un mensaje de alerta (diálogo) con la información ingresada, utilizando un AlertDialog.Builder para mostrar los detalles de la reserva.

Ademas, envía la misma información a la consola de desarrollador usando Log.d(), facilitando el registro de cada acción en tiempo real.

* **ConstraintLayout para la disposición de los controles:**

Como mencioné antes, elegí ConstraintLayout para organizar todos los elementos de la pantalla de forma estructurada y adaptable. Esto permite mantener una interfaz limpia y bien alineada, compatible con distintas pantallas y orientaciones de dispositivos.

* **Mensajes de error y validación básica:**

Para mejorar la experiencia del usuario, consideré validaciones básicas en los campos de entrada. Por ejemplo:

*-Verificar que el nombre y la cantidad de personas no estén vacíos antes de permitir el envío de la información.*

*-Asegurar que la cantidad de personas sea un valor numérico positivo.*

*-Mostrar mensajes de error si los campos no están correctamente llenados, ayudando al usuario a corregir cualquier error antes de completar la reserva.*

REFERENCIAS BIBLIOGRÁFICAS:

* **Ejemplo texto de lectura de IACC:**

IACC. (2024). *Herramientas de programación movil*

Semana 3