iacc instituto profesional

EVALUACIÓN

HERRAMIENTAS DE PROGRAMACIÓN MOVIL
Semana 6

Michel Brevis
25-11-2024
Técnico en Análisis y Programación Computacional

DESARROLLO:

Imagina que has sido contratado como desarrollador de aplicaciones móviles por la empresa Control Total. Esta entrega el servicio de control de asistencia de los trabajadores, por lo cual quieren desarrollar una nueva aplicación en Android. Para efectuar esto 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).
- La aplicación debe registrar los siguientes datos de manera persistente: RUT, nombre y apellido del trabajador, fecha y hora, además del tipo de marca (entrada o salida).
- Cuando el trabajador registre su asistencia, deberá mostrar un mensaje de confirmación que la marca de asistencia fue guardada de forma correcta.
- La aplicación debe listar los registros de asistencia de los trabajadores.
- La aplicación debe ser capaz de compartir parte de los datos de asistencia (RUT, fecha y hora, y si es entrada o solida).

Dado este contexto, a continuación, responde lo siguiente:

1. **¿Qué elementos del manejo de datos utilizarías para generar la persistencia de datos solicitada? Justifica tu elección con enfoque en el uso de las bases de datos SQLite y/o Realm según corresponda.**

Para garantizar la persistencia de datos en la aplicación que estamos planeando desarrollar, utilizaría SQLite como el sistema de gestión de bases de datos.

Tablas estructuradas:

Se creará una tabla llamada attendance para almacenar los datos solicitados: RUT, nombre, apellido, fecha y hora, y tipo de marca (entrada o salida). Cada columna de la tabla corresponde a un dato específico.

Consultas SQL:

Usaremos sentencias INSERT para guardar nuevos registros, SELECT para recuperar los registros almacenados y, si es necesario, UPDATE o DELETE para editar o eliminar datos.

Base de datos local:

SQLite permite que los datos se guarden directamente en el almacenamiento interno del dispositivo en un archivo .db. Esto asegura que los datos no se pierdan al cerrar la aplicación o reiniciar el dispositivo.

Integración con Android:

Utilizaré la clase SQLiteOpenHelper, que facilita la creación y manejo de la base de datos. Esta clase permite definir las operaciones iniciales de la base de datos y gestionar actualizaciones de su estructura.

La justificación del porque el uso de SQL radica en:

Persistencia requerida:

SQLite es perfecto para aplicaciones móviles que necesitan almacenar datos localmente de forma estructurada. Esto se ajusta a la necesidad de guardar los registros de asistencia de los trabajadores.

Eficiencia y simplicidad:

SQLite es ligero y eficiente, lo que lo hace adecuado para aplicaciones con un volumen moderado de datos como la nuestra. No requiere configuración adicional ni servidores externos.

Compatibilidad nativa:

Está integrado en Android, por lo que no necesitamos instalar librerías o dependencias adicionales. Esto simplifica el desarrollo y asegura un mejor rendimiento.

Adecuación a los requerimientos:

SQLite permite realizar operaciones como insertar, consultar y listar datos fácilmente mediante consultas SQL. Estas funcionalidades son fundamentales para registrar y mostrar los datos de asistencia.

2. **Utiliza un gestor de base de datos de acuerdo con los requerimientos planteados y describe su funcionamiento en la aplicación desarrollada.**

Como se dijo anteriormente, para los requerimientos planteados, se utilizó **SQLite** como el gestor de base de datos debido a su eficiencia, simplicidad y capacidad de manejar datos estructurados de manera local en el dispositivo Android. A continuación, describo su funcionamiento en la aplicación desarrollada:

Funcionamiento de SQLite en la aplicación

Estructura de la base de datos:

Se creó una tabla llamada attendance con columnas que representan los datos solicitados:

- RUT: Identificación del trabajador.
- Nombre y Apellido: Información personal.
- Fecha y Hora: Registro automático de cuándo se realiza la marca.
- Tipo de marca: Entrada o salida.

Cada registro es almacenado como una fila en esta tabla, asegurando que los datos estén organizados y sean fácilmente recuperables.

Operaciones básicas:

Insertar registros: Cada vez que un trabajador marca su asistencia, la aplicación guarda sus datos en la tabla utilizando una consulta SQL. Esto asegura que los registros sean persistentes.

Consultar datos: Para mostrar la lista de registros, la aplicación realiza una consulta que obtiene toda la información almacenada en la tabla. Los datos recuperados son formateados y mostrados en pantalla.

Archivo de base de datos:

SQLite guarda la base de datos en un archivo local dentro del sistema de archivos de Android. Este archivo está accesible solo para la aplicación, garantizando la seguridad y privacidad de los datos.

Flujo de trabajo dentro de la aplicación:

Registro de datos: Cuando el usuario completa el formulario de asistencia, los datos se validan y se insertan en la tabla attendance mediante una consulta SQL.

Visualización de registros: Al seleccionar la opción de listar registros, la aplicación consulta todos los datos de la tabla y los muestra de forma organizada en pantalla.

Confirmación de éxito: Después de guardar un registro, la aplicación muestra una notificación para confirmar que la operación se realizó correctamente.

SQLite actuó como el gestor de base de datos principal para garantizar la persistencia y organización de los registros de asistencia. Su integración con Android y su eficiencia en el manejo de datos estructurados lo convirtieron en la opción ideal para cumplir con los requerimientos del proyecto.

3.Construye el proveedor de contenidos de acuerdo con los requerimientos planteados y describe los pasos para su construcción.

En el caso de nuestra aplicación, este componente permitiría a otras aplicaciones acceder a los datos de asistencia registrados (como RUT, fecha, hora y tipo de marca)

Pasos para construir el proveedor de contenidos

1. Definir el propósito del proveedor

El proveedor de contenidos debe permitir que otras aplicaciones accedan a los datos específicos de asistencia que almacenamos en la base de datos SQLite:

- Datos compartidos: RUT, fecha y hora, tipo de marca (entrada/salida).
- Datos no compartidos: Información sensible como nombres o datos no requeridos.

2. Crear la clase del proveedor

Se necesita una clase que extienda ContentProvider. Esta clase actúa como intermediaria entre la base de datos SQLite de la aplicación y las solicitudes externas. La clase debe implementar métodos básicos:

- query: Recupera datos de la base de datos.
- insert: Permite agregar nuevos registros.
- update y delete (opcionales): Para editar o eliminar registros, si es necesario.

3. Definir el URI del proveedor

Cada proveedor de contenidos tiene un URI único, que es como una dirección web que las aplicaciones usan para acceder a los datos. El URI de nuestro proveedor podría ser algo como:

content://com.example.michel_brevis.attendance/attendance

4. Registrar el proveedor en AndroidManifest.xml

El proveedor debe ser declarado en el archivo AndroidManifest.xml con su URI único. También podemos establecer permisos para limitar el acceso a los datos, como:

- **<permission>**: Para garantizar que solo ciertas aplicaciones puedan usar el proveedor.
- **<exported>**: Para definir si el proveedor será accesible desde fuera de la aplicación.

5. Configurar permisos de seguridad

Para proteger los datos, debemos controlar quién puede acceder al proveedor. Por ejemplo:

- Requiriendo permisos explícitos para aplicaciones externas.
- Proporcionando solo los datos necesarios para las consultas externas.

6. Implementar la lógica en los métodos principales

En la clase del proveedor, implementaríamos la lógica para:

- **query:** Consultar los datos relevantes de la tabla attendance según el URI solicitado.
- **insert:** Agregar nuevos registros (si el proveedor permite escribir datos).
- **update y delete (opcionales):** Si se necesita modificar o eliminar registros.

Lamentablemente, no me fue posible incluir esta función a mi aplicación debido a un error que no pude resolver.

REFERENCIAS BIBLIOGRÁFICAS:

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

IACC. (2024). *Herramientas de programación movil*
Semana 6