SANTO®
TOMÁS

INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA

Almacenes

Nombre Alumno: Kevin Andres Sandoval Mosquera
Profesor: Manuel Merino Piceros
Asignatura: Programación Android
Carrera: Ingeniería en Informática
Fecha: 05-11-2024

Introducción

El proyecto ser basa sobre la gestión eficiente de inventarios es fundamental para el éxito de cualquier establecimiento que maneje productos físicos. Con esto se necesitan herramientas que faciliten el registro y seguimiento de los productos en almacenes. se presenta el desarrollo de una aplicación móvil en Android Studio diseñada específicamente para optimizar el proceso de registro de productos en el almacén.

La aplicación permitirá a los usuarios registrar de manera ágil y precisa la entrada y salida de productos, utilizando interfaces intuitivas y funcionalidades que simplifican las tareas diarias. A través de esta herramienta, se busca reducir errores humanos aumentando significativamente en la eficiencia operativa.

La importancia y relevancia de este proyecto radican en su capacidad para agilizar la gestión del inventario. Al implementar una solución digital, Con esto espera no solo mejorar la productividad del personal, sino también ofrecer datos en tiempo real que faciliten la toma de decisiones. Contar con un sistema eficaz para el manejo de inventarios se convierte en un factor clave para garantizar un servicio al cliente de alta calidad.

Objetivos de la Aplicación

1. Qué Problema Resuelve la Aplicación

La aplicación de gestión de inventario en almacenes registrar y gestionar productos de forma digital ya que manualmente puede llevar a errores en el inventario, pérdidas de productos y dificultades para realizar un seguimiento efectivo de las existencias.

2. Beneficios Esperados para los Usuarios

- **Facilidad de Uso:** La interfaz intuitiva permite a los usuarios gestionar productos sin complicaciones, reduciendo el tiempo necesario para realizar tareas administrativas.

- **Acceso Rápido a Información:** Los usuarios pueden visualizar rápidamente el inventario disponible, lo que les ayuda a tomar decisiones informadas sobre compras y ventas.

- **Mejora en la Organización:** Al permitir la actualización y eliminación de productos fácilmente, la aplicación ayuda a mantener un inventario limpio y organizado.

- **Ahorro de Tiempo:** La automatización de procesos como la adición y eliminación de productos reduce el tiempo dedicado a la gestión manual del inventario.

3. Metas Concretas que se Esperan Alcanzar con la Aplicación

- **Implementación Completa del CRUD:** Asegurar que todas las funcionalidades estén completamente operativas y sean fáciles de usar.

- **-Comodidad del Usuario:** Alcanzar un sistema intuitivo y amigable para el usuario pueda usar con facilidad.

Descripción de la Aplicación

La aplicación es un sistema de gestión de productos diseñado para facilitar el registro, actualización y eliminación de productos en un almacén. Permite a los usuarios ingresar información sobre productos, incluyendo su nombre y cantidad, y gestionar esta información de manera eficiente a través de una interfaz intuitiva. La aplicación utiliza una base de datos SQLite para almacenar los datos de los productos, lo que permite un acceso rápido y eficiente a la información.

1. Funcionalidades Clave de la Aplicación

- **Agregar Productos:** Los usuarios pueden ingresar el nombre y la cantidad de un producto nuevo. Esta información se almacena en una base de datos local.

- **Visualizar Productos:** La aplicación muestra una lista de todos los productos registrados, permitiendo a los usuarios ver rápidamente qué productos están disponibles y en qué cantidades.

- **Editar Productos:** Los usuarios pueden seleccionar un producto existente para editar su nombre o cantidad. Esto facilita la actualización de la información sin necesidad de eliminar y volver a agregar el producto.

- **Eliminar Productos:** La aplicación permite a los usuarios eliminar productos que ya no son necesarios, manteniendo así la base de datos actualizada y organizada.

- **Interfaz Intuitiva:** La aplicación cuenta con una interfaz gráfica amigable que facilita la navegación y el uso por parte del usuario.

2. Descripción del Diseño y Navegación de la Aplicación

El diseño de la aplicación se basa en un enfoque simple y funcional, con el objetivo de proporcionar una experiencia de usuario fluida.

- **Pantalla Principal:** Al abrir la aplicación, los usuarios son recibidos por una pantalla principal que incluye campos para ingresar el nombre del producto y su cantidad, junto con botones para agregar, editar y eliminar productos.

- **ListView:** Debajo de los campos de entrada, se encuentra un `ListView` que muestra todos los productos registrados. Los usuarios pueden tocar cualquier elemento en esta lista para seleccionarlo para edición o eliminación.

- **Edición Oculta:** El campo para editar un producto se encuentra oculto por defecto y se muestra solo cuando el usuario selecciona un producto en la lista. Esto mantiene la interfaz limpia y evita confusiones.

- **Interacción Clara:** Cada botón tiene una función clara (Agregar, Editar, Eliminar), lo que permite a los usuarios entender fácilmente cómo interactuar con la aplicación.

3. Breve Explicación de las Herramientas y Tecnologías Empleadas en el Desarrollo

- **Android Studio:** Es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Proporciona herramientas completas para diseñar interfaces gráficas, escribir código, depurar aplicaciones y gestionar bases de datos.

- **Java:** El lenguaje principal utilizado para desarrollar la lógica detrás de la aplicación. Java es ampliamente utilizado en el desarrollo Android debido a su robustez y facilidad para manejar operaciones complejas.

- **SQLite:** Se utiliza como sistema de gestión de bases de datos local. SQLite es ligero y fácil de usar, lo que lo convierte en una excelente opción para aplicaciones móviles que requieren almacenamiento persistente.

- **XML:** Se utiliza para definir el diseño visual (layouts) de la interfaz gráfica. XML permite separar claramente la lógica del negocio del diseño visual, facilitando así el mantenimiento del código.

- **Material Design:** Aunque no se menciona explícitamente en el código proporcionado, se puede utilizar como guía para mejorar la estética visual y la usabilidad general de la aplicación.

Explicación de las Nuevas Funcionalidades que la Aplicación Contiene (CRUD)

La aplicación de gestión de productos ha incorporado funcionalidades CRUD (Crear, Leer, Actualizar, Eliminar) que permiten a los usuarios interactuar eficientemente con los datos de los productos. Estas operaciones son fundamentales para cualquier aplicación que maneje información y son esenciales para mejorar la experiencia del usuario.

1. Nuevas Características Añadidas Durante el Desarrollo

Crear (Create):

- Los usuarios pueden agregar nuevos productos ingresando su nombre y cantidad. Esta funcionalidad permite que la base de datos se actualice con información relevante y actual.

Leer (Read):

- La aplicación muestra una lista de todos los productos almacenados en la base de datos. Los usuarios pueden ver fácilmente qué productos están disponibles y en qué cantidades, lo que facilita la gestión del inventario.

Actualizar (Update):

- -Los usuarios pueden seleccionar un producto existente para editar su nombre o cantidad. Esta característica permite realizar cambios sin necesidad de eliminar y volver a agregar el producto, lo que ahorra tiempo y esfuerzo.

Eliminar (Delete):

- La aplicación permite a los usuarios eliminar productos que ya no son necesarios. Esto ayuda a mantener la base de datos organizada y relevante, eliminando información obsoleta.

2. Cómo Estas Funcionalidades Mejoran la Experiencia del Usuario

- **Facilidad de Uso:** Al proporcionar una interfaz intuitiva para realizar operaciones CRUD, los usuarios pueden gestionar sus productos sin complicaciones.

- **Acceso Rápido a Información:** La capacidad de leer datos rápidamente permite a los usuarios obtener información sobre el inventario sin tener que navegar por múltiples pantallas o formularios complejos.

- **Eficiencia en la Gestión:** Las funciones de actualización y eliminación permiten a los usuarios modificar o eliminar información sin necesidad de pasos adicionales, lo que agiliza el proceso de gestión de datos.

- **Organización**: La capacidad de eliminar productos innecesarios ayuda a mantener una base de datos limpia y organizada, lo cual es crucial para cualquier sistema que maneje información dinámica.

Demostración de conexión a la base de datos

1. **Explicar cómo la aplicación se conecta a la base de datos**

La aplicación utiliza SQLite como sistema de gestión de bases de datos local. La conexión a la base de datos se establece mediante la clase SQLiteOpenHelper, que proporciona métodos para crear, abrir y gestionar la base de datos.

- **Creación de la Clase Database:** Se crea una clase Database que extiende SQLiteOpenHelper. Esta clase es responsable de manejar la creación y actualización de la base de datos.

```
public class Database extends SQLiteOpenHelper {  2 usages
    private static final String DATABASE_NAME="tareas.db";  1 usage

    private static final String Table_Name="task";  6 usages

    private static final String Col_1= "ID";  no usages

    private static final String Col_2= "task";  2 usages
```

- **Constructor:** En el constructor de la clase Database, se llama al constructor de SQLiteOpenHelper con el nombre de la base de datos y su versión.

```
public Database(Context context) {  1 usage
    super(context, DATABASE_NAME,  factory: null,  version: 1);
}
```

- **Método onCreate:** Este método se invoca cuando la base de datos es creada por primera vez. Aquí se definen las tablas y las columnas que se utilizarán en la base de datos.

```
@Override  1 usage
public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + Table_Name + "(ID INTEGER PRIMARY KEY AUTOINCREMENT, TASK TEXT)");
}
```

- **Método onUpgrade:** Este método se llama cuando la versión de la base de datos cambia. Permite realizar cambios en la estructura de la base de datos, como agregar nuevas tablas o modificar las existentes.

```
@Override  no usages
public void onUpgrade(SQLiteDatabase db, int i, int i1) {
    db.execSQL("DROP TABLE IF EXISTS " + Table_Name);
    onCreate(db);
}
```

- **Apertura y Cierre:** Cuando se realizar operaciones sobre la base de datos, se abre una conexión utilizando getWritableDatabase() o getReadableDatabase(), dependiendo si se necesita escribir o leer datos.

```
public boolean InsertTask(String task){  1 usage
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(Col_2, task);
    long result = db.insert(Table_Name,  nullColumnHack: null,contentValues);
    return result != -1;
}
public ArrayList<String> getAllTask()  2 usages
{
    ArrayList<String> task = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor res = db.rawQuery( sql: "SELECT * FROM " + Table_Name,  selectionArgs: null);
    if (res.moveToFirst()){
        do{
            task.add(res.getString( i: 1));
        }while(res.moveToNext());
    }
    return task;
```

2. Ejemplos de operaciones CRUD

- **Create:** Añade a la base de datos los datos ingresados.

```
public boolean InsertTask(String task){  1 usage
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(Col_2, task);
    long result = db.insert(Table_Name,  nullColumnHack: null,contentValues);
    return result != -1;
}
```

- **Read:** Muestra en la list los datos de la base de datos.

```
public ArrayList<String> getAllTask()  2 usages
{
    ArrayList<String> task = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor res = db.rawQuery( sql: "SELECT * FROM " + Table_Name,  selectionArgs: null);
    if (res.moveToFirst()){
        do{
            task.add(res.getString( i: 1));
        }while(res.moveToNext());
    }
    return task;
}
```

- **Update:** Actualiza datos.

```
public void updateTask(String oldTask, String newTask) {  1 usage
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(Col_2, newTask);
    db.update(Table_Name, contentValues,  whereClause: "TASK = ?", new String[]{oldTask});
}
```

- **Delete:** Elimina datos.

```
public void deleteTask(String task) {  1 usage
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(Table_Name,  whereClause: "TASK = ?", new String[]{task});
}
```

3. **Descripción de cómo se manejaron posibles errores de conexión**

Aquí hay algunas estrategias implementadas para manejar posibles errores:

- **Verificación del Estado:** Antes de realizar operaciones críticas, como insertar o actualizar registros, es útil verificar si la conexión a la base de datos está activa. Esto puede hacerse mediante métodos que validen el estado del objeto SQLiteDatabase.

- **Mensajes al Usuario:** En caso de un error durante las operaciones CRUD, es recomendable mostrar mensajes claros al usuario indicando el problema y sugiriendo acciones a seguir (por ejemplo, intentar nuevamente más tarde).

- **Cierre del Cursor y Base de Datos:** Asegurarse siempre de cerrar los cursores y conexiones a la base de datos después del uso es fundamental para evitar fugas de memoria y mantener un rendimiento óptimo.

Elementos Finales para que la Aplicación Quede Terminada

1. Descripción de las Pruebas que Aún Deben Realizarse

- **Pruebas Funcionales:** Verificar que todas las funcionalidades CRUD (Crear, Leer, Actualizar, Eliminar) funcionen correctamente. Esto incluye agregar productos, editar sus detalles, eliminarlos y asegurarse de que la lista se actualice adecuadamente.

- **Pruebas de Usabilidad:** Evaluar la experiencia del usuario al interactuar con la aplicación.

- **Pruebas de Seguridad:** Evaluar la seguridad de los datos almacenados en la base de datos local y asegurarse de que no haya vulnerabilidades que puedan ser explotadas por usuarios malintencionados.

2. Aspectos por Optimizar o Corregir Antes de la Finalización

- **Optimización del Código:** Revisar el código para eliminar redundancias y mejorar la legibilidad.

- **Manejo de Errores:** Mejorar el manejo de errores en las operaciones CRUD. Asegurarse de que se proporcionen mensajes claros al usuario en caso de fallas y que se registren errores para su posterior análisis.

- **Interfaz Gráfica:** Revisar el diseño visual para asegurar que sea atractivo y fácil de usar. Considerar implementar principios de Material Design para mejorar la estética y usabilidad.

- **Rendimiento de la Base de Datos:** Evaluar consultas SQL para asegurarse de que sean eficientes. Si es necesario, optimizar las consultas o considerar el uso de índices en las tablas.

- **Documentación:** Completar toda la documentación técnica relacionada con el código, así como crear un manual del usuario que explique cómo utilizar la aplicación.

3. Planificación de las Últimas Etapas del Desarrollo

Revisión Final del Código:

- Realizar una revisión del código para identificar áreas que necesiten mejoras y/o correcciones.
- Implementar cambios basados en las revisiones.

Realización de Pruebas:

- Ejecutar todas las pruebas mencionadas anteriormente.
- Documentar los resultados y realizar ajustes según sea necesario.

Optimización Final:

- Implementar optimizaciones basadas en los resultados de las pruebas.
- Asegurarse de que todas las funcionalidades estén funcionando correctamente después de los cambios.

Documentación:

- Completar toda la documentación necesarios en el código.

Despliegue:

- Realizar pruebas finales antes de la entrega final.

Conclusión y Cierre

Reflexión sobre el Proceso de Desarrollo

El desarrollo de esta aplicación de gestión de productos ha sido muy buena experiencia que me ha ayudado a comprender más sobre conexiones a base de datos y programación java. He aprendido a implementar un sistema CRUD (Crear, Leer, Actualizar, Eliminar) y a trabajar con bases de datos (SQLite) en Android Studio.

Reconocimientos y Agradecimientos

Agradezco al profesor Manuel Merino por su apoyo y orientación durante el proyecto. También agradezco a mis compañeros por su valiosa retroalimentación.