SANTO®
TOMÁS

Informe App BarberReserver

Carrera: Ingeniería En Informática
Asignatura: Programación Android
Docente: Manuel Merino Piceros
Alumno: Rubén Torres Arias
Fecha Entrega: 05-11-2024

Índice

Introducción

Breve contexto y descripción del proyecto.

Se dice que el sector laboral menos perjudicado con la llegada de la Inteligencia Artificial y la robótica será el de la barbería y peluquería. Anticipando un crecimiento del sector de la barbería, surge la necesidad de digitalizar ciertos procesos que hasta ahora se realizaban de manera manual, como la gestión de clientes, el seguimiento de pagos, y el registro de citas. La aplicación BarberReserver nace con el objetivo de ayudar a los barberos a gestionar las citas y seguimiento con sus clientes y su ganancia monetaria de manera eficiente, sencilla y accesible desde su dispositivo móvil. BarberReserver permite a los barberos mantener un historial detallado de los clientes, registrar sus preferencias, tomar notas sobre sus estilos de corte, y llevar un control financiero de sus transacciones, todo en un solo lugar.

Importancia y relevancia del proyecto.

La digitalización en el sector de los servicios personales, como la barbería, permite optimizar el tiempo de los profesionales y mejorar su organización. La gestión manual de los clientes puede ser tediosa y está sujeta a errores o pérdidas de información. BarberReserver busca resolver estos problemas proporcionando una plataforma con funcionalidades como la generación de reportes financieros en PDF o Excel. Los barberos pueden optimizar sus procesos de gestión y centrarse en lo más importante: sus clientes. Dado que la app es una solución local, no requiere conexión constante a internet, lo que hace a la app accesible y económica para pequeños emprendedores del rubro.

Objetivos de la aplicación

¿Qué problema resuelve la aplicación?

La aplicación BarberReserver resuelve el problema de la gestión ineficiente de los clientes en barberías. Muchos barberos manejan manualmente sus citas y detalles de clientes, lo que puede llevar a pérdida de información o falta de seguimiento de los servicios realizados. Además, el control de pagos y medios de pago suele ser desorganizado. Esta app facilita:
Organización de clientes: Permite almacenar detalles importantes de los clientes, como su historial de cortes de cabello, imágenes de estilos anteriores, y notas sobre sus preferencias.
Control de citas: Registro sencillo de citas, con la posibilidad de asignar fechas, servicios y medios de pago.
Gestión financiera: Los barberos pueden llevar un control de los ingresos generados y generar reportes financieros en formato PDF.

Beneficios esperados para los usuarios

Ahorro de tiempo: La aplicación simplifica el proceso de gestionar citas y clientes, lo que permite al barbero concentrarse en su trabajo en lugar de preocuparse por la administración.
Mejora en la atención al cliente: Al tener acceso al historial de cada cliente, el barbero puede ofrecer un servicio más personalizado, asegurando que las preferencias del cliente se respeten en cada visita.
Organización financiera: Con la capacidad de registrar medios de pago y generar reportes diarios, semanales o mensuales, de acuerdo a las preferencias del barbero, este puede tener un control más claro sobre sus ingresos y mejorar su planificación financiera.

Metas concretas que se esperan alcanzar con la aplicación

Facilitar la gestión de citas: Los barberos podrán registrar y consultar fácilmente todas sus citas, siendo capaces de anticipar el día con información detallada sobre los clientes y los servicios a realizar.
Mejorar la organización del barbero: La aplicación ayudará a organizar la información de los clientes, evitando la pérdida de detalles importantes y mejorando la calidad del servicio.
Control financiero efectivo: Al generar reportes financieros en PDF, el barbero podrá compartir sus resultados diarios de manera sencilla, por ejemplo, enviando estos reportes a su jefe o contable o a sus colegas o subordinados.

Adaptabilidad al uso local: La aplicación se ejecuta de manera totalmente local, sin necesidad de conexión a internet ni almacenamiento en la nube, garantizando su accesibilidad y bajo costo para barberos independientes.

Descripción de la aplicación

Funcionalidades clave de la aplicación

Registro de clientes: El barbero puede registrar nuevos clientes, guardando información como nombre, imagen del corte de cabello anterior, medio de pago preferido, y notas adicionales sobre preferencias de estilo.

Gestión de citas: La app permite agendar citas de manera rápida y eficiente, con opciones para seleccionar el tipo de servicio, fecha y hora, y el medio de pago que se utilizará.

Historial del cliente: El barbero puede acceder al historial de cada cliente, donde se registran los servicios previos y las observaciones que el barbero ha hecho.

Generación de reportes: Los barberos pueden generar un reporte en formato PDF que incluye todos los pagos del día, con detalles sobre los clientes y servicios realizados. Este reporte puede ser compartido fácilmente por correo electrónico o WhatsApp.

Control de servicios y pagos: Se incluye un seguimiento de los servicios realizados y los pagos asociados, permitiendo al barbero llevar un registro claro de su flujo de caja.

Descripción del diseño y navegación de la aplicación

La aplicación está diseñada para ser intuitiva y fácil de usar. Primero, en el Login, el usuario se registra y se limpian los campos automáticamente. Luego se vuelven a escribir las credenciales y el usuario ingresa. La pantalla principal es un dashboard donde el barbero puede ver sus citas del día de manera organizada. Desde aquí, se puede acceder a opciones para añadir nuevos clientes o registrar citas. La navegación se realiza mediante botones y menús simples que permiten moverse fácil y rápidamente entre las funcionalidades principales. La interfaz está dividida en secciones claras: Clientes, Citas, y Reportes. Cada una de estas secciones tiene una interfaz dedicada para gestionar la información relevante, garantizando que todo el flujo de trabajo sea eficiente y sin complicaciones. Por ahora, sólo se tiene la interfaz de clientes con su respectivo CRUD.

Breve explicación de las herramientas y tecnologías empleadas en el desarrollo

Lenguaje de programación: La aplicación está desarrollada en Java, utilizando Android Studio como entorno de desarrollo integrado (IDE).

Base de datos local: La información se almacena de manera local en el dispositivo mediante SQLite, la base de datos embebida en Android. Para facilitar el acceso y gestión de los datos, se utiliza la biblioteca Room, que proporciona una capa de abstracción sobre SQLite y permite el manejo de datos a través de objetos Java, lo que simplifica la interacción con la base de datos.

Generación de PDF: La generación de reportes en formato PDF se realizará utilizando la biblioteca iText, que permite a los barberos crear documentos PDF con los detalles de las transacciones del día y enviarlos por correo o WhatsApp.

Diseño de interfaz: La interfaz de usuario sigue las pautas de Material Design, lo que garantiza una experiencia visual coherente y moderna para los usuarios. Se prioriza la simplicidad y funcionalidad, permitiendo que los barberos accedan a la información de manera rápida y eficiente.

Explicación de las nuevas funcionalidades que la aplicación contiene

Nuevas características añadidas durante el desarrollo

Selector de fecha y hora: Se agregó un calendario interactivo para la selección de fecha y hora de las citas, lo que simplifica y agiliza el proceso de programación.

CRUD de citas: Se incluyó la capacidad de crear, ver, editar y eliminar citas previamente registradas. Esto permite a los barberos almacenar y visualizar fácilmente los clientes del día, además de permitirles modificar detalles en caso de cambios de última hora o cancelar citas si es necesario.

Pulsación larga para eliminar citas: Los usuarios pueden eliminar citas con una pulsación larga en el ListView, lo que hace el proceso más intuitivo.

Cómo estas funcionalidades mejoran la experiencia del usuario

El selector de fecha y hora facilita la entrada de datos y reduce errores al programar citas, lo que asegura un manejo más preciso de la agenda.

La opción de editar citas brindan al usuario la posibilidad de ajustar detalles sin necesidad de cancelar la cita completa, mejorando la eficiencia y la organización.

La pulsación larga para eliminar simplifica el flujo de trabajo y permite una navegación más fluida.

La gestión del estado de las citas ayuda a los barberos a llevar un seguimiento claro de las citas realizadas y las pendientes, mejorando la planificación y la calidad del servicio al cliente.

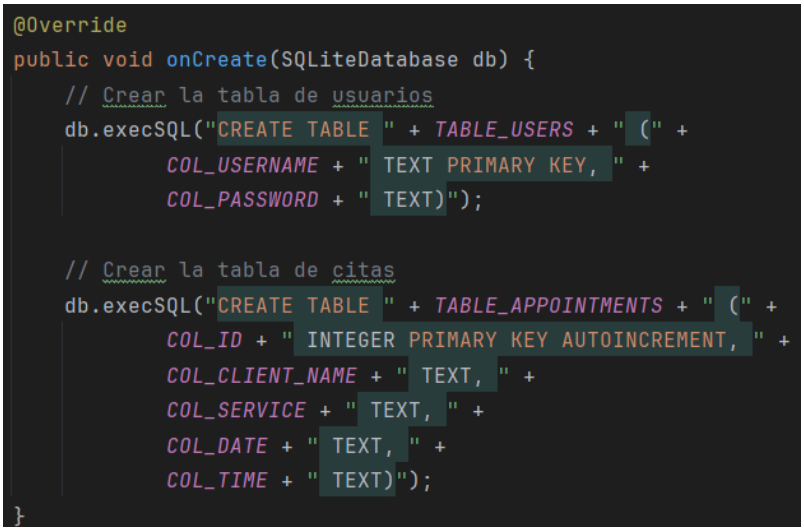
Demostración de conexión a la base de datos

Explicar cómo la aplicación se conecta a la base de datos

La aplicación BarberReserver se conecta a una base de datos local mediante SQLite, la base de datos embebida en Android. Esta base de datos se utiliza para almacenar información de usuarios, clientes, citas y servicios. La conexión a la base de datos se maneja a través de la clase DatabaseHelper, que extiende SQLiteOpenHelper y proporciona métodos para crear, leer, actualizar y eliminar registros en las tablas de clientes y usuarios.

Cada vez que se interactúa con la base de datos (por ejemplo, para agregar una cita), se abre una conexión y se ejecuta una consulta SQL para realizar la operación requerida.

En las siguientes imágenes podemos ver la creación de las tablas de la base de datos y un ejemplo donde se obtienen datos subidos a la base de datos, respectivamente.



```
@Override
public void onCreate(SQLiteDatabase db) {
    // Crear la tabla de usuarios
    db.execSQL("CREATE TABLE " + TABLE_USERS + " (" +
            COL_USERNAME + " TEXT PRIMARY KEY, " +
            COL_PASSWORD + " TEXT)");

    // Crear la tabla de citas
    db.execSQL("CREATE TABLE " + TABLE_APPOINTMENTS + " (" +
            COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            COL_CLIENT_NAME + " TEXT, " +
            COL_SERVICE + " TEXT, " +
            COL_DATE + " TEXT, " +
            COL_TIME + " TEXT)");
}
```

```
// Cargar citas desde la base de datos y actualizar el ListView
private void loadAppointments() {  4 usages
    appointmentList = new ArrayList<>();
    Cursor cursor = db.getAllAppointments();
    if (cursor.getCount() == 0) {
        Toast.makeText( context: this,  text: "No hay citas registradas", Toast.LENGTH_SHORT).show();
    } else {
        while (cursor.moveToNext()) {
            // Formato: ID - Cliente - Servicio - Fecha - Hora
            appointmentList.add(cursor.getString( i: 0) + " - " + cursor.getString( i: 1) + " - " + cursor.get
        }
    }

    // Vincular el ListView con el adaptador
    adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, appointmentList);
    listViewAppointments.setAdapter(adapter);
```

Ejemplos de operaciones CRUD

La aplicación realiza las siguientes operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre la base de datos:

Crear: Cuando se registra una nueva cita, se inserta un nuevo registro en la tabla appointments. El método insertAppointment() en DatabaseHelper maneja esta operación.

Leer: La carga de todas las citas desde la base de datos se realiza con el método getAllAppointments(), que devuelve un cursor con los registros almacenados.

Actualizar: Las citas se pueden actualizar mediante el método updateAppointment(), que modifica los datos existentes en función del ID de la cita.

Eliminar: La eliminación de citas se realiza con el método deleteAppointment(), que borra un registro según su ID.

Descripción de cómo se manejaron posibles errores de conexión

Al ser una aplicación local, no existen problemas de conectividad a internet. Sin embargo, se implementaron mecanismos de control de errores para evitar fallos durante las operaciones de lectura y escritura en la base de datos. Por ejemplo, si una operación de inserción o actualización falla, se captura una excepción y se muestra un mensaje de error al usuario a través de un Toast, informándole del problema y solicitando que vuelva a intentarlo. Esto asegura una experiencia más robusta y a prueba de fallos. En la siguiente imagen podemos ver un ejemplo donde hay algunos manejos de errores utilizando simplemente un flujo de control de decisión con if-else.

```java
// Agregar una nueva cita
private void addAppointment() {  1 usage
    String clientName = etClientName.getText().toString();
    String service = spinnerService.getSelectedItem().toString();
    String date = etDate.getText().toString();
    String time = etTime.getText().toString();

    if (clientName.isEmpty() || date.isEmpty() || time.isEmpty()) {
        Toast.makeText( context: this,  text: "Por favor ingrese todos los campos", Toast.LENGTH_SHORT).show();
        return;
    }

    boolean isInserted = db.insertAppointment(clientName, service, date, time);
    if (isInserted) {
        Toast.makeText( context: this,  text: "Cita agregada exitosamente", Toast.LENGTH_SHORT).show();
        clearFields();
        loadAppointments(); // Recargar el ListView después de agregar
    } else {
        Toast.makeText( context: this,  text: "Error al agregar la cita", Toast.LENGTH_SHORT).show();
    }
}
```

Elementos finales para que la aplicación quede terminada

Descripción de las pruebas que aún deben realizarse

Pruebas funcionales o unitarias: Asegurarse de que todas las funcionalidades, como el registro, la edición y eliminación de citas, funcionen correctamente en diferentes dispositivos y versiones de Android.
Pruebas de interfaz de usuario (UI): Verificar que la navegación y el diseño sean intuitivos y que la disposición de los elementos funcione en pantallas de distintos tamaños.
Pruebas de estrés: Probar la aplicación con una gran cantidad de citas y clientes para garantizar que la aplicación funcione de manera fluida bajo carga.
Pruebas de errores: Validar que los errores de entrada del usuario (como fechas inválidas o campos vacíos) se manejen correctamente y no provoquen fallos en la aplicación.

Aspectos por optimizar o corregir antes de la finalización

Mejora en la validación de campos: Implementar validaciones más estrictas en los campos de entrada, como la validación del formato de la fecha y la hora.
Optimización de la interfaz de usuario: Mejorar la disposición y el diseño visual para que la aplicación sea más atractiva y fácil de usar, siguiendo las pautas de Material Design y siguiendo la estructura de colores de la aplicación.
Manejo avanzado de reportes: Incorporar la posibilidad de generar reportes no solo en PDF, sino también en otros formatos como Excel, para mayor versatilidad.

Planificación de las últimas etapas del desarrollo

Semana 1: Finalizar módulos restantes, realizando pruebas unitarias, funcionales y de UI, ajustando cualquier error encontrado durante el proceso.
Semana 2: Implementar mejoras en la validación de campos.
Semana 3: Realizar pruebas finales con datos reales, optimizar la generación de reportes y preparar la versión final para ser entregada.

Conclusión y cierre

Reflexión sobre el proceso de desarrollo y los aprendizajes obtenidos

El desarrollo de la aplicación BarberReserver fue una experiencia que está permitiendo aplicar y expandir conocimientos en desarrollo de aplicaciones móviles con Android. Durante el proceso, se ha aprendido a gestionar de manera eficiente bases de datos locales mediante SQLite y a implementar interfaces de usuario intuitivas. Además, la implementación de nuevas funcionalidades, como la validación de fecha y hora, así como la futura incorporación de generar reportes en PDF y rutas a imágenes, mejorará la experiencia de usuario.

Reconocimientos y agradecimientos a personas o entidades que apoyaron el proyecto

Agradezco a mis profesores y compañeros por su apoyo y orientación durante el desarrollo de este proyecto. Su retroalimentación fue fundamental para mejorar la calidad de la aplicación y asegurar que cumpliera con todos los requisitos. También quiero agradecer a los usuarios de prueba (barberos con los que me he atendido) que colaboraron proporcionando sugerencias y observaciones clave para optimizar la aplicación. Finalmente, agradezco a las comunidades de desarrolladores, herramientas de inteligencia artificial generativa y recursos en línea, que proporcionaron documentación y ejemplos durante el proceso de desarrollo.