Gestor de Tareas con Flutter y Floor

Descripción

Esta aplicación es un gestor de tareas desarrollado en Flutter, utilizando Floor como base de datos local. Permite a los usuarios añadir, visualizar, marcar como completadas y eliminar tareas, asegurando la persistencia de datos mediante SQLite.

La aplicación es la misma que el proyecto anterior pero le hemos añadido unos campos de formulario en una nueva pantalla para el login, además de los mensajes emergentes y navegación entre pantallas que te lleva del login, en caso de ser correcto, a la pantalla principal de la aplicación anterior, la que hemos mejorado visualmente. Para así cumplir con todas las especificaciones del enunciado.

Objetivos

- Implementar un sistema de autenticación básico con usuario y contraseña predefinidos.
- Usar Floor para gestionar la persistencia de datos.
- Permitir la creación, visualización, actualización y eliminación de tareas.
- Implementar al proyecto anterior navegación entre pantallas
- Implementar mensajes emergentes, en este caso mediante snackBars

Componentes Principales

database.dart – Configuración de la Base de Datos

Define la base de datos utilizando Floor, especificando que manejará la entidad `Tarea` y su acceso a través de `TareaDao`.

- `@Database(version: 1, entities: [Tarea])`: Especifica la versión de la base de datos y las entidades que maneja.
- `AppDatabase extends FloorDatabase`: Clase abstracta que actúa como contenedor de la base de datos.
- `TareaDao get tareaDao`: Proporciona acceso al DAO para gestionar tareas.

tarea.dart – Modelo de la Tarea

Define la estructura de la entidad `Tarea`, que se almacena en la base de datos.

- `@PrimaryKey(autoGenerate: true)`: Declara el ID como clave primaria y lo genera automáticamente.
- `final String title`: Almacena el título de la tarea.
- `final bool isCompleted`: Indica si la tarea está completada.
- `Tarea.copyWith(...)`: Permite crear una nueva tarea basada en otra existente, modificando solo algunos atributos.

tareaDao.dart – Acceso a los Datos

Define las operaciones para la entidad `Tarea`.

- `@Query('SELECT * FROM Tarea ORDER BY id DESC')`: Obtiene todas las tareas ordenadas por ID en orden descendente.
- `Future<List<Tarea>> getAllTasks()`: Devuelve la lista de tareas almacenadas.
- `@insert Future<void> insertTask(Tarea task)`: Inserta una nueva tarea en la base de datos.
- `@update Future<void> updateTask(Tarea task)`: Actualiza una tarea existente.
- `@delete Future<void> deleteTask(Tarea task)`: Elimina una tarea de la base de datos.

loginScreen.dart – Pantalla de Inicio de Sesión

Gestiona el acceso a la aplicación mediante credenciales predefinidas.

- `final String _usuarioPredeterminado = En este caso hemos usado 'UsuarioPrueba'`
- `final String _contrasenaPredeterminada = En este caso es '12345'`

Todo lo que no sea esto nos dará un login incorrecto.

Método `_validarLogin()`

- Verifica si el usuario y la contraseña ingresados coinciden con los valores predefinidos.
- Si las credenciales son correctas, muestra un mensaje y redirige a `MainScreen`.
- Si las credenciales son incorrectas, muestra un mensaje de error.

mainScreen.dart – Gestión de Tareas

Pantalla principal donde los usuarios pueden añadir, visualizar, completar y eliminar tareas.

Método `_initializeDatabase()`

- Inicializa la base de datos y obtiene la instancia de `TareaDao`.

Método `_loadTasks()`

- Carga todas las tareas desde la base de datos y actualiza la lista mostrada en pantalla.

Método `_addTask(String title)`

- Crea una nueva tarea con el título ingresado.
- Inserta la tarea en la base de datos y actualiza la lista.

Método `_toggleCompletion(Tarea task)`

- Cambia el estado de completado de una tarea.
- Llama a `updateTask()` para actualizar la tarea en la base de datos.

Método `_deleteTasks()`

- Filtra y elimina todas las tareas que están marcadas como completadas.

Método `_deleteTask(Tarea task)`

- Elimina una tarea individualmente.

Método `_showTaskDialog(BuildContext context)`

- Muestra un cuadro de diálogo donde el usuario puede ingresar el título de una nueva tarea.

main.dart – Punto de Entrada de la Aplicación

Define las rutas de navegación y establece la pantalla de inicio (`LoginScreen`).

- `initialRoute: '/login'`: Define la pantalla de inicio de sesión como la primera vista al abrir la aplicación.
- `onGenerateRoute`: Gestiona la navegación entre `LoginScreen` y `MainScreen`.