Revisión y Entrega de la Actividad

Jankarlos Crespo Hernández R00637294

Finaliza tu aplicación ejecutando y probando cada funcionalidad:

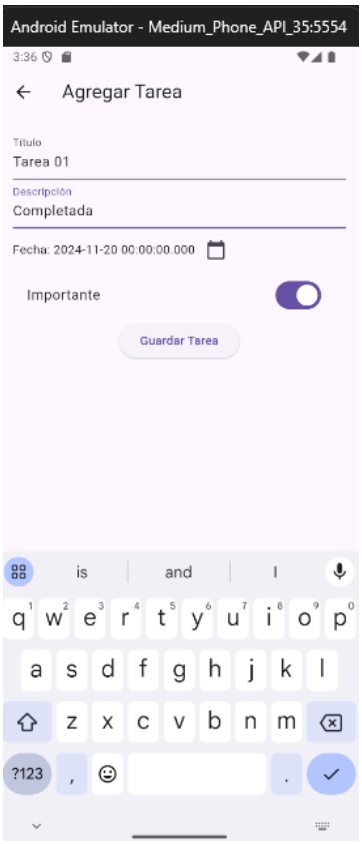
1. Agrega, edita y elimina tareas, y verifica que los datos se guarden correctamente.
2. Usa el Switch para marcar una tarea importante.
3. Comprueba el funcionamiento de los diálogos y los Snackbar.
4. Presenta la aplicación al profesor para evaluar las funcionalidades implementadas.

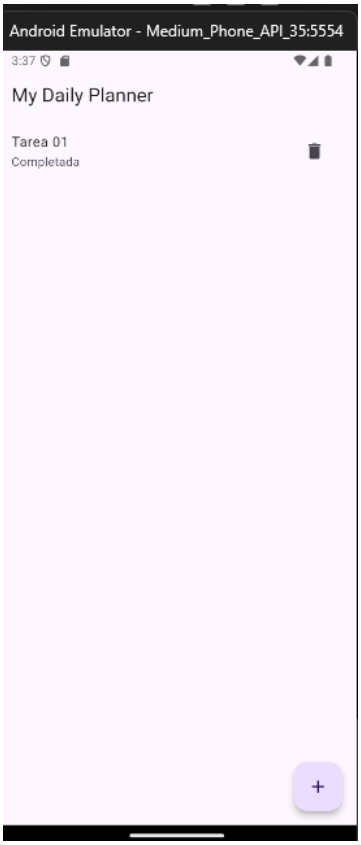
5. Entrega:

a. Desarrolla un documento en PDF que demuestre la funcionalidad de la aplicación y cada pantalla.

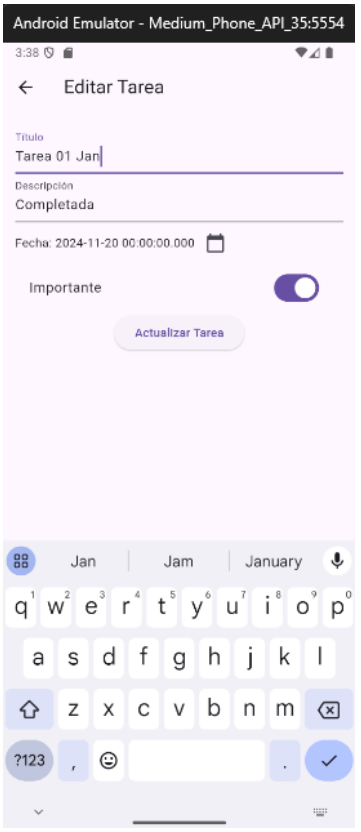
b. Añade todo el código generado por ti al final del documento.

c. Entrega el documento en el enlace provisto para esta actividad.





(Guardando Tarea)



(Editando Tarea)

Código (Main):

```dart
import 'package:flutter/material.dart';
import 'screens/task_list_screen.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'My Daily Planner',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: TaskListScreen(),
    );
  }
}
```

task_list_screen:

```dart
import 'package:flutter/material.dart';
import '../models/task.dart';
import '../screens/task_form_screen.dart';

class TaskListScreen extends StatefulWidget {
  @override
  _TaskListScreenState createState() => _TaskListScreenState();
}

class _TaskListScreenState extends State<TaskListScreen> {
  List<Task> tasks = [];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('My Daily Planner')),
      body: ListView.builder(
        itemCount: tasks.length,
        itemBuilder: (context, index) {
```

```
          final task = tasks[index];
          return ListTile(
            title: Text(task.title),
            subtitle: Text(task.description),
            trailing: IconButton(
              icon: Icon(Icons.delete),
              onPressed: () {
                setState(() {
                  showDialog(
                    context: context,
                    builder: (BuildContext context) {
                      return AlertDialog(
                        title: Text('Eliminar Tarea'),
                        content: Text('¿Está seguro de eliminar esta tarea?'),
                        actions: [
                          TextButton(
                            child: Text('Cancelar'),
                            onPressed: () => Navigator.of(context).pop(),
                          ),
                          TextButton(
                            child: Text('Eliminar'),
                            onPressed: () {
                              setState(() {
                                tasks.removeAt(index);
                              });
                              Navigator.of(context).pop();
                              ScaffoldMessenger.of(context).showSnackBar(
                                  SnackBar(content: Text('Tarea eliminada')));
                            },
                          ),
                        ],
                      );
                    },
                  );
                });
              },
            ),
            onTap: () async {
              // Transición para editar la tarea
              final updatedTask = await Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => TaskFormScreen(task: task),
                ),
              );
```

```
              if (updatedTask != null) {
                setState(() {
                  tasks[index] = updatedTask; // Actualiza la tarea en la lista
                });
              }
            },
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.add),
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => TaskFormScreen()),
          ).then((newTask) {
            if (newTask != null) {
              setState(() {
                tasks.add(newTask);
              });
            }
          });
        },
      ),
    );
  }
}
```

task_form_screens:

```
import 'package:flutter/material.dart';
import '../models/task.dart';

class TaskFormScreen extends StatefulWidget {
  final Task? task; // Tarea opcional para editar

  TaskFormScreen({this.task});

  @override
  _TaskFormScreenState createState() => _TaskFormScreenState();
}

class _TaskFormScreenState extends State<TaskFormScreen> {
```

```dart
late TextEditingController _titleController;
late TextEditingController _descriptionController;
DateTime? _selectedDate;
bool _isImportant = false;

@override
void initState() {
  super.initState();
  _titleController = TextEditingController(text: widget.task?.title ?? '');
  _descriptionController =
      TextEditingController(text: widget.task?.description ?? '');
  _selectedDate = widget.task?.dueDate;
  _isImportant = widget.task?.isImportant ?? false;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.task == null ? 'Agregar Tarea' : 'Editar Tarea'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          TextField(
            controller: _titleController,
            decoration: InputDecoration(labelText: 'Título'),
          ),
          TextField(
            controller: _descriptionController,
            decoration: InputDecoration(labelText: 'Descripción'),
          ),
          Row(
            children: [
              Text(_selectedDate != null
                  ? 'Fecha: ${_selectedDate!.toLocal()}'
                  : 'Seleccione una fecha'),
              IconButton(
                icon: Icon(Icons.calendar_today),
                onPressed: () async {
                  DateTime? pickedDate = await showDatePicker(
                    context: context,
                    initialDate: DateTime.now(),
                    firstDate: DateTime(2000),
```

```
                    lastDate: DateTime(2101),
                  );
                  if (pickedDate != null) {
                    setState(() {
                      _selectedDate = pickedDate;
                    });
                  }
                },
              ),
            ],
          ),
          SwitchListTile(
            title: Text('Importante'),
            value: _isImportant,
            onChanged: (bool value) {
              setState(() {
                _isImportant = value;
              });
            },
          ),
          ElevatedButton(
            child: Text(
                widget.task == null ? 'Guardar Tarea' : 'Actualizar Tarea'),
            onPressed: () {
              final updatedTask = Task(
                title: _titleController.text,
                description: _descriptionController.text,
                dueDate: _selectedDate ?? DateTime.now(),
                isImportant: _isImportant,
              );
              Navigator.pop(context, updatedTask);
            },
          ),
        ],
      ),
    ),
  );
 }
}
```