Main.dart:

```
import 'package:flutter/material.dart';
import 'tela.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Agenda de Contatos',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomeScreen(),
    );
  }
}
```

Tela.dart:

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

```
import 'db.dart';
import 'add_contato_tela.dart';
import 'edit_contato_tela.dart';

//Versão 3.0 Aplicativo Projeto Integrador

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final DB db = DB();
  List<Contato> contatos = [];

  @override
  void initState() {
    super.initState();
    _loadContatos();
  }

  // Carrega os contatos da base de dados
  _loadContatos() async {
    contatos = await db.getContatos();
    setState(() {});
  }
```

```
// Exibe a confirmação de exclusão
void _showDeleteConfirmation(int id) {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text("Excluir Contato"),
      content: Text("Tem certeza que deseja excluir este contato?"),
      actions: [
        TextButton(
          onPressed: () {
            _deleteContato(id); // Se o usuário confirmar, deleta o contato
            Navigator.pop(context); // Fecha o diálogo
          },
          child: Text("Sim"),
        ),
        TextButton(
          onPressed: () => Navigator.pop(context), // Fecha o diálogo sem excluir
          child: Text("Não"),
        ),
      ],
    ),
  );
}

// Função que deleta o contato
_deleteContato(int id) async {
  await db.deleteContato(id); // Chama o método da base de dados para excluir o contato
```

```
  _loadContatos(); // Atualiza a lista de contatos
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("Agenda de Contatos"),
      titleTextStyle: TextStyle(color: Colors.white, fontSize: 32, fontWeight: FontWeight.bold),
      backgroundColor: Colors.blueGrey[900],
      elevation: 4.0,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          ElevatedButton(
            onPressed: () async {
              // Navega para a tela de adicionar contato
              await Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => AddContactScreen()),
              );
              _loadContatos(); // Atualiza a lista após adicionar
            },
            child: Text("Adicionar Contato"),
```

```
          style: ElevatedButton.styleFrom(padding: EdgeInsets.symmetric(vertical: 16)),
        ),
        SizedBox(height: 16),
        Expanded(
          child: ListView.builder(
            itemCount: contatos.length,
            itemBuilder: (context, index) {
              final contato = contatos[index];
              return Card(
                margin: EdgeInsets.symmetric(vertical: 8),
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
                elevation: 4,
                child: ListTile(
                  contentPadding: EdgeInsets.symmetric(vertical: 12, horizontal: 16),
                  title: Text(contato.nome, style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
                  subtitle: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(contato.telefone, style: TextStyle(fontSize: 18, color: Colors.grey[600])),
                      if (contato.email != null) Text(contato.email!, style: TextStyle(fontSize: 16, color: Colors.grey[600])),
                    ],
                  ),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
```

```
                    icon: Icon(Icons.edit, color: Colors.blueGrey),
                    onPressed: () async {
                      await Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => EditContactScreen(contato: contato)),
                      );
                      _loadContatos(); // Atualiza a lista após editar
                    },
                  ),
                  IconButton(
                    icon: Icon(Icons.delete, color: Colors.red),
                    onPressed: () => _showDeleteConfirmation(contato.id!),
                  ),
                ],
              ),
            ),
          );
        },
      ),
    ),
  ],
),
),
);
}
}
```

add_contato_tela.dart:

```
import 'package:flutter/material.dart';
import 'db.dart';

class AddContactScreen extends StatefulWidget {
  @override
  _AddContactScreenState createState() => _AddContactScreenState();
}

class _AddContactScreenState extends State<AddContactScreen> {
  final _nomeController = TextEditingController();
  final _telefoneController = TextEditingController();
  final _emailController = TextEditingController(); // Novo controller para e-mail

  final DB db = DB();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Adicionar Contato"),
        backgroundColor: Colors.blueGrey[900],
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
```

```
children: [
  TextField(
    controller: _nomeController,
    decoration: InputDecoration(labelText: 'Nome'),
  ),
  TextField(
    controller: _telefoneController,
    decoration: InputDecoration(labelText: 'Telefone'),
  ),
  TextField(
    controller: _emailController, // Campo de e-mail
    decoration: InputDecoration(labelText: 'E-mail'),
  ),
  SizedBox(height: 16),
  ElevatedButton(
    onPressed: () async {
      final nome = _nomeController.text;
      final telefone = _telefoneController.text;
      final email = _emailController.text; // Pega o e-mail

      if (nome.isEmpty || telefone.isEmpty) {
        // Validar se os campos obrigatórios estão preenchidos
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text('Nome e Telefone são obrigatórios!'),
        ));
        return;
      }
```

```
            final contato = Contato(
              nome: nome,
              telefone: telefone,
              email: email.isNotEmpty ? email : null, // Se o e-mail não for vazio, inclui
            );

            // Tente adicionar o contato
            try {
              await db.insertContato(contato); // Método para inserir no banco
               Navigator.pop(context); // Volta para a tela anterior
             } catch (e) {
               ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                 content: Text('Erro ao adicionar contato: $e'),
               ));
             }
            },
            child: Text("Salvar Contato"),
          ),
        ],
      ),
    ),
  );
 }
}
```

edit_contato_tela.dart

```
import 'package:flutter/material.dart';
import 'db.dart';

class EditContactScreen extends StatefulWidget {
  final Contato contato;

  EditContactScreen({required this.contato});

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

class _EditContactScreenState extends State<EditContactScreen> {
  late TextEditingController _nomeController;
  late TextEditingController _telefoneController;
  late TextEditingController _emailController; // Novo controller para e-mail

  final DB db = DB();

  @override
  void initState() {
    super.initState();
```

```
  _nomeController = TextEditingController(text: widget.contato.nome);
  _telefoneController = TextEditingController(text: widget.contato.telefone);
  _emailController = TextEditingController(text: widget.contato.email ?? ''); // Preenche o campo de e-mail
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text("Editar Contato"),
    backgroundColor: Colors.blueGrey[900],
   ),
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: Column(
     children: [
      TextField(
       controller: _nomeController,
       decoration: InputDecoration(labelText: 'Nome'),
      ),
      TextField(
       controller: _telefoneController,
       decoration: InputDecoration(labelText: 'Telefone'),
      ),
      TextField(
```

```
        controller: _emailController, // Campo de e-mail
        decoration: InputDecoration(labelText: 'E-mail'),
      ),
      SizedBox(height: 16),
      ElevatedButton(
        onPressed: () async {
          final nome = _nomeController.text;
          final telefone = _telefoneController.text;
          final email = _emailController.text; // Pega o e-mail

          final contato = Contato(
            id: widget.contato.id,
            nome: nome,
            telefone: telefone,
            email: email,
          );

          await db.updateContato(contato); // Método para atualizar no banco
          Navigator.pop(context); // Volta para a tela anterior
        },
        child: Text("Salvar Alterações"),
      ),
    ],
  ),
),
```

```
    );
  }
}
```

db.dart:

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

class Contato {
  final int? id;
  final String nome;
  final String telefone;
  final String? email;

  Contato({this.id, required this.nome, required this.telefone, this.email});

  // Converte um Contato para um Map (para salvar no banco de dados)
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'nome': nome,
       'telefone': telefone,
      'email': email,
    };
  }
```

```
  // Converte um Map para um Contato (quando carregado do banco)
  factory Contato.fromMap(Map<String, dynamic> map) {
    return Contato(
      id: map['id'],
      nome: map['nome'],
      telefone: map['telefone'],
      email: map['email'],
    );
  }
}

class DB {
  static Database? _database;

  Future<Database> get database async {
    if (_database != null) return _database!;
    _database = await _initDB();
    return _database!;
  }

  _initDB() async {
    final dbPath = await getDatabasesPath();
    final path = join(dbPath, 'agenda_contatos.db');
    return await openDatabase(path, version: 2, onCreate: (db, version) async {
      await db.execute('''
      CREATE TABLE contatos(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
      nome TEXT,
      telefone TEXT,
      email TEXT  -- Adicionando o campo email
      )
  ''');
  }, onUpgrade: (db, oldVersion, newVersion) async {
    if (oldVersion < 2) {
      // Caso a versão anterior seja menor que 2, alteramos a tabela
      await db.execute('''
      ALTER TABLE contatos ADD COLUMN email TEXT;
  '');
    }
  });
}



Future<int> insertContato(Contato contato) async {
  final db = await database;
  return await db.insert(
    'contatos',
    contato.toMap(),
    conflictAlgorithm: ConflictAlgorithm.replace,
  );
}



 Future<List<Contato>> getContatos() async {
```

```
    final db = await database;
    final List<Map<String, dynamic>> maps = await db.query('contatos');
    return List.generate(maps.length, (i) {
      return Contato.fromMap(maps[i]);
    });
  }

  Future<int> updateContato(Contato contato) async {
    final db = await database;
    return await db.update(
      'contatos',
      contato.toMap(),
      where: 'id = ?',
      whereArgs: [contato.id],
    );
  }

  Future<int> deleteContato(int id) async {
    final db = await database;
    return await db.delete(
      'contatos',
      where: 'id = ?',
      whereArgs: [id],
    );
  }
}
```

Mudança no Pubspec.yaml:

```
dependencies:
  flutter:
    sdk: flutter
  sqflite: ^2.0.0+4
  path_provider: ^2.0.10
```

Documentação do Aplicativo Agenda de Contatos

Este é um aplicativo de gerenciamento de contatos simples, utilizando o Flutter como framework de desenvolvimento e o SQLite como banco de dados local. O aplicativo permite ao usuário adicionar, editar e excluir contatos. Cada contato possui um nome, telefone e, opcionalmente, um e-mail.

Estrutura do Projeto

1. main.dart

Responsável por inicializar o aplicativo.

- **Função main:** Inicializa o aplicativo chamando o MyApp.
- **Classe MyApp:**
  - Configura o MaterialApp.
  - Define o tema principal e desativa a exibição do banner de debug.
  - Define a tela inicial como HomeScreen.

2. tela.dart

Contém a lógica e a interface da tela principal do aplicativo.

- **Classe HomeScreen:**
  - Exibe a lista de contatos.
  - Permite adicionar, editar e excluir contatos.
- **Funções principais:**
  - **_loadContatos:** Carrega os contatos do banco de dados.
  - **_showDeleteConfirmation:** Exibe um diálogo de confirmação para exclusão.
  - **_deleteContato:** Remove o contato do banco de dados e atualiza a lista.
- **Componentes:**
  - Botão para adicionar contatos.
  - ListView para exibir os contatos em cartões estilizados.

3. add_contato_tela.dart

Gerencia a tela de adição de novos contatos.

- **Classe AddContactScreen:**
  - Contém campos de entrada para nome, telefone e email.
  - Salva os dados no banco de dados ao clicar em "Salvar Contato".
  - Valida se os campos obrigatórios foram preenchidos.

4. edit_contato_tela.dart

Gerencia a tela de edição de contatos existentes.

- **Classe EditContactScreen:**
  - Permite editar os campos nome, telefone e email.
  - Atualiza os dados no banco de dados ao clicar em "Salvar Alterações".

5. db.dart

Responsável pela interação com o banco de dados SQLite.

- **Classe DB:**
  - Configura e inicializa o banco de dados.
  - Define métodos para CRUD:
    - **insertContato:** Insere um novo contato.
    - **getContatos:** Recupera todos os contatos.
    - **updateContato:** Atualiza um contato existente.
    - **deleteContato:** Exclui um contato.
- **Classe Contato:**
  - Modelo para representar um contato.
  - Converte os dados entre Map e objetos para facilitar a interação com o banco de dados.

Dependências Utilizadas

As seguintes dependências foram adicionadas no arquivo pubspec.yaml:

- **sqflite:** Para gerenciar o banco de dados SQLite.
- **path_provider:** Para obter o caminho do diretório do sistema para armazenar o banco de dados.

Fluxo do Aplicativo

- Exibe a lista de contatos recuperados do banco de dados.
- Oferece opções para adicionar, editar ou excluir contatos.

1. **Adição de Contato (AddContactScreen)**:
   - Permite inserir o nome, telefone e email de um novo contato.
   - Salva o contato no banco de dados local.
2. **Edição de Contato (EditContactScreen)**:
   - Permite alterar os dados de um contato existente.
   - Atualiza o banco de dados com as alterações feitas.
3. **Banco de Dados (db.dart)**:
   - Armazena e gerencia os dados localmente utilizando SQLite.
   -

Funcionalidades

- **Adicionar Contato:** Permite inserir um novo contato com nome, telefone e e-mail.
- **Editar Contato:** Permite modificar os dados de um contato existente.
- **Excluir Contato:** Exclui um contato após confirmação do usuário.
- **Armazenamento Local:** Utiliza SQLite para persistir os dados de contatos.