PARTE 3: QUALIDADE DE SOFTWARE E TESTES

Disciplina: Computação Móvel			
Período: 8º	Turma: Noturna	Valor da Atividade: 1.0	Nota Obtida:
Professor: Edgard da Cunha Pontes		Avaliação Processual - 2° Bimestre	

ATIVIDADE EM GRUPO

Máximo de Integrantes: 3 pessoas (se possível, manter o mesmo grupo da Avaliação Processual I)

Data de entrega: até o dia 11/11/2025 (terça-feira)

FORMATO DE ENTREGA

- **Por e-mail:** edgardpontes@professor.multivix.edu.br
- **Atualização Final do repositório no GitHub** com o código-fonte completo do projeto.
- **Relatório da Parte 3**, conforme as especificações detalhadas abaixo, no formato de um arquivo `README.md` atualizado, incluindo os resultados da execução dos testes.

OBJETIVO

O foco desta terceira e última parte é garantir a qualidade e a robustez do aplicativo, utilizando a ferramenta de testes nativa do Flutter (`flutter_test`). Os alunos devem demonstrar a capacidade de validar a lógica de negócios (Repositório) e a interatividade da interface (*Widgets*).

- **Testes de Unidade:** Escrever testes para validar as operações CRUD nos Repositórios (ex: `DisciplinaRepository` e `AlunoRepository`).
- **Testes de *Widget*:** Escrever testes que validem a renderização e o comportamento da interface de usuário em resposta a eventos.
- **Boas Práticas:** Refatorar o código-fonte para seguir os princípios de "Clean Code" e convenções de nomenclatura da linguagem Dart.

DESENVOLVIMENTO

Contexto Geral: O aplicativo está funcional e persistente. Agora, é necessário provar, através de testes automatizados, que as funcionalidades críticas (salvar, listar, atualizar e deletar) operam conforme o esperado, protegendo o código contra futuros erros (regressões).

Enunciado da Atividade: Implementação de Testes Automatizados

1) Preparação do Ambiente de Testes:

- Garanta que o projeto possui a estrutura correta para testes (`test/` na raiz do projeto).
- Configure quaisquer *mocks* ou dependências necessárias para isolar a camada de repositório (ex: mockar a conexão do banco de dados, se aplicável, ou utilizar um banco de dados temporário em memória para testes).

2) Desenvolvimento de Testes de Unidade ("*Unit Tests*"):

- Escreva no **mínimo 4 Testes de Unidade** nos arquivos de teste relacionados aos Repositórios (ex: `DisciplinaRepository` e `AlunoRepository`).
- **Obrigatório testar as operações CRUD principais:**
  - **Teste 1:** Garantir que o método de **Criação (Salvar)** insere um objeto no banco e o método de **Leitura (Listar)** o retorna corretamente.
  - **Teste 2:** Verificar se o método de **Exclusão (Delete)** remove o objeto e a lista de leitura subsequente o confirma.
  - **Teste 3:** Validar se o método de **Atualização (Update)** modifica corretamente os dados de um objeto.

3) Desenvolvimento de Testes de Widget ("*Widget Tests*"):

- Escreva **no mínimo 2 Testes de Widget** focados nas telas principais ou componentes reutilizáveis.
- **Obrigatório testar a interface e a interatividade:**
  - **Teste 4:** Testar se a **Tela de Listagem** carrega o *widget* correto quando a lista de Disciplinas/Alunos está **vazia** (ex: exibe uma mensagem "Nenhum

cadastro encontrado").

- **Teste 5:** Testar a **interação com o formulário**: Simular o preenchimento de um campo e verificar se o valor do *widget* foi alterado (sem a necessidade de salvar no banco).

4) Refatoração e Boas Práticas ("*Clean Code*"):

- Revisar todo o código do projeto, garantindo:
  - Nomes de variáveis e métodos claros e em inglês ou português consistentemente.
  - Remoção de código comentado ou inutilizado.
  - Uso do modifier `const` em *widgets* que não mudam (para otimização).
  - Aplicação consistente de formatação (`flutter format .`).

REQUISITOS MÍNIMOS

- **Relatório da Parte 2 (`README.md`):**
  - **Resultados dos Testes:** Inclua uma captura de tela do terminal mostrando o resultado da execução dos testes (`flutter test`) com **todos os testes passando** (em verde).
  - **Descrição dos Testes:** Escolha o teste de Unidade mais complexo e o teste de *Widget* mais relevante e explique **o que** ele testa e **por que** ele é importante para a qualidade do software.
  - **Refatoração:** Liste as principais melhorias de "*Clean Code*" aplicadas ao projeto nesta fase final.

CRITÉRIOS DE AVALIAÇÃO (Parte 1)

- **Testes de Unidade - Cobertura e Correção(0,40 ponto):**
  - Implementação de no mínimo 3 "*Unit Tests*" funcionais e eficazes para validar as operações CRUD nos Repositórios. Uso adequado das bibliotecas de teste e mocks.
- **Testes de Widget - Funcionalidade da UI (0,30 ponto):**
  - Implementação de no mínimo 2 "*Widget Testes*" para validar a

renderização e uma interação da interface. Uso correto dos *Finders* e `tester.pumpWidget()`.

- **Boas Práticas de Código "*Clean Code*" (0,20 ponto):**
  - Código limpo, bem organizado, com convenções de nomenclatura consistentes, uso de `const` e separação de responsabilidades mantidas (`Model`/`Repository`/`Provider`/`UI`).
- **Relatório e Evidências (0,10 ponto):**
  - O relatório (`README.md`) deve documentar os testes, a importância da refatoração e incluir a prova de execução (`flutter test`) com sucesso.

AVALIAÇÃO

A Avaliação Processual II (Partes 1, 2 e 3) representa um projeto completo, onde o aluno demonstra o domínio de toda a cadeia de valor do desenvolvimento móvel em Flutter: da lógica em Dart (Parte 1), passando pela arquitetura reativa e interface (Parte 2), até a qualidade de software e testes (Parte 3). O sucesso neste trabalho garante a excelência na formação do futuro profissional.

Observações:

- Este é um trabalho em grupo, e espera-se que todos os membros contribuam de forma ativa e significativa para o desenvolvimento do projeto, dividindo as tarefas e colaborando nas decisões.