# **Documento de Estratégia de Testes**

## **Projeto: Calculadora Flexível para Redes Ópticas PON**

### **1. Introdução**

Este documento descreve a abordagem de testes para o sistema "Calculadora PON Web". O objetivo é garantir a fiabilidade, precisão e robustez da aplicação através da implementação de testes automatizados. Os testes são divididos em duas categorias principais: Testes de Unidade e Testes de Integração.

### **2. Ferramentas Utilizadas**

* **Pytest:** É o framework de testes padrão para Python, escolhido pela sua simplicidade, poder e ecossistema rico em plugins.
* **Cliente de Testes do Flask:** O próprio Flask fornece um cliente de testes que nos permite simular requisições web (GET, POST, etc.) sem a necessidade de um servidor real, tornando os testes de integração rápidos e eficientes.

### **3. Estrutura dos Testes**

Para manter a organização, todos os ficheiros de teste serão colocados numa pasta tests/ na raiz do projeto.

/calculadora-pon-web/  
├── app.py  
├── calculadora\_pon.py  
├── templates/  
├── static/  
└── tests/ <-- Nova pasta  
 ├── \_\_init\_\_.py (ficheiro vazio)  
 └── test\_calculadora.py (Testes de Unidade para o Model)  
 └── test\_app.py (Testes de Integração para o Controller)

### **4. Tipos de Testes**

#### **4.1. Testes de Unidade (Model)**

* **Foco:** calculadora\_pon.py
* **Objetivo:** Validar a lógica de negócio de forma isolada. Queremos garantir que os cálculos matemáticos estão corretos para todos os cenários.
* **Abordagem:**
  1. Instanciar a classe ParametrosPON com dados de teste.
  2. Passar esses parâmetros para a CalculadoraPON.
  3. Chamar o método calcular().
  4. Verificar (com assert) se o resultado retornado é exatamente o esperado.
  5. Testar cenários de sucesso (ex: cálculo de margem viável), cenários de borda (ex: margem negativa) e cenários de erro (ex: tentar calcular com múltiplos campos vazios, o que deve levantar uma exceção ValueError).

#### **4.2. Testes de Integração (Controller e View)**

* **Foco:** app.py
* **Objetivo:** Validar o fluxo completo de uma requisição HTTP, desde a rota até à renderização do template.
* **Abordagem:**
  1. Usar o cliente de testes do Flask para simular requisições.
  2. **Teste da Página Inicial:** Fazer um GET para a rota / e verificar se a resposta tem status\_code 200 (OK) e se o HTML contém o título da página.
  3. **Teste de Cálculo com Sucesso:** Fazer um POST para a rota /calcular com dados de formulário válidos. Verificar se a resposta é 200 (OK) e se o HTML da página de resultado contém o valor calculado.
  4. **Teste de Cálculo com Erro:** Fazer um POST para /calcular com dados inválidos (ex: dois campos vazios). Verificar se a página de resultado exibe a mensagem de erro esperada.

### **5. Como Executar os Testes**

1. **Instalar o Pytest:**  
   pip install pytest
2. Executar os testes:  
   A partir da pasta raiz do projeto (calculadora-pon-web/), basta executar o comando:  
   pytest  
     
   O Pytest irá descobrir e executar automaticamente todos os ficheiros de teste dentro da pasta tests/.