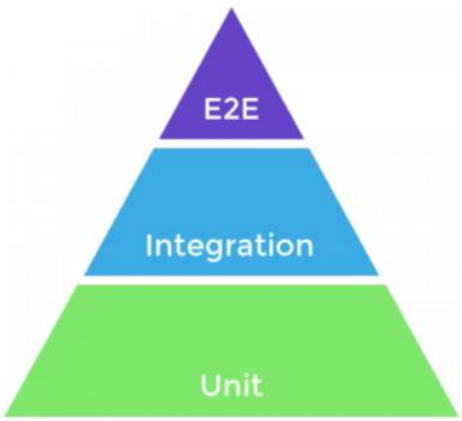
1 Plano de Testes (*draft*)

O presente documento tem como objetivo definir a estratégia global de testes a aplicar no projeto SeePaw.

Sendo um documento em versão *draft*, o plano será progressivamente atualizado ao longo do ciclo de desenvolvimento, acompanhando a evolução da aplicação e o aumento da complexidade dos testes nas próximas milestones.

1.1 Abordagem de Testes

A abordagem de testes adotada no projeto SeePaw baseia-se na pirâmide de testes, ilustrada na figura seguinte, que representa a distribuição equilibrada entre os diferentes tipos de teste ao longo do processo de desenvolvimento.



A pirâmide reflete o princípio de que quanto mais baixo o nível de teste, maior a quantidade e a rapidez de execução, enquanto os níveis superiores são menos numerosos, mas mais abrangentes e próximos da experiência real do utilizador.

A pirâmide de testes é composta por três camadas principais:

- **Testes Unitários:** formam a base da pirâmide e são executados em maior número. Verificam o correto funcionamento de métodos e componentes isolados.
- **Testes de Integração:** situam-se na camada intermédia e garantem que os diferentes módulos do sistema comunicam corretamente entre si (por exemplo, entre o controlador, os serviços e a base de dados).
- **Testes End-to-End (E2E) / Testes de Sistema:** representam o topo da pirâmide e validam o comportamento global da aplicação do ponto de vista do utilizador, simulando interações reais com a interface e verificando o cumprimento dos requisitos funcionais.

1.2 Ferramentas e Tecnologias de Teste

A tabela seguinte apresenta um resumo das tecnologias escolhidas e o respetivo propósito dentro da estratégia de testes.

Tipo de Teste	Objetivo Principal	Ferramentas/ Tecnologias	Observações
Unitário /Backend	Validar métodos e componentes isolados	xUnit (.NET Core)	Utilizado no backend; suporta *mocking*.
Integração	Testar o fluxo entre componentes (controller → serviço → BD)	Postman	Postman para requests
End-to-End (E2E)	Simular o comportamento real do utilizador	Playwright – WebApp Espresso - Mobile	Automatiza cenários críticos no *frontend* web e na aplicação móvel.
Aceitação / Cliente	Validar se o sistema cumpre os requisitos e expectativas do cliente	—	Executados manualmente pelos utilizadores ou docentes durante as fases finais de entrega e demonstração.

1.3 Critérios de Cobertura e Prioridades de Teste

A estratégia de testes da aplicação SeePaw estabelece um objetivo global de cobertura mínima de 80% do código, assegurando que a maioria das funcionalidades e fluxos principais são exercitados por testes automatizados.

Embora algumas áreas possam exigir uma cobertura superior (como os processos críticos de *fostering* e *ownership*), o foco não está apenas na percentagem atingida, mas na qualidade dos testes.

1.4 Integração com o Fluxo de Desenvolvimento e CI/CD

A estratégia de testes da aplicação SeePaw está diretamente integrada no fluxo de desenvolvimento e nas pipelines de CI/CD.

A estratégia de versionamento do projeto SeePaw baseia-se em duas branches principais; develop e main, complementadas por branches temporárias de feature. A tabela seguinte apresenta a função de cada uma e os testes que nelas são executados.

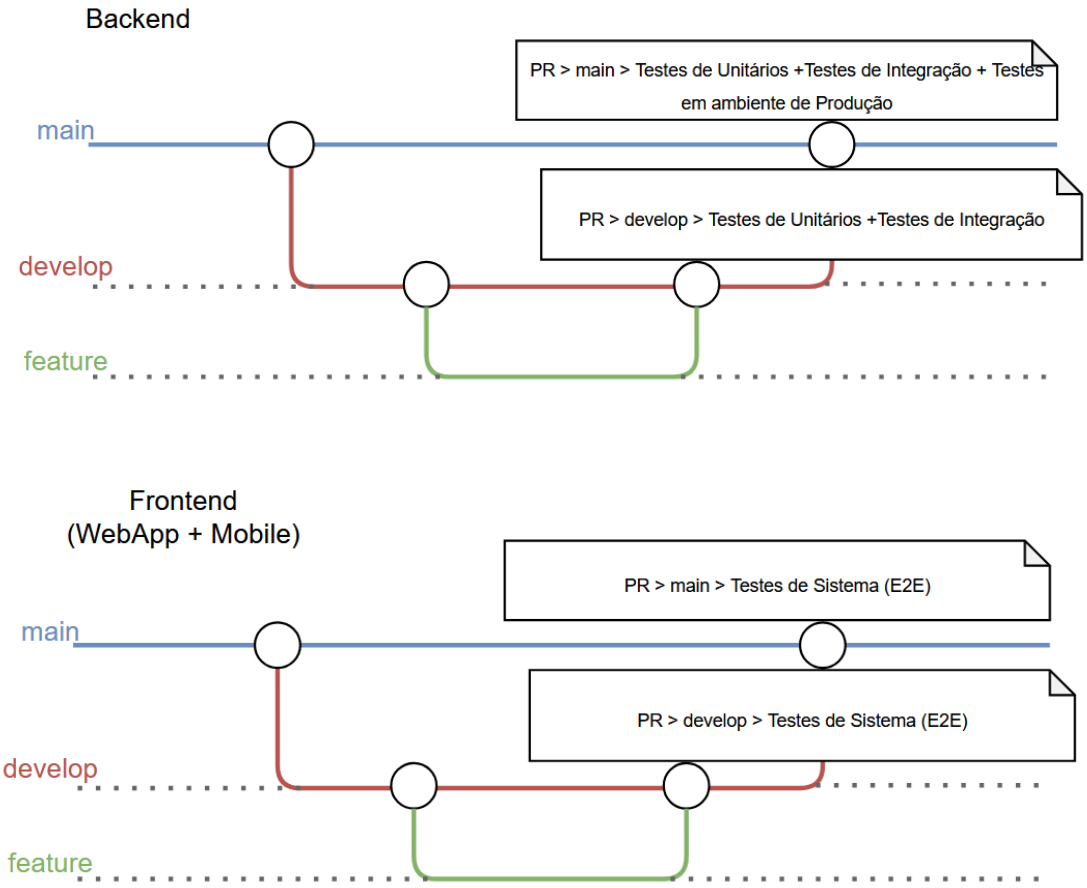
Camada do Sistema	Branch	Função	Tipo de Testes Executados
backend	**feature/**	Desenvolvimento de novas funcionalidades isoladas.	Nenhum teste automatizado. O código é validado localmente pelo programador antes de integração.
	develop	Integração contínua e validação conjunta do código.	Testes unitários, e de integração assegurando a correta interação entre componentes e a estabilidade global.
	main	Versão estável e pronta para entrega ou demonstração.	Testes unitários, de integração e ainda uns breves testes em ambiente de produção, garantindo a conformidade total antes da publicação.
frontend	**feature/**	Desenvolvimento de novas funcionalidades na interface WebApp ou Mobile.	Nenhum teste automatizado. O código é validado localmente pelo programador antes de integração.
	develop	Integração contínua e validação conjunta do frontend.	Testes de sistema (E2E) automatizados, garantindo o correto funcionamento das funcionalidades desenvolvidas.
	main	Versão final pronta para entrega e demonstração.	Testes de sistema (E2E) automatizados, simulando cenários reais de utilização e garantindo a estabilidade da versão final.

As figuras seguintes apresentam a integração dos diferentes tipos de teste no fluxo de desenvolvimento e nas pipelines de CI/CD do projeto SeePaw, diferenciando o comportamento entre o backend e o frontend (web e mobile).

No backend, a pipeline garante a execução automática de testes unitários e de integração em cada *pull request* para a branch develop, assegurando a qualidade do código antes da sua integração. Quando o código é promovido para a branch main, são novamente executados testes unitários, de integração e testes

em ambiente de produção, de modo a garantir a estabilidade e o correto funcionamento da versão estável.

No frontend, que engloba a aplicação WebApp e a Mobile, são realizados apenas testes de sistema (E2E), automatizados e integrados na pipeline. Estes testes simulam cenários reais de utilização e validam o comportamento das funcionalidades implementadas. O fluxo de integração segue a mesma lógica: as *feature branches* são validadas na develop, enquanto as versões finais são testadas integralmente na main antes da publicação.



1.5 Quality Gates

Os *Quality Gates* são verificações automáticas de qualidade que atuam nas pipelines de integração contínua, assegurando que apenas código devidamente testado e conforme aos critérios definidos é integrado nas branches principais do projeto.

A tabela seguinte apresenta os principais critérios de verificação de qualidade aplicados durante o processo de integração contínua. Estes critérios definem os requisitos mínimos que o código deve cumprir antes de ser integrado.

Tipo de Critério	Condição Mínima	Aplicação
Execução de Testes	Todos os testes devem passar com sucesso	develop, main
Cobertura de Código	Cobertura >= 80 %	develop, main
Inspeção de código	< 10 avisos não críticos Complexidade ciclomática < 15	develop, main
Erros Críticos	Nenhum erro de compilação, ou build	develop, main
Testes E2E	≥ 90% de sucesso (quando aplicável)	develop, main