1 — Workflow Geral (Kanban + transições)

Columns / States (nomes em inglês — transições em português)

1. Backlog
2. Selected for Sprint
3. In Progress
4. In Review
5. Ready for Testing
6. In Testing
7. Final Approval
8. Blocked
9. Done

Fluxograma principal — transições (em português)

1. **Backlog → Selected for Sprint**
   Transição: "Item priorizado e preparado para entrar na sprint."
   Condição: PO aprova prioridade; item estimado e com Definition of Ready.
2. **Selected for Sprint → In Progress**
   Transição: "Trabalho iniciado pelo membro responsável."
   Condição: Dev atribuído; ambiente configurado; branch criada.
3. **In Progress → In Review**
   Transição: "Implementação concluída; enviar para revisão de código ou revisão técnica."
   Condição: Todos os requisitos implementados; build local OK; testes unitários iniciais OK.
4. **In Review → Ready for Testing**
   Transição: "Revisão aprovada; código integrado e pronto para QA."
   Condição: Pull request aprovado; merge em branch de integração; release note mínimo.
5. **Ready for Testing → In Testing**
   Transição: "QA inicia execução de casos de teste e testes de aceitação."
   Condição: Ambiente de testing disponibilizado; dados de teste definidos.
6. **In Testing → Final Approval**
   Transição: "Testes passados; aguardando validação final (PO/Stakeholder)."

Condição: Todos os casos críticos passam; evidências anexadas ao caso de teste.

7. **Final Approval → Done**
   Transição: "PO aceitou; item concluído e fechado."
   Condição: Aceite formal via comentário no ticket; release agendado/deploy realizado.

Transições de tratamento de exceções / ciclo de correção

- **In Testing → In Progress**
  Transição: "Falha nos testes; retornar para correção pelo dev."
  Ação: QA cria bug linked; atribui ao responsável; descreve repro e logs.
- **In Review → In Progress**
  Transição: "Revisão exige mudanças; retornar ao dev."
  Ação: Reviewer adiciona comentários no PR e lista subtasks.
- **Final Approval → In Progress**
  Transição: "PO recusou / solicitou alterações; voltar ao desenvolvimento."
  Ação: PO cria nota de alteração com critérios novos.
- **Any (qualquer estado) → Blocked**
  Transição: "Bloqueio identificado — registrar impedimento e responsável."
  Ação: Definir impedimento, tempo estimado de resolução e owner.
- **Blocked → In Progress**
  Transição: "Impedimento resolvido; retomar trabalho."
- **Done → Reopen (opcional)**
  Transição: "Se surgir novo problema após entrega, reabrir o item."
- **Selected for Sprint → Backlog**
  Transição: "Remover da sprint (despriorizar / adiar)."

---

2 — Fluxo específico para Bugs (Bug lifecycle)

Este fluxo define como um defeito é tratado desde a deteção até ao encerramento.

Estados (bug-specific): New → Triaged → Confirmed → In Progress → In Review → Ready for Testing → Verified → Closed → Reopened (opcional)
Outros: Blocked (p/ dependências externas), Deferred (baixa prioridade)

Transições e descrições (português)

1. **New → Triaged**
   Transição: "Bug reportado; triagem inicial para validar informação e

prioridade."
Ações: QA/developer confirma repro steps; adiciona severidade e ambiente.

2. **Triaged → Confirmed**
   Transição: "Bug repro; prioridade e severidade definidas."
   Ações: Anexar logs, screenshots, passos de reprodução.

3. **Confirmed → In Progress**
   Transição: "Trabalho de correção iniciado pelo dev."
   Ações: Criar branch de correção (hotfix), referenciar ticket original.

4. **In Progress → In Review**
   Transição: "Correção implementada; enviar para revisão."
   Ações: Anexar unit tests que cobrem o bug; mensagem clara no PR.

5. **In Review → Ready for Testing**
   Transição: "Revisão aprovada; código integrado."
   Ações: Deploy em ambiente de teste e notificar QA.

6. **Ready for Testing → Verified**
   Transição: "QA executou casos de teste relacionados; bug verificado como corrigido."
   Ação: QA adiciona evidências e fecha o teste de regressão.

7. **Verified → Closed**
   Transição: "PO/Stakeholder confirma resolução e aceita o patch."
   Ação: Ticket fechado; release nota atualizada.

8. **Verified → Reopened**
   Transição: "Correção não resolveu totalmente; reabrir para investigação."

Regras adicionais para Bugs

- **Severidades:** Critical, High, Medium, Low — definidas na triagem.
- **SLA interno:** Critical → 24h; High → 72h (exemplo; ajustar conforme projeto).
- **Linkagem obrigatória:** Bug deve linkar a US afetada e ao release associado.
- **Releases de hotfix:** Hotfixs passam por checklist mínimo (testes automatizados + smoke tests).
- **Regressão:** Para qualquer bug crítico, executar suite de regressão relacionada.

---

3 — Documento: ÉPICO 1 — AUTENTICAÇÃO E CADASTRO (resumo)

Projeto: SwagLabs Shopping – Plataforma Web de E-Commerce
Metodologia: Scrum
Ambiente: Jira + Confluence
Responsável: Product Owner

Visão do Épico: Garantir que utilizadores podem criar contas, autenticar-se e recuperar acessos de forma segura, simples e consistente, reduzindo abandono, fraudes e suporte manual.

Objetivos: reduzir barreiras, proteger dados, permitir autosserviço, recuperação segura.

Regras de negócio (sintético): email único, validação forte de senha, gestão de sessões, bloqueio após N tentativas, logging de segurança.

---

4 — User Stories (desenvolvidas conforme pedido)

Vou apresentar 2 User Stories (US 1.1 e US 1.2) reescritas com: depois do Valor de Negócio vem **Visão do Cliente**; requisitos **muito** detalhados; critérios em formato de **cenários**; subtasks especificadas.

US 1.1 — Login

Key Details

Título: US 1.1 — Login
Descrição resumida: Como utilizador registado, quero iniciar sessão com minhas credenciais para aceder à plataforma e realizar compras.

Valor de Negócio: Permite acesso seguro às funcionalidades internas, reduz churn e garante controlo de sessão.

Visão do Cliente:
"O cliente espera iniciar sessão rapidamente com email/senha ou opção social, receber feedback claro em caso de erro, e sentir confiança que a sessão é segura (HTTPS, indicação visual)."

Requisitos (detalhados e específicos)

1. Página de login responsiva (desktop/mobile) com foco acessível (tab order, labels, aria-labels).

2. Campos obrigatórios: **Email** (campo tipo email, maxlength 254) e **Password** (campo tipo password, maxlength 128).
3. Validação cliente:
    - Email: formato correto; domínio permitido qualquer.
    - Password: não validar força no login, apenas presença.
    - Mostrar mensagem inline para cada campo inválido.
4. Validação servidor:
    - Verificar credenciais contra base hashed (bcrypt/argon2).
    - Resposta genérica em caso de credenciais inválidas: “Email ou password incorretos”.
5. Segurança:
    - Comunicação via HTTPS; remove headers sensíveis.
    - Limitar tentativas: bloqueio temporário após 5 tentativas num intervalo de 15 minutos.
    - Registar tentativa falhada com IP e timestamp.
    - Implementar proteção CSRF (token) e rate limiting.
6. Fluxo de sessão:
    - Criar token JWT com expiração definida (ex.: 1h) + refresh token seguro (opcional).
    - Armazenamento do token no cookie httpOnly, Secure, SameSite=strict.
7. Opções UI adicionais:
    - Link “Esqueci a password”.
    - Checkbox “Manter sessão” (persistência com refresh token).
    - Botões de login social (externos) com redirecionamento OAuth (Google/GitHub) — opcional e documentado.
8. Feedback e UX:
    - Loader ao submeter.
    - Mensagem clara para bloqueio por excesso de tentativas com instruções (ex.: aguardar X minutos).

    - Erros devem evitar revelar qual campo está errado para não divulgar existência de emails.

9. Logging & auditoria:
    - Guardar logs de login bem-sucedido e falhado (não armazenar password plain).

10. Testes esperados:

- Unit tests para validação de input e serviços de autenticação.
- Integration tests cobrindo sucesso, falha e bloqueio por tentativas.

Critérios de Aceite — Cenários (Gherkin / Cenário)

Cenário 1 — Login bem-sucedido com credenciais válidas
Dado que um utilizador existe com email "user@example.com" e password válida
Quando o utilizador preenche o email e password corretos e clica em "Entrar"
Então o utilizador é redirecionado para a página inicial e recebe um cookie httpOnly de sessão

Cenário 2 — Credenciais inválidas
Dado que não existe correspondência para as credenciais fornecidas
Quando o utilizador submete email/password incorretos
Então é apresentado o erro "Email ou password incorretos" e nenhuma sessão é criada

Cenário 3 — Bloqueio após tentativas múltiplas
Dado que o utilizador fez 5 tentativas de login falhadas nos últimos 15 minutos
Quando fizer a 6ª tentativa
Então o sistema bloqueia o acesso por 15 minutos e mostra mensagem de bloqueio

Cenário 4 — Campos obrigatórios
Dado que o formulário está vazio
Quando o utilizador submete sem preencher campos
Então são mostradas mensagens inline de validação para cada campo obrigatório

Subtasks (detalhadas)

1. (FE) Criar layout responsivo da página de login com ARIA e validações cliente.
2. (FE) Implementar UX: loader, mensagens inline, checkbox "Manter sessão" e links.
3. (BE) Implementar endpoint POST /auth/login — validação, hashing, tokens.

4. (BE) Implementar limitação de tentativas por IP e conta; logging de eventos.
5. (BE/Infra) Configurar cookies seguros (httpOnly, Secure, SameSite).
6. (QA) Escrever e executar testes unitários e de integração para autenticação.
7. (SEC) Revisão de segurança (pen test básico, verificar cabeçalhos).
8. (DOC) Atualizar documentação de API e security checklist.

Componentes Afetados: Authentication, Frontend (Login), Backend (Sessions), Security.

Prioridade: Alta — **Story Points:** 3

US 1.2 — Cadastro de Novo Utilizador

Key Details

Título: US 1.2 — Cadastro de Novo Utilizador
Descrição resumida: Como visitante, quero criar uma conta para aceder funcionalidades exclusivas.

Valor de Negócio: Aumenta aquisição de utilizadores e retenção via conta personalizada.

Visão do Cliente:
"O utilizador pretende criar conta de modo claro e rápido, com feedback sobre erros, confirmação de criação e orientação sobre requisitos de password. Quer sentir confiança que dados sensíveis são protegidos."

Requisitos (detalhados e específicos)

1. Página/formulário de registo responsivo com campos obrigatórios:
    - **Full Name** (min 2 chars, max 100, only letters and spaces allowed),
    - **Email** (valid email, maxlength 254),
    - **Password** (min 8 chars, max 128; must contain upper, lower, number, symbol) — ex.: política configurável,
    - **Confirm Password** (must match),
    - *Opcional:* CAPTCHA (reCAPTCHA v3) para prevenir bots.
2. UX e validação:
    - Validação inline na escrita para email e força de password.
    - Visual de sucesso (toastr/modal) após criação.

3. Backend:
    - Armazenar password hasheada com salt (bcrypt/argon2) e nunca logar password plain.
    - Verificar unicidade de email (index única na DB).
    - Endpoint POST /users com resposta padronizada e código 201.
4. Verificação de email:
    - Enviar email com token temporário (expira em 24h) para confirmar conta.
    - Token único; endpoint GET /users/confirm?token=.
    - Não permitir login antes da confirmação (ou permitir login limitado com aviso).
5. Segurança:
    - Rate limiting por IP para registos.
    - Proteção contra injeção e validação server-side.
6. Dados adicionais:
    - Armazenar metadata: created_at, source (web/mobile), IP de registo.
7. GDPR / LGPD:
    - Checkbox de aceitação de termos (obrigatório); link para política de privacidade.
    - Permitir remoção de conta / dados mediante pedido.
8. Testes:
    - Unit & integration tests para fluxo de criação, confirmação por email e handling de tokens expirados.
9. Logging & monitorização:
    - Registar eventos de criação, falha por email duplicado e confirmação.

Critérios de Aceite — Cenários

Cenário 1 — Cadastro bem-sucedido
Dado que o visitante preenche nome, email válido e password que cumpra a política

Quando submete o formulário de registo
Então a conta é criada, o utilizador recebe um email de confirmação e o status da conta é "PendingConfirmation"

Cenário 2 — Email já existente
Dado que o email já existe no sistema
Quando o visitante tenta registar com esse email
Então o sistema retorna erro 409 com mensagem "Email já registado"

Cenário 3 — Password não cumpre política
Dado que a password não cumpre os requisitos mínimos
Quando submete o formulário
Então é apresentada mensagem específica indicando os critérios não cumpridos

Cenário 4 — Confirmação de email com token expirado
Dado que o token de confirmação expirou (mais de 24h)
Quando o utilizador tenta confirmar via link
Então o sistema informa “Token expirado” e oferece reenvio de token

Subtasks (detalhadas)

1. (FE) Criar layout de registo com validações inline e UX (tooltips de requisitos de password).
2. (FE) Implementar reCAPTCHA (opcional) e mensagens de sucesso/erro.
3. (BE) Endpoint POST /users — criar utilizador, validar input, hashear password.
4. (BE) Endpoint POST /users/confirm e serviço de geração e validação de tokens.
5. (Infra/Email) Configurar serviço de email (template de confirmação e fila).
6. (BE) Implementar verificação de email único (index DB) e tratamento de 409.
7. (QA) Criar testes unitários e integração (incluindo token expirado).
8. (SEC) Revisão de requisitos de privacidade e armazenagem de dados.
9. (DOC) Atualizar API docs e fluxogramas.

Componentes Afetados: Users, Backend, Email Service, Frontend (Signup).
Prioridade: Alta — **Story Points:** 5

---

5 — Mind-map (textual) — para US 1.2 (exemplo)

Mind-map (nó raiz): Cadastro de Novo Utilizador

- Formulário
    - Full Name (validação)
    - Email (validação de formato)
    - Password (política de força)
    - Confirm Password
    - Checkbox termos & privacidade
    - CAPTCHA (opcional)
- Backend
    - Endpoint POST /users
    - Validação server-side
    - Hashing de password
    - Index email único
- Email Confirmation
    - Gerar token (UUID + expiry 24h)
    - Template email (link único)
    - Endpoint de confirmação
    - Reenvio de token
- Segurança
    - Rate limit
    - Proteção CSRF/Injection
    - Logging de eventos
- UX
    - Mensagens inline
    - Loader e feedback
    - Redirecionamento pós-registro
- Testes
    - Unit tests (validações)

- Integration tests (fluxo completo)
- Testes de token expirado

---

6 — Casos de Teste (Zephyr Scale / formato profissional)

Cada caso inclui: Test Key (simulado), Title, Type, Priority, Precondition, Steps (numerados), Expected Result, Attachments/Evidences, Automation Candidate, Linked Requirements/US.

Test Case 1 (Step-by-Step)

Test Key: TC-AUTH-001
Title: Login com credenciais válidas (fluxo completo)
Type: Functional / Regression
Priority: High
Preconditions: Existe utilizador registado e confirmado com email user@example.com e password P@ssw0rd123. Ambiente de teste disponível, cookies habilitados.
Steps:

1. Aceder à página /login.
2. Introduzir user@example.com no campo Email.
3. Introduzir P@ssw0rd123 no campo Password.
4. Marcar/desmarcar “Manter sessão” (opcional).
5. Clicar no botão “Entrar”.
   Expected Result:

- O utilizador é autenticado; redirecionado para dashboard/landing page; cookie httpOnly de sessão é criado; código HTTP 200/302 recebido.
  Evidences: Screenshot da página inicial pós-login; cabeçalhos da resposta (sem token no corpo).
  Automation candidate: Sim (automatizável com Selenium / Cypress).
  Linked To: US 1.1

Test Case 2 (Step-by-Step)

Test Key: TC-AUTH-002
Title: Bloqueio por tentativas inválidas (limite e mensagem)

Type: Security / Negative
Priority: Critical
Preconditions: Conta de teste brute@example.com existe. Não há bloqueios ativos para o IP.
Steps:

1. Aceder a /login.
2. Repetir 5 vezes: introduzir brute@example.com e password errada wrongpass e submeter.
3. Na 6ª tentativa, submeter novamente credenciais erradas.
   Expected Result:

- Após 5 tentativas falhadas, 6ª tentativa resulta em mensagem de bloqueio: “Conta temporariamente bloqueada por 15 minutos” (ou similar). Registo de evento de bloqueio com IP aparece nos logs.
  Evidences: Logs de autenticação; screenshots do erro; timestamp.
  Automation candidate: Parcial (simular tentativas via scripts, mas depende do throttle do ambiente).
  Linked To: US 1.1, Security Requirements.

---

7 — Casos BDD (formato Gherkin) — 2 exemplos

BDD 1 — Login Successful

Feature: User Login

Scenario: User logs in with valid credentials

Given the user "user@example.com" exists and is confirmed

And the user is on the Login page

When the user enters valid email and password

And clicks "Entrar"

Then the user should be redirected to the home page

And a httpOnly session cookie should be presente

BDD 2 — Signup and Confirm Email

Feature: User Signup and Email Confirmation

Scenario: New user signs up and confirms email within token validity

Given the visitor is on the Signup page

When the visitor fills the form with valid name, email and password

And submits the registration form

Then the system should create a user with status "PendingConfirmation"

And send an email with a confirmation token

When the user clicks the confirmation link before token expiry

Then the user's status should change to "Active" and the user can authenticate

---

8 — Mapeamento para Zephyr Scale / Estrutura de Testes sugerida

- **Test Cycle:** Sprint X — Regression: Auth
    - TC-AUTH-001 (positive)
    - TC-AUTH-002 (negative / security)
    - TC-AUTH-003 (signup happy path) — (podes criar baseado na US 1.2)
    - TC-AUTH-004 (signup token expired) — teste de rejeição

Para cada Test Case no Zephyr/Scale inclua: Precondition, Steps, Expected Result, Test Data (email, passwords), Attachments (screenshots/logs), Test Type (Manual/Automated), Automation Script link (se existir).