Plano de testes

Estratégia

- Unitários:
    - Funções de formatação e agregação do dashboard (formatNumber, formatBudget, contagem por status/gerência).
    - Tratamento de datas e seleção de perfis em cartões/detalhes de projeto.
- Integração (API): validar respostas, códigos de status e fallback de cada rota (/api/projects, /api/projects/{id}, /api/export/excel, /api/requests, /api/gamification, /api/ai/chat).
- E2E (Playwright/Cypress): cenários completos para UC-01…UC-09 (autenticação, filtros de projetos, exportação, envio de solicitações, conclusão de etapa, chat).
- Usabilidade: avaliar feedback visual dos formulários (mensagens de sucesso/erro, estados desabilitados, spinner).
- Desempenho: medir tempo de carregamento das listas (projetos, trilhas) e custo da exportação em ambientes com Supabase populado; testar fallback com mocks para ambientes offline.

Critérios de aceitação por requisito

- UC-01: login aceita credenciais válidas, bloqueia sem Supabase ou senha incorreta; redireciona para /projects.
- UC-02: dashboard exibe métricas corretas ou zeros em caso de erro; gráficos mostram quantidades esperadas.
- UC-03: filtros retornam subconjunto adequado e mensagens de estado (carregando/erro/sem resultados).
- UC-04: detalhes exibem participantes e datas formatadas; 404 para ID inexistente; 401 protegido quando Supabase ativo.
- UC-05: exportação gera arquivo com colunas previstas e 200 OK; erros exibem alerta amigável.
- UC-06: campos obrigatórios validados no cliente e servidor; feedback de sucesso/erro visível.
- UC-07: trilhas e badges carregam quando dados existem; mensagens adequadas quando não há conteúdo ou Supabase desativado.
- UC-08: conclusão de etapa atualiza progresso e badges; usuários não logados recebem bloqueio; erros de backend tratados.

- UC-09: chat alterna estados, registra mensagens, mostra falhas de rede ou falta de chave API.

Dados de teste

- Mocks internos: utilizar mockProjects, mockMetrics, mockInnovationTrails, mockBadges para cenários offline e comparações de resposta.
- Supabase: preparar fixtures nas tabelas citadas (projects, participants, innovation_trails, trail_stages, badges, user_gamification_progress) e garantir que a RPC complete_stage_and_update_progress exista conforme esperado.
- OpenRouter: chave de teste com limites conhecidos; simular respostas de erro (HTTP 4xx/5xx) para validar UX.

Ambiente

- Local: Node 20+, pnpm install, variáveis .env.local (NEXT_PUBLIC_SUPABASE_URL, NEXT_PUBLIC_SUPABASE_ANON_KEY, OPENROUTER_API_KEY).
- CI/CD: pipeline recomendado executando pnpm lint e pnpm build antes do deploy Vercel (conforme orientação do README).
- Dependências externas: instância Supabase com RLS adequado e OpenRouter habilitado.

Métricas e critérios de saída

- Cobertura: alvo ≥80% para unidades/critérios críticos (dashboard, APIs sensíveis).
- Qualidade: zero erros críticos de lint (pnpm lint) e build (pnpm build).
- Performance: tempos de resposta aceitáveis (p.ex. <2s para /api/projects com 100 registros; <5s para export).
- UX: todos os fluxos possuem mensagens de sucesso/erro verificadas manualmente.
- Saída: testes unitários/integrados/e2e aprovados; regressões bloqueadas; aprovação manual do PMO para exportações e dados sensíveis.