Plano de Testes – StudyHubBackend (Testes de API)

Versão: v1.2 | Data: 08/02/2026

Objetivos e Propósito

Este documento descreve o plano de testes do backend do **StudyHub**, plataforma colaborativa para estudantes, que permite criar, organizar e compartilhar **cards**, **folders** e **summaries**. O foco é validar a API REST: autenticação JWT, regras de acesso (Professor/Aluno), persistência no PostgreSQL, e consistência das respostas HTTP (códigos e formato).

Objetivos do plano: (1) listar requisitos de teste, (2) descrever estratégia e técnicas, (3) identificar recursos necessários, (4) listar artefatos, (5) listar tarefas da atividade de teste.

1. Requisitos (Alvos para teste)

Rotas fornecidas:
• USERS: POST /api/users, POST /api/users/login, GET /api/users, GET /api/users/:id, PUT /api/users/:id, DELETE /api/users/:id.
• CARDS: POST /api/cards (authenticate + onlyTeacher), GET/PUT/DELETE /api/cards/:id, GET /api/cards/disciplina/:disciplina.
• FOLDERS: GET /api/folders, GET /api/folders/:id, POST/PUT/DELETE /api/folders (authenticate).
• SUMMARIES: GET /api/summaries, GET /api/summaries/:id, POST/PUT/DELETE /api/summaries (authenticate).
Segurança: rotas protegidas por *authenticate* (JWT).
Regras de acesso: Professor pode criar/editar/deletar conteúdos; Aluno apenas visualiza.

2. Estratégia

Abordagem: testes de integração de API (endpoint + middleware + controllers/services + banco).
Ferramentas: Mocha + Chai (asserções), Supertest (HTTP). Teste manual via Swagger/Insomnia para evidências.
Critério de conclusão: 100% dos testes críticos aprovados (auth + 401/403 + CRUD). Ausência de erros 5xx nos fluxos esperados.

3. Ambiente e dados de teste

• PostgreSQL de teste isolado (schema limpo).
• Seeds/fixtures: 1 Professor e 1 Aluno + pelo menos 2 cards, 2 folders, 2 summaries para cenários de leitura e 404.
• Rotina de limpeza: truncar tabelas ou usar transações/rollback a cada suite.

4. Matriz de permissões (Professor x Aluno)

Recurso	Ação	Professor	Aluno
Users	Criar (cadastro)	■	■
Users	Login	■	■
Users	Listar/Buscar/Editar/Deletar	■ (com token)	■ (com token)

Cards	Criar	■	■ (403)
Cards	Ler (getById / disciplina)	■	■
Cards	Editar/Deletar	■ (regra)	■ (403 - esperado)
Folders	Criar/Editar/Deletar	■ (regra)	■ (403 - esperado)
Folders	Ler	■	■
Summaries	Criar/Editar/Deletar	■ (regra)	■ (403 - esperado)
Summaries	Ler	■	■

5. Contrato de erros (recomendação)

Para facilitar testes e debug, recomenda-se padronizar respostas de erro, por exemplo:

```
{ "message": "...", "details": ["..."] }
```

E garantir status consistentes: 400/422 validação, 401 autenticação, 403 autorização, 404 não encontrado.

6. Tipos de Teste – Funcionalidades (API)

Cada caso de teste descreve objetivo, técnica, critério de completude e considerações especiais, no estilo solicitado.

Objetivo

Cadastrar usuário (Professor/Aluno).

Técnica

Requisição via Supertest (automatizado) e/ou Swagger (manual).

Critérios de completude

Retornar usuário criado e status **201 Created**. Não retornar senha no corpo.

Considerações Especiais

Rota: **POST /api/users**.
Testar também campos faltando (esperado 400/422) e email duplicado (409/400).

Exemplo de payload

```
{ "name": "Ana", "email": "ana@email.com", "password": "123456", "role": "TEACHER" }
```

Objetivo

Login do usuário e geração de JWT.

Técnica

Supertest/Swagger: enviar credenciais válidas.

Critérios de completude

Retornar **200 OK** com token JWT (string).

Considerações Especiais

Rota: **POST /api/users/login**.
Testar credenciais inválidas (esperado 401).

Exemplo de payload

```
{ "email": "ana@email.com", "password": "123456" }
```

Objetivo

Acessar rota protegida sem token.

Técnica

Supertest: omitir header Authorization.

Critérios de completude

Retornar **401 Unauthorized** e mensagem de erro.

Considerações Especiais

Rota: **GET /api/users (ou qualquer rota com authenticate)**.

Objetivo

Criar card como Professor.

Técnica

Supertest com Authorization: Bearer .

Critérios de completude

Retornar card criado e **201 Created**.

Considerações Especiais

Rota: **POST /api/cards**.
Validar que Aluno recebe 403 (onlyTeacher).

Exemplo de payload

```
{ "titulo": "Árvore Binária", "disciplina": "ED", "conteudo": "Resumo..." }
```

Objetivo

Buscar card por id.

Técnica

Supertest com token válido.

Critérios de completude

Se existir: **200 OK** + card. Se não existir: **404 Not Found**.

Considerações Especiais

Rota: **GET /api/cards/:id**.

Objetivo

Editar e deletar card respeitando regra de acesso.

Técnica

Supertest: PUT/DELETE com tokenProfessor e tokenAluno.

Critérios de completude

Professor: **200 OK** no PUT e **204** no DELETE. Aluno: **403 Forbidden** (esperado).

Considerações Especiais

Rota: **PUT /api/cards/:id e DELETE /api/cards/:id**.
Se Aluno conseguir alterar (200/204), registrar como defeito: falta de onlyTeacher nessas rotas.

Objetivo

Listar cards por disciplina.

Técnica

Supertest com token válido.

Critérios de completude

Retornar **200 OK** e lista filtrada (ou vazia se não houver).

Considerações Especiais

Rota: **GET /api/cards/disciplina/:disciplina**.
Testar disciplina com espaços/acentos (urlencode) e disciplina sem cards (lista vazia).

Objetivo

CRUD de folders (com permissão).

Técnica

Supertest com tokenProfessor/tokenAluno.

Critérios de completude

GET: 200 OK. POST/PUT/DELETE: Professor ok; Aluno deve retornar 403 (esperado pela regra).

Considerações Especiais

Rota: **GET /api/folders, POST /api/folders, PUT/DELETE /api/folders/:id**.
Se não houver onlyTeacher, os testes ajudam a detectar inconsistência com o requisito.

Objetivo

CRUD de summaries (com permissão).

Técnica

Supertest com tokenProfessor/tokenAluno.

Critérios de completude

GET: 200 OK. POST/PUT/DELETE: Professor ok; Aluno deve retornar 403 (esperado pela regra).

Considerações Especiais

Rota: **GET /api/summaries, POST /api/summaries, PUT/DELETE /api/summaries/:id**.

7. Casos de Teste (Resumo por rota)

ID	Rota	Cenário	Técnica	Esperado
AUTH-01	POST /api/users/login	Login válido	Supertest	200 + token
AUTH-02	POST /api/users/login	Login inválido	Supertest	401
AUTH-03	Rota protegida	Sem token	Supertest	401
USR-01	POST /api/users	Cadastro válido	Supertest	201
USR-02	GET /api/users	Listar com token	Supertest	200
USR-03	GET /api/users/:id	Buscar inexistente	Supertest	404
CARD-01	POST /api/cards	Criar (Professor)	Supertest	201
CARD-02	POST /api/cards	Criar (Aluno)	Supertest	403
CARD-03	GET /api/cards/disciplina/:disc	Filtrar por disciplina	Supertest	200
FOLD-01	GET /api/folders	Listar	Supertest	200
FOLD-02	POST /api/folders	Criar (prof/aluno)	Supertest	201/403
SUM-01	GET /api/summaries	Listar	Supertest	200
SUM-02	POST /api/summaries	Criar (prof/aluno)	Supertest	201/403

8. Teste de Performance

Objetivo: medir tempo de resposta do login e endpoints GET.
Técnica: executar 10 vezes e registrar média/pior caso (Insomnia ou script).
Critério: média <= 3s (ou requisito da disciplina).

9. Teste de Carga

Objetivo: verificar responsividade com 30+ registros por entidade.
Técnica: seed/popular banco; repetir GETs e filtros; registrar tempos.
Critério: sem erros 5xx e sem degradação relevante.

10. Tolerância a falhas / Recuperação

Objetivo: garantir que dados persistidos não se perdem após reinício do servidor.
Técnica: interromper/reiniciar o backend e validar dados no banco.
Critério: consistência mantida; erros controlados quando houver requisição interrompida.

11. Análise Estática

Técnica: TypeScript strict, ESLint, build; opcional SonarLint/CodeQL.
Critério: zero erros de compilação e sem falhas críticas de lint.

12. Cronograma (modelo)

Tarefa	Data de início	Data de entrega
Planejamento	—	—
Configuração do ambiente	—	—
Implementação dos testes	—	—
Execução	—	—
Avaliação e Relatório	—	—

13. Relatório de Resultados (modelo)

Após a execução, registrar por módulo (Users/Cards/Folders/Summaries): quantos casos passaram/falharam, evidências (prints/logs), e defeitos. Exemplo de defeito a detectar: endpoints PUT/DELETE de conteúdo sem *onlyTeacher*, permitindo alteração por Aluno.