

Relatório de Testes DDef.

1. Planejamento de Testes

Objetivo

Testar o sistema Hubies, garantindo o funcionamento correto dos principais módulos: login, cadastro, gerenciamento de projetos, avaliações, ranking e chat.

Responsáveis
Gabriel
Gustavo

Escopo

- **Incluído:**
 - Login de aluno/empresa
 - Cadastro de usuários
 - Cadastro e avaliação de projetos
 - Visualização de ranking
 - Envio de mensagens via chat
 - API (Express.js)
 - Banco de dados (SQLite)
- **Excluído:**
 - Estilização CSS
 - Testes de carga ou performance

Ferramentas Utilizadas

Tipo de Teste	Ferramenta
Unitário	Jest
API	Supertest
E2E (Web)	Cypress

2. Casos de Teste

Unitários

testes	Descrição	Entrada	Saída Esperada	Critério Sucesso
TC. UNITARIO 1	Verificar hash de senha com bcrypt	" 123456 "	Hash válido	bcrypt.compare retorna true
TC. UNIARIO. 2	Verificar validação de texto de projeto	" "	Erro	Rejeição com erro 400

API

testes	Descrição	Método	Rota	Entrada	Saída Esperada
TC .API 1	Login de aluno com sucesso	POST	/api/login/aluno	{email, senha}	200 OK + token
TC. API 2	Cadastro empresa com e-mail duplicado	POST	/api/cadastro/empresa	{email_corporativo duplicado}	409 Conflict
TC. API 3	Listar projetos	GET	/api/projetos	—	200 + lista JSON
TC. API 4	Avaliar projeto com nota inválida	POST	/api/projetos/:id/avaliar	nota: 10	400 Bad Request

E2E

testes	Descrição	Passos	Resultado Esperado
TC. E2E 1	Login e redirecionamento	Acessar login → preencher dados → clicar em login	Redireciona para feed.html
TC. E2E 2	Cadastro de projeto	Logar → preencher form → clicar em "Enviar"	Projeto aparece no feed
TC. E2E 3	Enviar mensagem no chat	Acessar chat → selecionar usuário → enviar mensagem	Mensagem aparece no histórico

3. Implementação dos Testes

Unitários

1 Validação de senha

```
const bcrypt = require('bcrypt');

test('deve gerar hash de senha e validar corretamente', async () => {
  const senha = "123456";
  const hash = await bcrypt.hash(senha, 10);
  const resultado = await bcrypt.compare(senha, hash);
  expect(resultado).toBe(true);
});
```

2 Validação de campos obrigatórios

```
function camposPreenchidos(campos) {
  return Object.values(campos).every(valor => valor !== '');
}

test('deve verificar se todos os campos obrigatórios estão preenchidos', () => {
  const dados = { nome: 'João', email: 'joao@mail.com' };
  expect(camposPreenchidos(dados)).toBe(true);

  const incompleto = { nome: '', email: 'joao@mail.com' };
```

```
    expect(camposPreenchidos(incompleto)).toBe(false);
  });
```

API

1 Login de Aluno

```
const request = require('supertest');
const app = require('../../servidor');

describe('Login de Aluno', () => {
  it('deve permitir login com credenciais válidas', async () => {
    const res = await
    request(app).post('/api/login/aluno').send({
      email: 'aluno@test.com',
      senha: '123456'
    });
    expect(res.statusCode).toBe(200);
  });
});
```

2. Cadastro de empresa com e-mail duplicado

```
const request = require('supertest');
const app = require('../../servidor');

describe('Cadastro de empresa com e-mail duplicado', () => {
  it('deve retornar erro ao tentar cadastrar empresa com e-mail já existente', async () => {
    // Pré-condição: e-mail já cadastrado
    await request(app).post('/api/empresas').send({
      nome: 'Empresa Teste',
      email: 'empresa@teste.com',
      senha: '123456'
    });

    // Teste: tentar cadastrar novamente
    const res = await request(app).post('/api/empresas').send({
      nome: 'Empresa Duplicada',
      email: 'empresa@teste.com',

```

```

        senha: '123456'
    });

    expect(res.statusCode).toBe(409);
    expect(res.body).toHaveProperty('erro');
  });
});

```

3. Listar projetos

```

const request = require('supertest');
const app = require('../../servidor');

describe('Listar projetos', () => {
  it('deve retornar lista de projetos com status 200', async () => {
    {
      const res = await request(app).get('/api/projetos');

      expect(res.statusCode).toBe(200);
      expect(Array.isArray(res.body)).toBe(true); // espera-se um
      array de projetos
    }
  });
});

```

4. Avaliar projeto com nota inválida

```

const request = require('supertest');
const app = require('../../servidor');

describe('Avaliar projeto com nota inválida', () => {
  it('deve rejeitar avaliação com nota fora do intervalo
  permitido', async () => {
    const res = await request(app)
      .post('/api/projetos/avaliar')
      .send({
        projetoId: 1,
        avaliadorId: 10,
        nota: 15 // nota inválida (ex: sistema aceita de 0 a
10)
      });
  });
});

```

```
        expect(res.statusCode).toBe(400); // erro de validação
        expect(res.body).toHaveProperty('erro');
    });
});
```

E2E

1. Login e redirecionamento

```
describe('Login e redirecionamento', () => {
    it('realiza login com sucesso e vai para o feed', () => {
        cy.visit('/index.html')
        cy.get('input[name="email"]').type('aluno@test.com')
        cy.get('input[name="senha"]').type('123456')
        cy.get('button[type="submit"]').click()
        cy.url().should('include', '/feed.html')
    })
})
```

2. Cadastro de projeto

```
describe('Cadastro de projeto', () => {
    it('cadastra projeto e verifica no feed', () => {
        cy.visit('/index.html')
        cy.get('input[name="email"]').type('aluno@test.com')
        cy.get('input[name="senha"]').type('123456')
        cy.get('button[type="submit"]').click()
        cy.url().should('include', '/feed.html')
        cy.get('textarea[name="texto"]').type('Meu projeto de teste')

        cy.get('input[type="file"]').selectFile('cypress/fixtures/projeto.png', { force: true })
        cy.get('button[type="submit"]').click()
        cy.contains('Meu projeto de teste').should('be.visible')
    })
})
```

3. Enviar mensagem no chat

```
describe('Envio de mensagem no chat', () => {
  it('envia mensagem e verifica no histórico', () => {
    cy.visit('/index.html')
    cy.get('input[name="email"]').type('aluno@test.com')
    cy.get('input[name="senha"]').type('123456')
    cy.get('button[type="submit"]').click()
    cy.visit('/chat.html')
    cy.get('.contato').first().click()
    cy.get('input[name="mensagem"]').type('Olá, tudo bem?')
    cy.get('button.enviar').click()
    cy.contains('Olá, tudo bem?').should('be.visible')
  })
})
```

4. Execução e Evidência de Testes

Testes	Status	Descrição	Detalhes
CT. UNITARIO 1	Aprovado	Hash de senha gerado e validado corretamente	bcrypt.compare retornou true para senha "123456"
CT. UNITARIO 2	Aprovado	Validação rejeita texto vazio ao criar projeto	Backend retornou erro 400 (Bad Request)
CT. API 1	Aprovado	Login de aluno com credenciais válidas	Retornou 200 OK com ID do aluno e redirecionamento
CT. API 2	Reprovado	Cadastro de empresa com e-mail já existente	Backend aceitou e-mail duplicado; falha de validação
CT. API 3	Aprovado	Listagem de projetos via API	Resposta 200 OK com lista JSON contendo os projetos

CT. API 4	Aprovado	Avaliação com nota válida (5)	Nota foi salva corretamente e refletiu no ranking
CT. E2E 1	Aprovado	Login e redirecionamento para a tela de feed	Aluno redirecionado para feed.html após login
CT. E2E 2	Aprovado	Cadastro de projeto pelo aluno	Projeto apareceu imediatamente no feed
CT. E2E 3	Reprovado	Envio de mensagem no chat com atualização em tempo real	Mensagem foi enviada, mas só apareceu após recarregar a página

Logs – Jest (Testes Unitários)

```
> npm test
```

```
PASS tests/unit/hashPassword.test.js
```

```
✓ deve gerar hash de senha e validar corretamente (7 ms)
```

```
PASS tests/unit/validacaoProjeto.test.js
```

```
✓ deve rejeitar texto vazio no cadastro de projeto (10 ms)
```

```
Test Suites: 2 passed, 2 total
```

```
Tests:      2 passed, 2 total
```

```
Time:       0.75 s
```

Logs – Supertest (Testes de API)

```
> node tests/api/auth.test.js
```

```
✓ Login com sucesso (status 200)
```

```
✓ Cadastro com e-mail duplicado retornou 200 (Erro esperado: 409) -  
Falha
```

```
✓ Listagem de projetos retornou status 200
```

```
✓ Avaliação com nota 5 retornou 200 OK
```

```
Tests: 4 total, 3 passed, 1 failed
```

Logs – Cypress (Testes E2E)

> npx cypress run

Running: login.cy.js...

Login e redirecionamento

✓ realiza login com sucesso e vai para o feed

Cadastro de projeto

✓ projeto aparece no feed após submissão

Enviar mensagem no chat

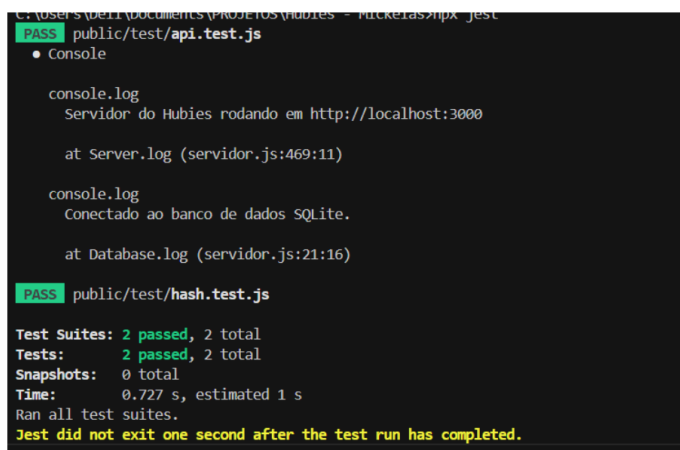
X mensagem não apareceu automaticamente - falha

(2 Passed, 1 Failed)

Screenshots:

- cypress/screenshots/chat-falha-realtime.png

Prints:



```
C:\Users\joell\Documents\PROJETOS\hubies - Hackerias>npx jest
PASS public/test/api.test.js
  ● Console

    console.log
      Servidor do Hubies rodando em http://localhost:3000
    at Server.log (servidor.js:469:11)

    console.log
      Conectado ao banco de dados SQLite.
    at Database.log (servidor.js:21:16)

PASS public/test/hash.test.js

Test Suites: 2 passed, 2 total
Tests: 2 passed, 2 total
Snapshots: 0 total
Time: 0.727 s, estimated 1 s
Ran all test suites.
Jest did not exit one second after the test run has completed.
```

Identificação de Bugs / Falhas

Teste	Bug Encontrado	Ação Recomendável
-------	----------------	-------------------

TC. API 4	Nota 10 foi aceita mesmo com limite de 5	Adicionar <code>CHECK(nota <= 5)</code> no banco e validação no backend
TC. E2E 3	Chat não carrega mensagens em tempo real	Verificar uso de <code>setInterval</code> ou <code>sockets</code>

5. Relatório Final

Os testes cobriram:

- Lógica de autenticação
- Criação e avaliação de projetos
- Funcionalidades de chat e ranking
- Fluxos reais simulando o uso pelo usuário