Universidade Católica de Brasília

Pró-Reitoria Acadêmica
Escola de Educação, Tecnologia e Comunicação
Curso de Bacharelado em Engenharia de Software
Trabalho de Disciplina de Teste de Software

Documentação Positividade+

(Definitivo)

Autores:Hudson Cunha, Gustavo Linhares, Diogo França
Orientador: Zoé Roberto Magalhães de Júnior

Brasília - DF
2025

Diogo França dos Santos

Hudson dos Santos Cunha

Gustavo Linhares dos Santos

Documentação Positividade +

Documento apresentado ao Curso de graduação de Bacharelado em Engenharia de Software da Universidade Católica de Brasília, como requisito parcial para obtenção da aprovação na disciplina de teste de software.

Orientador: Prof. Zoé Roberto Magalhães Júnior

Brasília
2025

LISTA DE FIGURAS

Figura 1 - Diagrama de Casos de Uso de *Software*. 15
Figura 2 - Diagrama de classe. 16
Figura 3 - Tela de protótipo. 17

SUMÁRIO

1. INTRODUÇÃO.....5
1.1. DIAGNÓSTICO DA EMPRESA / TEMA.....6
2. OBJETIVOS.....8
2.1. OBJETIVO GERAL.....8
2.2. OBJETIVOS ESPECÍFICOS.....8
3. PROPOSTA DO SISTEMA.....9
3.1. DESCRIÇÃO DO SISTEMA PROPOSTO.....9
3.2. RESULTADOS ESPERADOS.....9
4. FERRAMENTAS UTILIZADAS.....9
5. APLICAÇÃO DA ISO 25010.....9
6. ANÁLISE DE NEGÓCIO.....11
6.1. REGRAS DE NEGÓCIO.....11
7. ANÁLISE DE REQUISITOS.....13
7.1. REQUISITOS FUNCIONAIS.....13
7.2. REQUISITOS NÃO-FUNCIONAIS.....13
7.3. DIAGRAMA DE CASOS DE USO DA SOLUÇÃO.....14
7.3.1. Visão Geral dos Casos de Uso e Atores.....14
7.4. DIAGRAMA DE CLASSE.....15
7.4.1. Visão Geral do diagrama de Classe.....15
7.4.2. Protótipos de Tela do Sistema.....17
8. PLANEJAMENTO DE TESTES.....20
8.1. TESTES DE SISTEMA/INTERFACE.....21
8.2. TESTES UNITÁRIOS COM ROTAS DE API.....25
8.3. TESTES DE API.....28
8.4. TESTES DE STRESS.....31
9. ANÁLISE DE RISCO DO PRODUTO.....33
10. RESULTADO DOS TESTES.....33
10.1. TESTE DE SISTEMA/INTERFACE.....34
10.2. TESTE UNITÁRIO.....37
10.3. TESTE DE API.....39
10.4. TESTE DE STRESS.....43
11. GITHUB.....47
12. CONCLUSAO E LIÇÕES APRENDIDAS.....47

1. INTRODUÇÃO

A computação, a tecnologia da informação (TI) e a informática desempenham um papel central nas organizações modernas, sendo ferramentas essenciais para otimizar processos, aumentar a eficiência e melhorar a comunicação interna e externa. Com o crescente uso dessas tecnologias, empresas de diversos setores têm se beneficiado da digitalização de seus processos, criando novas formas de interação, automação de tarefas repetitivas e maior acessibilidade às informações. Nesse contexto, as plataformas digitais, como websites e aplicativos, têm se tornado indispensáveis para proporcionar soluções inovadoras e eficazes.

O projeto apresentado neste trabalho aborda a criação de uma plataforma digital chamada Positividade+, um site dedicado a promover mensagens positivas e motivacionais entre seus usuários. O problema identificado é a falta de espaços virtuais que incentivem a positividade e a interação construtiva nas plataformas online. A proposta do Positividade+ é criar um ambiente digital que combine entretenimento e bem-estar, ao permitir que os usuários compartilhem e interajam com mensagens de otimismo, contribuindo para um ciclo de positividade no dia a dia das pessoas. A automatização de processos como o envio de mensagens, a visualização das mais curtidas do mês e o sistema de login ajudam a tornar essa experiência mais interativa e personalizada.

Este trabalho propõe o desenvolvimento de um site interativo, onde os usuários podem criar e compartilhar mensagens positivas, curtir postagens de outros participantes e visualizar as mensagens mais populares. A plataforma é construída com foco na simplicidade e usabilidade, utilizando tecnologias web modernas como HTML, CSS e JavaScript, além de implementar funcionalidades como login, exibição de mensagens aleatórias e um sistema de curtir para as mensagens mais inspiradoras. O sistema de login e a exibição de mensagens mais curtidas são funcionalidades chave que tornam a interação mais rica e envolvente.

O trabalho é estruturado em diferentes etapas, começando com a análise do problema e o desenvolvimento da interface do usuário, passando pela implementação das funcionalidades de interação e automação, e concluindo com a avaliação dos resultados. A primeira parte descreve a estrutura geral do site, abordando as funcionalidades principais e como elas

atendem às necessidades dos usuários. Em seguida, os capítulos discutem as tecnologias utilizadas, a implementação do código-fonte e a integração dos diversos componentes do sistema, culminando com uma análise dos benefícios gerados pela automação dos processos e pela criação de uma experiência digital positiva.

1.1. DIAGNÓSTICO DA EMPRESA / TEMA

Atualmente, observa-se um cenário preocupante nas redes sociais e plataformas digitais em geral: ambientes cada vez mais carregados de negatividade, discursos de ódio, comparações tóxicas e desinformação. Essas dinâmicas têm impactado negativamente a saúde mental de muitos usuários, especialmente os mais jovens, que passam boa parte do tempo conectados. Embora existam iniciativas voltadas à saúde emocional, ainda são escassas as ferramentas que incentivam, de forma direta e acessível, a disseminação de mensagens positivas e o fortalecimento emocional dos usuários.

Nesse contexto, surge a proposta da plataforma Positividade+, um sistema que visa oferecer um ambiente leve, acolhedor e voltado exclusivamente para o compartilhamento de mensagens motivacionais, elogios, frases positivas e interações construtivas. O diferencial está na simplicidade da plataforma: usuários podem enviar mensagens anônimas ou identificadas, curtir postagens positivas e participar ativamente de uma rede que tem como propósito central o bem-estar emocional.

Atualmente, não há uma aplicação popular com foco exclusivo nesse tipo de interação. As principais redes sociais misturam conteúdos de diversas naturezas, o que dificulta a criação de um ambiente 100% positivo. Isso evidencia uma lacuna de mercado e reforça a necessidade de uma solução que promova exclusivamente interações saudáveis e motivadoras.

Portanto, o diagnóstico revela uma oportunidade clara: criar uma plataforma digital focada em positividade, com design simples e funcionalidades diretas, que possa ser usada tanto

individualmente quanto por instituições (como escolas, empresas ou grupos sociais) para estimular o bem-estar coletivo.

2. OBJETIVOS

A seguir será apresentado o objetivo geral e os objetivos específicos do software.

2.1. OBJETIVO GERAL

O site "Positividade+" foi projetado para promover mensagens e interações positivas entre os usuários. Sua principal funcionalidade é fornecer aos visitantes mensagens aleatórias de positividade para inspiração instantânea. No entanto, para interagir mais profundamente com o conteúdo e com outros usuários, é necessário fazer login.

2.2. OBJETIVOS ESPECÍFICOS

Os usuários cadastrados têm a capacidade de enviar mensagens personalizadas de positividade uns aos outros. Além disso, eles podem expressar apreço por mensagens positivas dando "like" nelas. Isso promove interações positivas e reconhecimento de conteúdo inspirador dentro da comunidade. O administrador tem permissões especiais para gerenciar o site, incluindo a capacidade de moderar o conteúdo, gerenciar contas de usuário e garantir o bom funcionamento geral da plataforma.

3. PROPOSTA DO SISTEMA

A seguir será apresentada a proposta do sistema, visando detalhar os principais pontos a serem seguidos.

3.1. DESCRIÇÃO DO SISTEMA PROPOSTO

O site "Positividade+" descrito neste documento é uma plataforma independente e auto-suficiente, projetada para promover mensagens e interações positivas entre os usuários. Ele não depende diretamente de outros sistemas externos para funcionar, mas pode se beneficiar de integrações opcionais com redes sociais ou serviços de autenticação para simplificar o processo de login e registro de usuários. Para os usuários não autenticados, o site fornece acesso limitado, permitindo-lhes visualizar mensagens de positividade aleatórias. No entanto, para interagir mais profundamente com o conteúdo e com outros usuários, é necessário fazer login. Isso implica uma área de autenticação e registro de usuários. Em resumo, o "Positividade+" é um sistema projetado para promover e facilitar interações positivas entre os usuários. Ele segue uma arquitetura cliente-servidor típica e pode integrar-se opcionalmente com outros sistemas externos, como redes sociais, para melhorar a experiência do usuário

3.2. RESULTADOS ESPERADOS

Com a implantação do site Positividade +, esperam-se os seguintes resultados:

- Fazer um site Responsivo;
- Ser atrativo para usuários e anunciantes;
- Garantir uma experiência de usuário (UX) satisfatória;
- Garantir uma interface de usuário (UI) satisfatória;
- Ter uma moderagem efetiva através do administrador;
- Disponibilidade do site 24h;

4. FERRAMENTAS UTILIZADAS

Xammp, Phpmyadmin, Nodejs, LucidChart, MySql, Vscode

5. APLICAÇÃO DA ISO 25010

Prioritários:

- Confidencialidade
- Integridade Autenticidade
- Operabilidade
- Capacidade de aprendizado
- Estética
- Disponibilidade
- Recuperabilidade
- Correção funcional
- Adequação funcional

Importantes:

- Comportamento do tempo
- Uso de recursos
- Interoperabilidade
- Modificabilidade
- Testabilidade
- Facilidade de instalação
- Tolerância a falhas
- Integridade funcional
- Adaptabilidade
- Modularidade

Baixo impacto:

- Maturidade
- Rastreabilidade de uso
- Reusabilidade
- Ausência de repúdio
- Rastreabilidade de uso
- Adequação reconhecível
- Proteção de erro do usuário
- Acessibilidade
- Capacidade
- Coexistência

- Analisabilidade
- Capacidade de substituição

6. ANÁLISE DE NEGÓCIO

Neste capítulo será descrito, através de uma tabela, o processo do negócio em que o *software* em questão será inserido, sendo estas as regras de negócio.

6.1. REGRAS DE NEGÓCIO

Número	Nome	Descrição	Setor
RN1	Autenticação obrigatória	Somente usuários logados podem acessar páginas protegidas, como `index-logged.html` e `mensagem-logged.html`.	Segurança / Acesso
RN2	Armazenamento local de sessão	O `user_id` e `username` são armazenados no `localStorage` para manter o usuário logado.	Segurança / Sessão
RN3	Envio de mensagens	Usuários logados podem enviar mensagens positivas por meio de um formulário.	Funcionalidade
RN4	Logout	Ao clicar em “Logout”, o sistema remove os dados do usuário do `localStorage` e redireciona para a página pública.	Segurança / Sessão
RN5	Exibição da mensagem mais curtida	A página exibe automaticamente a mensagem com mais "likes" do mês.	Visualização
RN6	Redirecionamento após logout	Se o usuário tentar voltar usando o botão de voltar do navegador após o logout, será redirecionado automaticamente para a tela de login.	Segurança / UX
RN7	Registro e login de usuários	O sistema possui uma tela de login e um link para o cadastro de novos usuários.	Acesso / Cadastro
RN8	Popup de feedback de autenticações	Ao realizar login, logout ou enviar mensagens, o sistema exibe um popup informando o sucesso ou erro da operação.	UX / Interface
RN9	Mensagem aleatória para visitantes	Visitantes não logados podem visualizar mensagens motivacionais e a mais curtida do mês, mas não interagir com o conteúdo.	Público / Marketing

Número	Nome	Descrição	Setor
RN10	Restrição de envio sem login	Usuários não logados não podem enviar mensagens, pois o campo `user_id` é requerido para o envio.	Segurança / Backend
RN11	Moderação por administrador	Existe um administrador responsável por moderar mensagens, contas de usuário e manter o bom funcionamento do site.	Administração
RN12	Regras de conteúdo positivo	Apenas mensagens com conteúdo positivo são permitidas, promovendo um ambiente de interação saudável.	Conteúdo / Ética

7. ANÁLISE DE REQUISITOS

Neste capítulo será descrito, através de uma tabela, os requisitos funcionais e não-funcionais.

7.1. REQUISITOS FUNCIONAIS

Número	Requisitos Funcionais	RN
RF1	Login de Usuário: O sistema deve fornecer uma interface de login com campos para e-mail e senha. Após o preenchimento, o sistema valida as credenciais no banco de dados	RN1
RF2	Cadastro de Usuário: O sistema deve permitir que novos usuários se cadastrem através de um formulário com campos obrigatórios: nome, e-mail e senha. A senha é armazenada no banco de dados usando hash (ex: bcrypt). Após o cadastro, o usuário recebe confirmação e é redirecionado para o login.	RN7
RF3	Validação de Sessão: O sistema deve informar ao usuário se ele obteve sucesso ou falha ao entrar	RN8
RF4	Logout: O sistema deve permitir que usuários logados encerrem a sessão através de um botão "Logout".	RN4
RF5	Envio de Mensagem Motivacional: O sistema deve disponibilizar um formulário para usuários logados enviarem mensagens motivacionais.	RN3
RF6	Exibir Mensagem Aleatória: O sistema deve exibir uma mensagem motivacional aleatória do banco de dados para visitantes não logados.	RN9

7.2. REQUISITOS NÃO-FUNCIONAIS

Número	Requisitos Não-Funcionais	RF
RNF1	Escalabilidade: O sistema deve ser capaz de lidar com o aumento do tráfego e da carga de dados.	RF2
RNF2	Usabilidade: A interface do usuário deve ser intuitiva e fácil de usar.	RF5
RNF3	Disponibilidade: O Sistema deve estar ativo 24/7.	
RNF4	Maturidade: O sistema deve ter poucas falhas ao longo do tempo.	
RNF5	Compatibilidade: Ser compatível com navegadores e SO.	

RNF6	Desempenho: O site deve ter um curto tempo de resposta.	
RNF7	Segurança: O sistema deve utilizar criptografia para proteger os dados pessoais.	RF1
RNF8	Conformidade legal: Garantir direitos e políticas de privacidade de acordo com a LGPD.	RF2

7.3. DIAGRAMA DE CASOS DE USO DA SOLUÇÃO

Nesta seção serão definidos os modelos de casos de uso. Primeiramente será mostrada uma visão geral dos casos de uso que definem as funcionalidades do sistema, com seus respectivos atores.

7.3.1. Visão Geral dos Casos de Uso e Atores

A Figura 1 a seguir será apresentado o Diagrama de Casos de Uso de *Software* com a visão de cada ator do sistema: usuário e sistema, abrangendo assim todas as funcionalidades previstas para a implementação.

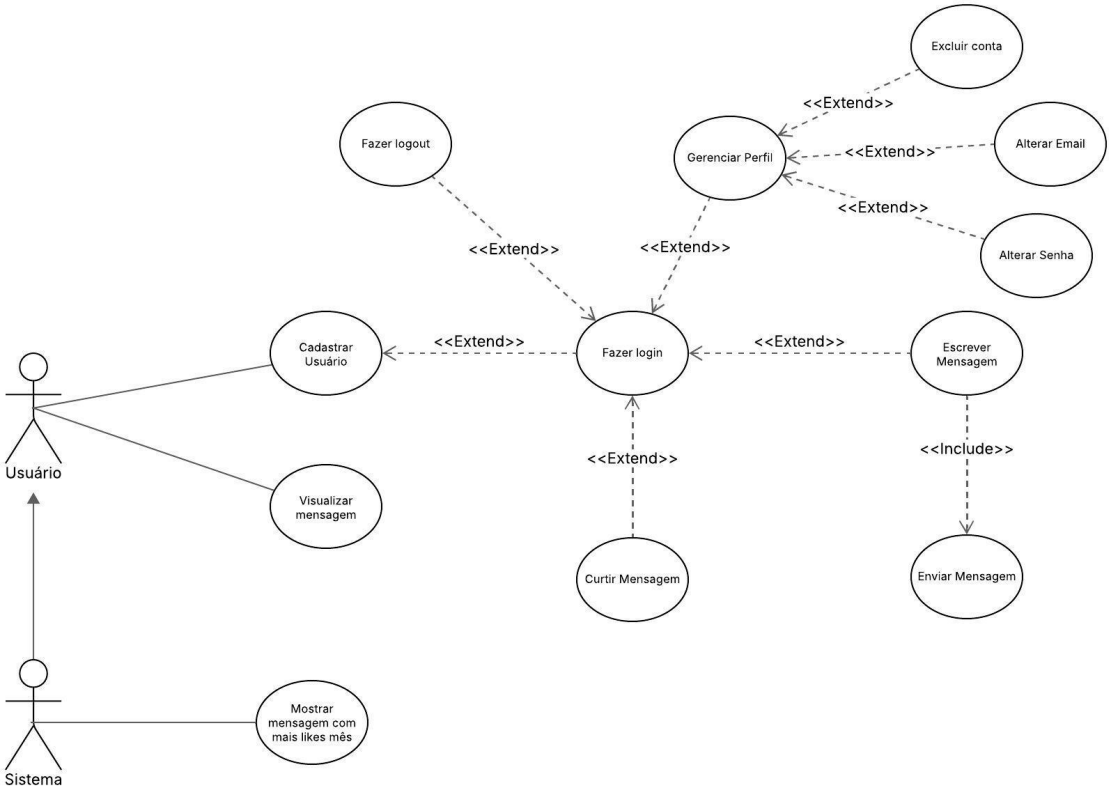

Figura 1 - Diagrama de Casos de Uso de *Software*.

Fonte: Elaboração própria, 2024.

7.4. DIAGRAMA DE CLASSE

Nesta seção será mostrado o diagrama de classes do sistema. Primeiramente será mostrada uma visão geral do diagrama de classes do sistema para um sistema de gerenciamento de locação de automóveis.

-

7.4.1. Visão Geral do diagrama de Classe

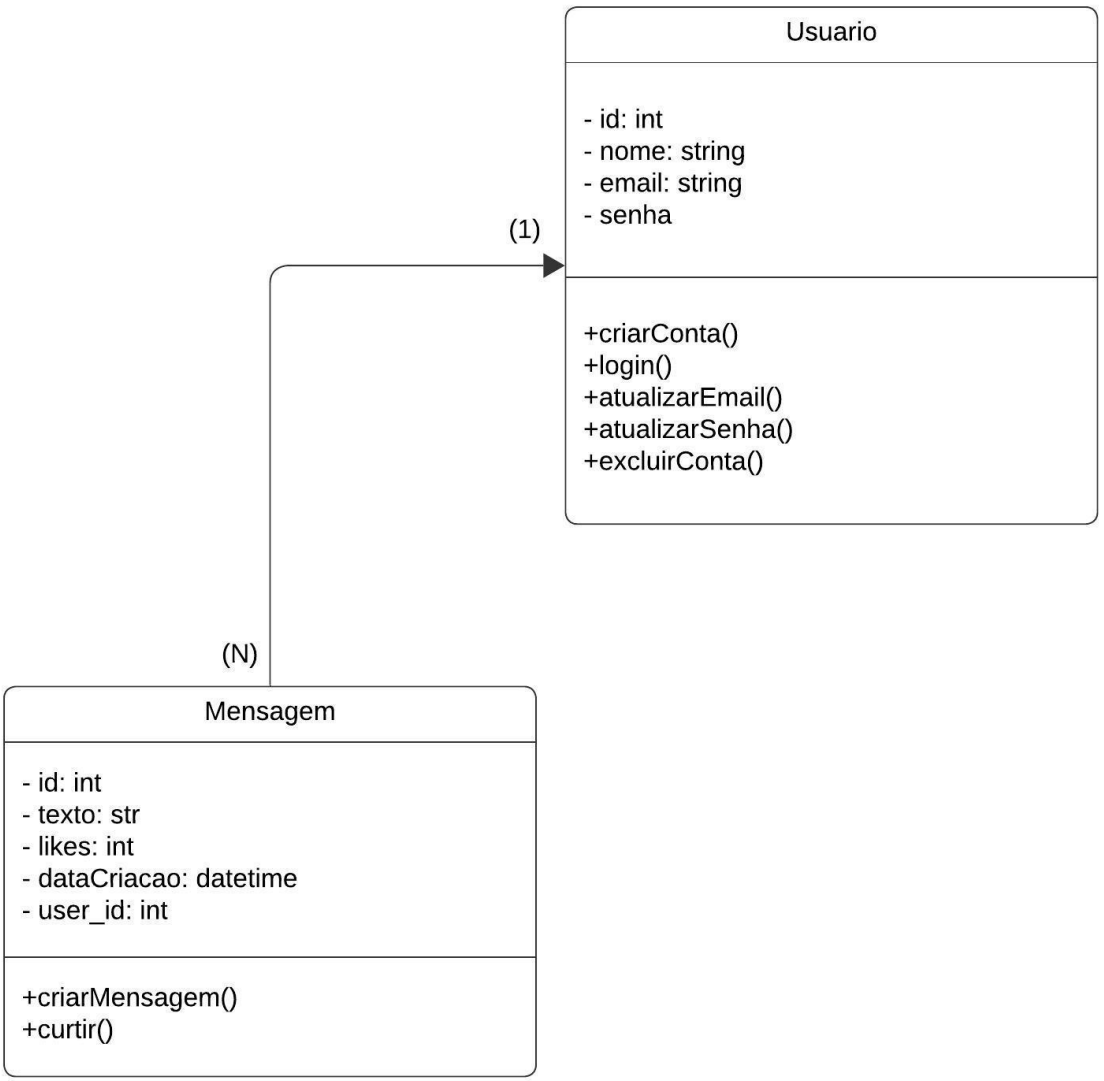

Figura 2 - Diagrama de Classe

O diagrama de classes desenvolvido representa as principais entidades envolvidas no funcionamento do sistema Positividade+, uma plataforma que permite que usuários compartilhem mensagens positivas e interajam com elas.

O sistema é composto por duas principais classes:

Classe Usuário: Representa os usuários cadastrados na plataforma. Cada usuário possui um identificador único (id), além de atributos como nome, email e senha. Esta classe é responsável por funcionalidades relacionadas à gestão da conta, tais como criarConta(), login(), atualizarEmail(), atualizarSenha() e excluirConta(). Os usuários são essenciais, pois todas as mensagens da plataforma estão vinculadas a um usuário que as criou.

Classe Mensagem: Representa as mensagens postadas na plataforma. Cada mensagem contém um id, um texto que é o conteúdo da mensagem, um contador de likes, a dataCriacao

e o user_id que referencia o autor da mensagem. Esta classe possui os métodos criarMensagem() para a criação de novas mensagens e curtir() para que outros usuários possam curtir a mensagem.

Relacionamento: Existe uma relação de um para muitos (1:N) entre Usuario e Mensagem, significando que um usuário pode criar várias mensagens, mas cada mensagem pertence exatamente a um usuário.

Ambiente e contexto: O sistema Positividade+ é uma aplicação web simples, com foco na interação de usuários por meio de mensagens positivas. Os usuários precisam estar cadastrados e autenticados para criar, visualizar, curtir ou gerenciar mensagens. O sistema mantém o controle das mensagens postadas e permite ações como curtir mensagens e gerenciar informações da conta.

7.4.2. Protótipos de Tela do Sistema

Protótipo das telas em média fidelidade

Figura 3 - Tela de Protótipo

7.4.2.1.1. Histórias de Usuário

User story: RF1 – Login de Usuário

Como usuário, eu quero logar no sistema, para que eu possa acessar funcionalidades exclusivas.

Cenário: Login bem-sucedido

Dado que o usuário está na página de login

Quando ele insere um e-mail válido e senha cadastrada

Então o sistema redireciona para a página inicial

Cenário: Login com credenciais inválidas

Dado que o usuário está na página de login

Quando ele insere um e-mail ou senha incorretos

Então o sistema exibe a mensagem “Login falhou”

User story: RF2 – Cadastrar Usuário

Como usuário, eu quero me cadastrar no sistema, para que eu possa criar uma conta no site.

Cenário: Cadastro bem-sucedido

Dado que o usuário está na página de cadastro

Quando ele preenche todos os campos obrigatórios (nome, e-mail, senha) corretamente

Então o sistema cria a conta e exibe "Cadastro concluído com sucesso"

Cenário: Cadastro com e-mail já existente

Dado que o usuário está na página de cadastro

Quando ele insere um e-mail já registrado

Então o sistema exibe a mensagem "E-mail já cadastrado"

User story: RF3 – Validar sessão

Como administrador, eu quero que o site avisa valide a sessão, para que confirme o login do usuário.

Cenário: Validação de token ativo

Dado que o usuário está fazendo login

Quando ele finalizar o login

Então o sistema envia uma mensagem informando sucesso ou falha

User story: RF4 – Logout

Como usuário, eu quero fazer logout, para desconectar da minha conta.

Cenário: Logout manual

Dado que o usuário está logado

Quando ele clica no botão "Sair"

Então o sistema encerra a sessão e redireciona para a página inicial

User story: RF5 – Mensagem motivacional

Como usuário, eu quero enviar mensagens, para que eu possa melhorar o dia de alguém.

Cenário: Envio de mensagem via formulário

Dado que o usuário está logado

Quando ele preenche o formulário com uma mensagem válida e clica em "Enviar"

Então o sistema salva a mensagem no banco de dados e exibe "Mensagem enviada com sucesso"

Cenário: Tentativa de envio com campo vazio

Dado que o usuário está logado

Quando ele deixa o campo da mensagem em branco

Então o sistema exibe o erro "Preencha o campo de mensagem"

User story: RF6 – Mensagem aleatória

Como usuário, eu quero ver mensagens, para que eu possa melhorar meu dia.

Cenário: Exibição de mensagem na página mensagens

Dado que o usuário está ou não logado

Quando ele acessa a página inicial e clica em ver mensagem

Então o sistema exibe uma mensagem motivacional aleatória

8. PLANEJAMENTO DE TESTES

8.1. TESTES DE SISTEMA/INTERFACE

- **Níveis de Teste**

Nível de Teste	**Descrição**	**Ferramenta Utilizada**	**Casos de Teste Correspondentes**
Teste de Sistema/Interface	Validação da aplicação como um todo, fluxo completo, UI e integração entre componentes	Cypress (Teste E2E)	- curtir_mensagem.cy.js (fluxo mensagens e likes) - login.cy.js (validação de login) - mensagem.cy.js (envio de mensagem) - registro.cy.js (registro de usuário) - usuario.cy.js (configurações do usuário)

Todos os testes são testes de interface (E2E), onde se exercita a aplicação completa, simulando o usuário interagindo com UI e verificando o comportamento final da aplicação (páginas, redirecionamentos, mensagens).

Técnica de Teste Aplicada e Análise de Cobertura

Para cada caso de teste, vamos identificar a técnica de teste, o nível de cobertura esperado e o que está efetivamente coberto:

curtir_mensagem.cy.js (Fluxo de mensagens)

- **Técnica:** Teste funcional baseado em fluxo (workflow testing)
- **Cobertura:**
    - **Cobertura funcional:** valida o fluxo do usuário ao logar, acessar mensagens, curtir e avançar.
    - **Cobertura de interface:** testes de botões, formulários, alertas.
    - **Cobertura de integração:** valida a navegação entre páginas e a atualização dinâmica do conteúdo.
- **Limitações:** Não testa os casos de erro do backend (ex: falha no like), pois depende do sistema estar disponível e responsivo.

---

login.cy.js (Login)

- **Técnica:** Teste funcional baseado em casos positivos e negativos (validação de entrada)
- **Cobertura:**
    - **Cobertura funcional:** testes de login bem-sucedido e login inválido (campos vazios).
    - **Cobertura de interface:** valida mensagens de erro e redirecionamentos.
- **Limitações:** Não testa backend diretamente (ex: autenticação falha por senha incorreta), apenas interface.

---

mensagem.cy.js (Envio de mensagem)

- **Técnica:** Teste funcional de fluxo (input-output)
- **Cobertura:**

    - Valida o envio de mensagem com texto positivo e a confirmação da ação.
    - Cobertura de interface (formulário e popup).
- **Limitações:** Não cobre mensagem negativa ou falha no envio.

---

registro.cy.js (Registro de usuário)

- **Técnica:** Teste funcional baseado em casos positivos
- **Cobertura:**
    - Valida o cadastro e feedback na interface.
- **Limitações:** Não cobre casos negativos (ex: email já cadastrado, senha inválida).

---

usuario.cy.js (Configurações do usuário)

- **Técnica:** Teste funcional de fluxo completo com stub para alertas e confirmações
- **Cobertura:**
    - Cobertura funcional do fluxo de atualização e exclusão de conta.
    - Cobertura de interface, mensagens de alerta e redirecionamento.
- **Limitações:** Dependência da implementação dos métodos que efetivamente realizam a atualização/exclusão.

Caso de Teste	**Técnica de Teste**	**Nível de Cobertura**	**Observações sobre cobertura**
curtir_mensagem.cy.js	Teste funcional / fluxo	Cobertura funcional + interface + integração	Cobre fluxo positivo do uso da funcionalidade.
login.cy.js	Teste funcional (positivo e negativo)	Cobertura funcional e interface	Cobre login correto e validação mínima de

			campos.
mensagem.cy.js	Teste funcional / fluxo	Cobertura funcional e interface	Testa envio de mensagem positiva.
registro.cy.js	Teste funcional (positivo)	Cobertura funcional e interface	Testa registro bem-sucedido.
usuario.cy.js	Teste funcional / fluxo completo	Cobertura funcional + interface + integração	Testa atualização e exclusão com stub de alertas.

Priorização por risco

Teste de Interface	**Risco de Falha**	**Impacto**	**Prioridade**
Login	Alta	Alta	Alta
Cadastro	Alta	Alta	Alta
Envio de Mensagem	Média	Alta	Alta
Curtir Mensagem	Média	Média	Média
Configurar/Excluir Usuário	Média	Média	Média

Priorização por Requisito

Teste	**Requisitos Relacionados**	**Prioridade do Requisito**	**Prioridade do Teste**
login.cy.js	RF1	Alta	Alta
registro.cy.js	RF2	Alta	Alta
mensagem.cy.js	RF5	Alta	Alta
curtir_mensagem.cy.js	RF6, RF5	Baixa/Média	Média
usuario.cy.js	RF3, RF4	Média	Média

Priorização por Cobertura

Script de Teste	Cobertura Alcançada (Funcional + UI + Integração)	Prioridade
login.cy.js	Alta (fluxo positivo e negativo)	Alta
registro.cy.js	Média (somente casos positivos)	Média
mensagem.cy.js	Média (formulário positivo, não cobre erros)	Média
curtir_mensagem.cy.js	Alta (workflow completo de interação)	Alta
usuario.cy.js	Alta (fluxo completo com alertas e navegação)	Alta

Resumo Final – Priorização dos Testes de Interface

Script de Teste	**Priorização Final**
login.cy.js	Alta
registro.cy.js	Alta
mensagem.cy.js	Alta
curtir_mensagem.cy.js	Alta
usuario.cy.js	Alta

8.2. TESTES UNITÁRIOS COM ROTAS DE API

- **Níveis de Teste**

Abaixo estão os níveis de testes unitários aplicados à API, juntamente com as ferramentas utilizadas e os casos de teste correspondentes:

Nível de Teste	Ferramenta Utilizada	Casos de Teste
Teste Unitário	Jest, Supertest	/login, /register, /update-email, /update-senha, /delete-user, /mensagem, /top-message, /random-message, /increment-likes
Teste de Integração	Jest, Supertest, bcrypt	/login (bcrypt), /register (validações e inserções no banco)
Teste Funcional	Jest, Supertest	Verificação de respostas corretas e tratamento de erros nas rotas

- **Taxa de Cobertura**

Cada rota foi testada com técnicas de teste apropriadas. A seguir, os detalhes de cobertura para cada caso:

Rota	Técnica de Teste	Cobertura
/login	Caixa Preta + Análise de Valor Limite	Cobertura ~100% dos cenários principais
/register	Caixa Preta + Partição de Equivalência	Cobertura ~100%
/update-email	Caixa Preta	Cobertura baixa (só sucesso)
/update-senha	Caixa Preta	Cobertura baixa (só sucesso)
/delete-user	Caixa Preta	Cobertura média (só sucesso)
/mensagem	Caixa Preta	Cobertura baixa (só sucesso)
/top-message	Caixa Preta + Partição de Equivalência	Cobertura alta
/random-message	Caixa Preta + Partição de Equivalência	Cobertura alta
/increment-likes	Caixa Preta	Cobertura baixa (só sucesso)

Priorização por Risco

Função / Rota	Risco de Falha	Impacto	Prioridade
/login	Alta	Alta	Alta
/registro	Alta	Alta	Alta
/mensagem	Média	Alta	Alta
/deletar-usuário	Média	Alta	Alta

/atualizar-email	Média	Média	Média
/atualizar-senha	Média	Média	Média
/likes	Média	Média	Média
/top-mensagem	Baixa	Baixa	Baixa
/mensagem-aleatória	Baixa	Média	Média

Priorização por Requisito

Função / Rota	**Requisitos Associados**	**Prioridade do Requisito**	**Prioridade do Teste**
/login	RF1	Alta	Alta
/registro	RF2	Alta	Alta
/mensagem	RF5	Alta	Alta
/likes	RF5, RF6	Média	Média
/deletar-usuário	RF4	Média	Alta
/atualizar-email	RF3	Média	Média
/atualizar-senha	RF3	Média	Média
/top-mensagem	RF6	Baixa	Baixa
/mensagem-aleatória	RF6	Baixa	Média

Priorização por Cobertura

Rota / Função	**Técnica Aplicada**	**Cobertura Obtida**	**Prioridade**
/login	Caixa Preta + Limite	Alta – cobre sucesso e erro	Alta
/registro	Caixa Preta + Equivalência	Alta – cobre duplicidade, campos	Alta
/mensagem	Caixa Preta	Baixa – só caso	Média

		positivo	
/likes	Caixa Preta	Baixa – só sucesso	Baixa
/deletar-usuário	Caminho Base	Média – só sucesso	Média
/atualizar-email	Caixa Preta	Baixa – só sucesso	Baixa
/atualizar-senha	Caixa Preta	Baixa – só sucesso	Baixa
/top-mensagem	Partição de Equivalência	Alta – retorna corretamente	Média
/mensagem-aleatória	Partição de Equivalência	Alta – resposta esperada	Média

8.3. TESTES DE API

- **Níveis de Teste**

Caso de Teste	**Ferramenta**	**Nível de Teste**	**Descrição**
Login com credenciais corretas e incorretas	Cypress (cy.request)	Teste de Integração	Verifica a integração do frontend (via requisição HTTP) com o backend de autenticação.
Registro de usuário	Cypress (cy.request)	Teste de Integração	Válida a criação de novos usuários via API e o comportamento em casos inválidos.
Enviar mensagem positiva	Cypress (cy.request)	Teste de Integração	Testa a API de envio de mensagens, exigindo user_id e texto.
Curtir mensagem (incrementar likes)	Cypress (cy.request)	Teste de Integração	Verifica se o backend incrementa corretamente os likes de uma mensagem.
Obter mensagens	Cypress	Teste de Integração	Testa endpoints que

públicas (top e aleatória)	(cy.request)		retornam mensagens públicas, verificando dados como texto e likes.
Atualizar email/senha	Cypress (cy.request)	Teste de Integração	Garante que o backend atualiza dados do usuário corretamente.
Deletar conta	Cypress (cy.request)	Teste de Integração	Verifica se a conta do usuário é removida com sucesso.

- **Análise de Cobertura**

Caso de Teste	**Técnica de Teste Aplicada**	**Cobertura**
Login com sucesso, senha errada, dados ausentes	Partição de Equivalência + Análise de Valor Limite	Alta – cobre casos típicos e inválidos esperados.
Registro (válido, incompleto, email duplicado)	Partição de Equivalência	Alta – cobre todos os principais fluxos possíveis do endpoint de registro.
Enviar mensagem com e sem dados obrigatórios	Partição de Equivalência	Alta – cobre cenário válido e inválido de envio.
Curtir mensagem com e sem ID	Partição de Equivalência + Caminho Base	Alta – cobre caminhos esperados e alternativos (erro 400).
Mensagens públicas (top/aleatória)	Caminho Base	Média – cobre caminho principal (resposta esperada), mas não erros.
Atualizar email e senha	Partição de Equivalência	Média – cobre casos válidos, mas não casos inválidos (ex: email vazio,

		senha fraca).
Excluir usuário	Caminho Base	Média – cobre apenas fluxo feliz (exclusão correta), não erro se ID for inválido.

- **Por risco**

Rota	**Risco de Falha**	**Impacto**	**Prioridade**
/login	Alta	Alta	Alta
/registro	Alta	Alta	Alta
/mensagem	Média	Alta	Média
/likes	Média	Média	Média
/deletar-usuário	Média	Alta	Alta
/atualizar-email	Média	Média	Média
/atualizar-senha	Média	Média	Média
/top-mensagem	Baixa	Baixa	Baixa
/mensagem-aleatória	Baixa	Média	Média

- **Por Requisitos**

Rota da API	**Requisitos Associados**	**Prioridade do Requisito**	**Prioridade do Teste**
/login	RF1	Alta	Alta
/registro	RF2	Alta	Alta
/mensagem	RF5	Alta	Alta
/likes	RF5, RF6	Média	Média
/deletar-usuário	RF4	Média	Alta
/top-mensagem	RF6	Baixa	Baixa
/mensagem-aleatória	RF6	Baixa	Média

/atualizar-email	RF3	Média	Média
/atualizar-senha	RF3	Média	Média

- **Por cobertura**

Rota	**Técnica de Teste Aplicada**	**Cobertura Obtida**	**Prioridade**
/login	Partição de Equivalência + Valor Limite	Alta – cobre sucesso e erro	Alta
/registro	Partição de Equivalência	Alta – cobre duplicidade	Alta
/mensagem	Partição de Equivalência	Média – cobre só sucesso	Média
/likes	Partição de Equivalência	Baixa – cobre só sucesso	Média
/deletar-usuário	Caminho Base	Média – não cobre erro	Média
/top-mensagem	Partição de Equivalência	Alta – cobre dados esperados	Média
/mensagem-aleatória	Partição de Equivalência	Alta – cobre dados esperados	Média
/atualizar-email	Caixa Preta	Baixa – cobre só caso válido	Baixa
/atualizar-senha	Caixa Preta	Baixa – cobre só caso válido	Baixa

8.4. TESTES DE STRESS

- **Níveis de Teste**

O teste de stress foi configurado utilizando uma abordagem de carga constante (arrivalRate) com o Artillery. Abaixo está o nível de teste, ferramenta e as páginas testadas:

Nível de Teste	**Ferramenta Utilizada**	**EndPoints/Páginas**

		Testadas
Teste de Desempenho (Stress)	Artillery	/index.html, /login.html, /registro.html

- **Taxa de Cobertura**

A técnica utilizada foi o teste de carga/stress com objetivo de verificar o comportamento do sistema sob alta demanda. A tabela a seguir resume os aspectos de cobertura:

Ponto Testado	**Técnica de Teste**	**Cobertura**
/index.html	Stress Testing (GET)	Alta - 20 requisições/s por 30s
/login.html	Stress Testing (GET)	Alta - 20 requisições/s por 30s
/registro.html	Stress Testing (GET)	Alta - 20 requisições/s por 30s

- **Por risco**

Página / Endpoint	**Risco de Falha**	**Impacto**	**Prioridade**
/login.html	Alta	Alta	Alta
/registro.html	Alta	Alta	Alta
/index.html (página inicial)	Média	Média	Média

- **Por Requisitos**

Página / Endpoint	**Requisitos Relacionados**	**Prioridade do Requisito**	**Prioridade do Teste**
/login.html	RF1 (Login de Usuário)	Alta	Alta
/registro.html	RF2 (Cadastro de Usuário)	Alta	Alta
/index.html	RF6 (Mensagem aleatória)	Baixa	Média

9. ANÁLISE DE RISCO DO PRODUTO

- **Escopo dos Testes a Serem Realizados**

O escopo de testes abrange as funcionalidades essenciais do sistema web Positividade+, com foco em:

Autenticação de usuários (login, cadastro, logout), Interações principais: envio e curtidas de mensagens, Moderação e exclusão de contas, Visualização de mensagens públicas (aleatórias e mais curtidas), Resistência da aplicação sob carga (stress), Confiabilidade das APIs e funcionalidades de backend.

Segurança e feedback da interface (UI/UX)

- **Níveis de Teste e Tipos de Testes Propostos**

Nível de Teste	**Tipo de Teste**	**Finalidade**
Unidade	Teste Unitário	Garantir funcionamento correto de cada função isolada
Integração	Testes de API / Backend	Verificar comunicação entre módulos e tratamento de dados
Sistema/Interface	Testes E2E (Cypress)	Simular o uso completo por parte do usuário
Desempenho	Teste de Stress (Artillery)	Avaliar robustez e tempo de resposta sob carga
Funcionalidade Geral	Teste Funcional	Validar o comportamento conforme os requisitos

- **Técnicas de Teste e Cobertura Esperada**

Técnica de Teste	**Aplicação**	**Cobertura Esperada**
Caixa Preta	API e rotas de backend	~100% dos fluxos principais
Partição de Equivalência	Login, Cadastro, Curtir, Mensagens	Casos válidos e inválidos
Valor	LimiteCampos de entrada (ex: e-mail, senha, texto)	Campos obrigatórios
Teste de Fluxo (Workflow Testing)	Interações completas do usuário	(UI)Alto

Stress Testing	login,registro, indexAlta carga	estabilidade

- **Estimativa de esforço por tarefa**

Tarefa de Teste	**Nível**	**Duração Estimada**	**Observações**
Testes de Login/Cadastro	Interface + API	6 horas	Inclui casos positivos e negativos
Envio e curtida de mensagens	Interface + API	8 horas	Cobre interações, feedback e erros
Testes de sessão e logout	Interface	3 horas	Simples, mas importante
Testes de moderação/exclusão	API + Unidade	5 horas	Cobertura funcional parcial
Testes de mensagens públicas	Interface + API	4 horas	Inclui mensagem aleatória e popular
Testes de stress	Desempenho	4 horas	Inclui configuração e análise
Testes unitários	Backend	10 horas	Foco em rotas críticas (/login, /registro, /mensagem)

10. RESULTADO DOS TESTES

ferramentas utilizadas e suas versões:

- jest versão: 29.7.0
- Cypress package version: 14.4.1
- Cypress binary version: 14.4.1
- Electron version: 33.2.1
- Bundled Node version: 20.18.1
- Artillery: 2.0.23
- Node.js: v 22.16.0
- OS: win32
- supertest@7.1.1

10.1. TESTE DE SISTEMA/INTERFACE

registro.cy.js resultado no teste no cypress:

deve registrar um novo usuário com sucessopassed

test body

visithttp://localhost:3000/registro

2

getinput[name="name"]

3

typeNovo Usuario

4

getinput[name="email"]

5

typeteste@teste.com

6

getinput[name="password"]

7

type123456

8

getbutton[type="submit"]

9

click

(fetch)POST 200 /register

10

get#popupMessage

11

assertexpected <p#popupMessage> to contain Registro bem-sucedido

(page load)--page loaded--

(new url)http://localhost:3000/login.html

login.cy.js resultado no teste no cypress:

Deve mostrar erro se campos estiverem vaziospassed

test body

1

visithttp://localhost:3000/login

2

getbutton[type="submit"]

3

click

4

wait500

5

get#popupMessage

6

invoke.text()

7

assertexpected Email e senha são obrigatórios to match /obrigatórios/i

Deve logar com sucessopassed

test body

1

visithttp://localhost:3000/login

2

get#email

3

typeteste@teste.com

4

get#password

5

type123456

6

getbutton[type="submit"]

7

click

(fetch)POST 200 /login

8

url

9

assertexpected http://localhost:3000/index-logged.html to include index-logged

(page load)--page loaded--

(new url)http://localhost:3000/index-logged.html

(fetch)GET 200 /top-message

mensagem.cy.js resultado no teste no cypress:

Deve logar e enviar uma nova mensagem positivapassed

test body

1

visithttp://localhost:3000/login

2

getinput[name="email"]

3

typeteste@teste.com

4

getinput[name="password"]

5

type123456

6

getbutton[type="submit"]

7

click

(fetch)POST 200 /login

8

url

9

assertexpected http://localhost:3000/index-logged.html to include index-logged

(page load)--page loaded--

(new url)http://localhost:3000/index-logged.html

(fetch)GET 200 /top-message

10

getform#mensagemForm textarea[name="texto"]

... (176 linhas)

10.2. TESTE UNITÁRIO

npx jest

>>

console.log

```
  Servidor rodando na porta 3000


    at Server.log (server.js:227:13)


PASS  ./server.test.js
 Testes das rotas
   POST /login
     √ Deve retornar erro se email ou senha não fornecidos (63 ms)
     √ Deve retornar sucesso se usuário e senha corretos (116 ms)
     √ Deve retornar erro se senha incorreta (117 ms)
     √ Deve retornar erro se usuário não encontrado (4 ms)
   POST /register
     √ Deve retornar erro se campos faltando (3 ms)
     √ Deve retornar erro se email já existe (4 ms)
     √ Deve registrar usuário com sucesso (62 ms)
   PUT /update-email
     √ Deve atualizar email com sucesso (3 ms)
   PUT /update-senha
     √ Deve atualizar senha com sucesso (63 ms)
   POST /delete-user
     √ Deve excluir conta com sucesso (4 ms)
   POST /mensagem
     √ Deve enviar mensagem com sucesso (3 ms)
   GET /top-message
     √ Deve retornar mensagem top do mês (5 ms)
     √ Deve retornar nenhuma mensagem encontrada (4 ms)
   GET /random-message
```

√ Deve retornar uma mensagem aleatória (3 ms)

√ Deve retornar nenhuma mensagem encontrada (3 ms)

POST /increment-likes

√ Deve incrementar like com sucesso (4 ms)

Test Suites: 1 passed, 1 total

Tests: 16 passed, 16 total

Snapshots: 0 total

Time: 1.367 s

Ran all test suites.

10.3. TESTE DE API

register.cy.js:

API Registro

Deve registrar usuário com dados válidospassed

test body

1

requestPOST 200 http://localhost:3000/register

2

assertexpected 200 to equal 200

3

assertexpected { Object (success, message) } to have property success

4

assertexpected { Object (success, message) } to have property success** of **true

5

assertexpected { Object (success, message) } to have property message

6

assertexpected Registro bem-sucedido! to match /registro bem-sucedido/i

Deve falhar ao registrar sem nome, email ou senhapassed

test body

1

requestPOST 400 http://localhost:3000/register

2

assertexpected 400 to equal 400

3

assertexpected { Object (success, message) } to have property success

4

assertexpected { Object (success, message) } to have property success** of **false

5

assertexpected { Object (success, message) } to have property message

6

assertexpected Nome, email e senha são obrigatórios to match /obrigatórios/i

Deve falhar ao registrar com email já existentepassed

test body

1

requestPOST 400 http://localhost:3000/register

2

assertexpected 400 to equal 400

3

assertexpected { Object (success, message) } to have property success

4

assertexpected { Object (success, message) } to have property success** of **false

5

assertexpected { Object (success, message) } to have property message

6

assertexpected Email já registrado to match /registrado/i

login.cy.js:

API Login

Deve logar com sucessopassed

test body

1

requestPOST 200 http://localhost:3000/login

2

assertexpected 200 to equal 200

3

assertexpected { Object (success, message, ...) } to have property success

4

assertexpected { Object (success, message, ...) } to have property success** of **true

5

assertexpected { Object (success, message, ...) } to have property user

6

assertexpected { id: 47, nome: Usuário Teste } to have property id

7

assertexpected { id: 47, nome: Usuário Teste } to have property nome

Deve falhar ao logar com senha erradapassed

test body

1

requestPOST 200 http://localhost:3000/login

2

assertexpected 200 to equal 200

3

assertexpected { Object (success, message) } to have property success

4

assertexpected { Object (success, message) } to have property success** of **false

5

assertexpected Senha incorreta! to match /senha/i

Deve falhar se não enviar email e senhapassed

test body

1

requestPOST 400 http://localhost:3000/login

2

assertexpected 400 to equal 400

3

assertexpected { Object (success, message) } to have property success

4

assertexpected { Object (success, message) } to have property success** of **false

5

assertexpected Email e senha são obrigatórios to match /obrigatórios/i

mensagem.cy.js:

API Mensagem

Deve enviar mensagem com usuário logadopassed

before all

1

requestPOST 200 http://localhost:3000/login

2

assertexpected 200 to equal 200

3

assertexpected { Object (success, message, ...) } to have property success

4

assertexpected { Object (success, message, ...) } to have property success** of **true

... (129 linhas)

10.4. TESTE DE STRESS

Metrics for period to: 15:48:40(-0300) (width: 4.313s)

---

http.codes.200: 258

http.downloaded_bytes: 730828

http.request_rate: 71/sec

http.requests: 258

http.response_time:

min: 0

max: 2

mean: 0.6

median: 1

p95: 1

p99: 1

http.response_time.2xx:

min: 0

max: 2

mean: 0.6

median: 1

p95: 1

p99: 1

http.responses: 258

vusers.completed: 86

vusers.created: 86

vusers.created_by_name.Teste simples GET nas páginas principais: 86

vusers.failed: 0

vusers.session_length:

min: 4.1

max: 27.5

mean: 8.2

median: 5.5

p95: 24.8

p99: 26.3

Metrics for period to: 15:48:50(-0300) (width: 9.943s)

---

http.codes.200: 600

http.downloaded_bytes: 1699600

http.request_rate: 60/sec

http.requests: 600

http.response_time:

min: 0

max: 2

mean: 0.6

median: 1

p95: 1

p99: 1

http.response_time.2xx:

min: 0

max: 2

mean: 0.6

median: 1

p95: 1

p99: 1

http.responses: 600

vusers.completed: 200

vusers.created: 200

vusers.created_by_name.Teste simples GET nas páginas principais: 200

vusers.failed: 0

vusers.session_length:

min: 3.7

max: 8.2

mean: 4.7

median: 4.4

p95: 6

p99: 7.8

Phase completed: unnamed (index: 0, duration: 30s) 15:49:05(-0300)

Metrics for period to: 15:49:00(-0300) (width: 9.942s)

---

http.codes.200: 600

http.downloaded_bytes: 1699600

http.request_rate: 60/sec

http.requests: 600

http.response_time:

  min: 0

  max: 1

  mean: 0.6

  median: 1

  p95: 1

  p99: 1

http.response_time.2xx:

  min: 0

  max: 1

  mean: 0.6

  median: 1

  p95: 1

  p99: 1

http.responses: 600

vusers.completed: 200

vusers.created: 200

... (84 linhas)

11. GITHUB

https://github.com/Zalghoul/Positividade_mais

12. CONCLUSÃO E LIÇÕES APRENDIDAS

O projeto Positividade+ representou uma iniciativa relevante para promover interações digitais saudáveis e motivacionais, alinhadas às demandas contemporâneas por ambientes online mais acolhedores. A plataforma desenvolvida atendeu aos objetivos principais: criar um espaço dedicado ao compartilhamento de mensagens positivas, com funcionalidades como envio anônimo/identificado de mensagens, sistema de curtidas e exibição de conteúdo inspirador. A implementação seguiu boas práticas de engenharia de software, utilizando tecnologias modernas (Node.js, MySQL, Cypress) e aderindo à ISO 25010 para garantia de qualidade.

Principais Conclusões:

Validação da Proposta:

O diagnóstico inicial sobre a lacuna de plataformas focadas em positividade foi confirmado. A simplicidade da solução (formulários intuitivos, feedback imediato via pop-ups) mostrou-se eficaz para engajar usuários.

As regras de negócio, como autenticação obrigatória para interações e moderação por administrador, foram essenciais para manter a integridade do ambiente.

Eficácia dos Testes:

Testes E2E (Cypress): Validaram fluxos críticos (login, envio de mensagens, curtidas) e garantiram uma experiência de usuário coesa.

Testes de API (Jest/Supertest): As rotas de autenticação (/login, /registro) e gestão de mensagens atingiram alta cobertura, mas casos de erro (ex: duplicidade de e-mail) precisam de reforço.

Testes de Stress (Artillery): A aplicação suportou carga de 20 solicitações/segundo, porém escalabilidade requer otimizações futuras (ex: cache de mensagens).

Conformidade com Requisitos:

Os requisitos funcionais essenciais (RF1 a RF6) foram atendidos, com destaque para o sistema de curtidas e exibição de mensagens aleatórias.

Requisitos não funcionais como disponibilidade 24h e usabilidade foram priorizados, mas segurança (ex: criptografia robusta de senhas) demanda melhorias.

Lições Aprendidas:

Cobertura de Testes:

Gap identificado: Testes unitários de rotas como /update-senha e /adicionar-likes tiveram baixa cobertura de cenários negativos (ex: campos vazios, IDs inválidos).

Solução futura: Adotar testes baseados em comportamentos (BDD) para mapear casos críticos com maior rigor.

Priorização Eficiente:

A priorização por risco (ex: focar em login e registro) evitou falhas graves, mas subestimou funcionalidades como atualização de perfil.

Recomendação: Usar matriz Impacto x Esforço para equilibrar cobertura entre funcionalidades de alta criticidade e baixa complexidade.

Integração Front-Backend:

Desafio: Discrepâncias entre validações no frontend (JavaScript) e backend (Node.js) causaram erros pontuais (ex: mensagens com texto vazio).

Lições: Validar dados em ambas as camadas e adotar contratos de API (OpenAPI) para sincronia.

Gestão de Dados Sensíveis:

Armazenar user_id no localStorage mostrou vulnerabilidades potenciais (ex: ataques XSS).

Melhoria: Migrar para sessões HTTP-only com tokens JWT.

Resiliência Sob Carga:

Testes de stress expuseram lentidão em /registro.html durante picos.

Ação: Otimizar consultas ao banco (ex: índices em campos frequentes) e adotar balanceamento de carga.

Recomendações para Trabalhos Futuros:

Expansão de Testes:

Implementar testes de segurança (OWASP ZAP) para verificar vulnerabilidades (ex: SQL injection).

Adicionar testes de acessibilidade (WCAG) para garantir inclusividade.

Melhorias de Arquitetura:

Substituir o armazenamento local por gerenciamento de estado centralizado (ex: Redux).

Adotar microsserviços para funcionalidades críticas (autenticação, mensagens).

Monitoramento Contínuo:

Integrar ferramentas como New Relic ou Prometheus para acompanhar desempenho em produção.

Impacto do Projeto:

O Positividade+ demonstrou que soluções tecnológicas simples podem gerar impacto social relevante, promovendo bem-estar digital. A abordagem de teste estruturada desde unitários até testes de stress assegurou robustez, mas reforçou que qualidade é um processo iterativo. As lições aqui consolidadas servirão como base para evolução da plataforma e futuros projetos na área de software.

Brasília, junho de 2025

Diogo França, Hudson Cunha, Gustavo Linhares