Tech Forge

Versão 2.0

Autores:

Beatriz Andrade Siquara

Breno Pimentel de Almeida Miranda

Luan Gabriel Santos Paim Dias

Pablo Santana dos Santos

Robson Graça dos Santos

Yana Barreto Luiz Simina

Revisor: Prof. Felipe Oliveira

2024

ÍNDICE DETALHADO

1. Introdução ao Documento

1.1. Tema

Uma plataforma web abrangente e intuitiva, destinada a simplificar, organizar e evidenciar as ferramentas de gerenciamento de software. Possui uma interface amigável ao usuário e recursos de filtros e pesquisa que oferece informações sobre cada aplicativo.

1.2. Objetivo do Projeto

Essa plataforma visa capacitar os usuários a tomarem decisões informadas sobre quais softwares melhor atenderão às suas necessidades específicas.

1.3. Delimitação do Problema

A delimitação do problema abrange a identificação e categorização de softwares de gerenciamento, a implementação de uma base de dados detalhada e constantemente atualizada, e a criação de mecanismos que facilitem a comparação e análise das diferentes opções disponíveis. Assim, a plataforma visa solucionar a dispersão e complexidade de informações, capacitando os usuários a tomarem decisões informadas e seguras sobre as ferramentas que melhor atendem às suas necessidades específicas. A meta é não apenas oferecer uma lista de opções, mas também fornecer insights significativos que orientem a escolha do software ideal, promovendo uma experiência de usuário eficiente e satisfatória.

1.4. Justificativa da Escolha do Tema

A escolha deste tema foi motivada pela observação das dificuldades enfrentadas por profissionais e empresas ao buscar ferramentas de gerenciamento de software que atendam de maneira eficiente às suas necessidades específicas. A proliferação de aplicativos disponíveis no mercado, combinada com a falta de um

recurso centralizado que ofereça informações claras e comparativas, resulta em um processo de seleção demorado e, muitas vezes, ineficaz.

1.5. Método de Trabalho

O modelo ágil consiste em abordagens de trabalho que são adaptáveis, flexíveis e orientadas para a entrega rápida e interativa de valor. As vantagens dessa metodologia faz com que seu negócio ganhe mais agilidade e seja mais eficiente nas execuções dos seus processos. As entregas finais tendem a ser mais organizadas e claras.

O Scrum é uma metodologia que otimiza recursos, custos e tempo, ajuda na organização de processos, identificando e eliminando possíveis problemas antes que se tornem irreversíveis, dessa forma ficará mais claro identificar o que pode estar atrapalhando a empresa.

O motivo de escolha para o modelo ágil consiste em diversos fatores analisados que favorecem a desenvoltura do nosso projeto. A modelagem foca em pensarmos antes de criarmos quaisquer modelos, colocando em pauta o objetivo do projeto, o que ajuda a compreender melhor aspectos que não estejam claros. O conceito ágil busca proatividade e agilidade, sem comprometer a qualidade do produto. Há uma hierarquia de demandas onde temos como prioridade satisfazer o usuário final. Com o Scrum podemos construir e gerenciar projetos de software, onde cada parte do software é concluída em um curto espaço de tempo e entregue ao cliente.

O controle do nosso Product Backlog está sendo feito através da ferramenta Notion, onde temos quatro sprints com um mês de duração (cada sprint). Durante as Sprint teremos os seguintes encontros:

Sprint planning

Acontecerá uma vez por mês com 1h30m de duração, na semana que irá começar uma nova sprint faremos o seu planejamento definindo quantas funcionalidades do product backlog podem ser realizadas dentro daquela print.

Daily

A daily são reuniões diárias que faremos com a duração de 30 minutos cada, onde discutiremos sobre:

- O que foi feito ontem que ajudou no alcance das metas?
- O que será feito hoje para ajudar a completar a sprint?
- Quais são os impedimentos ou dificuldades para a tarefa?

Sprint review

A sprint review acontecerá 4 vezes por mês com 1h de duração (4 vezes por sprint), sendo uma vez por semana. Nela discutiremos a evolução do projeto, tiraremos dúvidas e veremos o que está sendo feito e o que está sendo esperado.

São nessas revisões que aparecem possíveis mudanças a serem consideradas no backlog para atualizá-lo e iniciar o novo Sprint.

Sprint Retrospective

Acontecerá uma vez por mês no final de toda sprint para fazer uma retrospectiva sobre as atividades que foram realizadas, os aprendizados e desafios enfrentados por cada membro, possuindo 1h de duração. Veremos o que funcionou e o que não funcionou para que nós possamos melhorar a próxima sprint.

Product Backlog: Sprints e suas atividades realizadas com descrições detalhadas.

Sprint 1

- Configuramos o ambiente de desenvolvimento, incluindo ferramentas de controle de versão, IDE, ferramentas de engenharia de software que serão utilizadas.
- Definimos a metodologia ágil para ser aplicada no processo do desenvolvimento do projeto.
- Definimos o modelo de processo ágil para ser utilizado.
- Definimos os requisitos funcionais e não funcionais da plataforma.

Sprint 2

- Fizemos a documentação do modelo de processo de software selecionado, com justificativas e descrição detalhada das fases. Plano detalhado para cada fase do modelo de processo, destacando as atividades a serem realizadas.
- Definimos os papéis dos componentes do grupo e também as suas responsabilidades.
- Implementamos as práticas e cerimônias da metodologia ágil. Definimos a quantidade de reuniões, como serão feitas e quando acontecerão.
- Configuramos a ferramenta de gestão de projetos e colaboração (Notion).

Sprint 3

- Faremos a prototipação completa do nosso projeto com os responsáveis pela área, com a elaboração do header, footer, body, etc.
- Faremos o desenvolvimento da nossa aplicação de acordo com o que será apresentado no figma, com os responsáveis por esse papel.

Sprint 4

- Desenvolveremos a lógica de negócio para processar as requisições dos usuários.
- Iremos implementar os algoritmos de busca e filtros para facilitar a experiência do usuário.
- Apresentaremos o projeto.

2. Descrição Geral do Sistema

A plataforma web proposta é um sistema abrangente e intuitivo, projetado para simplificar, organizar e evidenciar as diversas ferramentas de gerenciamento de software disponíveis no mercado. O objetivo é proporcionar aos usuários uma interface amigável e recursos avançados de filtros e pesquisa, capacitando-os a tomar decisões informadas sobre quais softwares melhor atenderão às suas necessidades específicas.

3. Funcionalidades:

- **Pesquisa de Ferramentas:** Permitir que os usuários pesquisem por ferramentas de gerenciamento de software pelo nome.
- **Filtros de Pesquisa:** Oferecer opções de filtro baseadas em categorias para refinar os resultados da pesquisa.
- **Exibição de Ferramentas:** Apresentar uma lista organizada das principais ferramentas de gerenciamento de software.
- **Detalhes nos Cards:** Cada card de ferramenta deve incluir uma imagem, nome da ferramenta, link de redirecionamento para o site oficial e a categoria correspondente.
- **Compatibilidade de Dispositivos:** Garantir que o site seja responsivo e compatível tanto em dispositivos móveis quanto em desktop.
- **Navegação Facilitada:** Assegurar que a navegação seja intuitiva e fácil de usar, permitindo que os usuários encontrem rapidamente as informações desejadas.

4. Descrição do Problema

Quem é afetado pelo sistema?

- Profissionais de TI e Gerentes de Projetos que precisam selecionar as ferramentas adequadas para gerenciar projetos, equipes e fluxos de trabalho de forma eficiente.
- Desenvolvedores de Software que precisam de uma plataforma onde suas ferramentas possam ser apresentadas de forma clara e acessível aos potenciais usuários.

Qual é o impacto do sistema?

- Redução de Tempo e Esforço: Facilita o processo de busca e comparação de ferramentas de software, economizando tempo e esforço dos usuários.
- Aumento da Eficiência e Produtividade: As empresas e profissionais podem

escolher ferramentas que melhor atendam às suas necessidades específicas, resultando em operações mais eficientes e produtivas.

Qual seria uma boa solução para o problema?

- A solução proposta é a criação de uma plataforma web abrangente e intuitiva, que oferece recursos de mecanismo de busca, listagem de ferramenta e responsividade e navegação intuitiva

5. Principais Envolvidos e suas Características

5.1. Usuários do Sistema

- **Empresas de Tecnologia:** Empresas que desenvolvem software e precisam de ferramentas para gerenciar seus projetos, equipes e processos.
- **Startups:** Empresas em fase inicial que precisam de ferramentas flexíveis e escaláveis para apoiar seu crescimento e inovação.
- **Gerentes de Projeto:** Responsáveis pelo planejamento, execução e fechamento de projetos. Precisam de ferramentas para acompanhar o progresso, gerenciar recursos e comunicar-se com a equipe.
- **Profissionais de TI:** Encarregados de implementar, manter e suportar as ferramentas de software utilizadas pela empresa. Necessitam de uma visão clara das opções disponíveis e compatibilidade técnica.
- **Equipe de Desenvolvimento:** Desenvolvedores, designers e engenheiros de software que utilizam ferramentas de gerenciamento para coordenar tarefas, colaborar e acompanhar o desenvolvimento de software.

5.2. Desenvolvedores do Sistema

- **Gerentes de Projeto:** Responsáveis pela coordenação geral do desenvolvimento do sistema, definindo prazos, alocando recursos e acompanhando o progresso.
- **Desenvolvedores de Frontend:** Especialistas em HTML, CSS, JavaScript (React.js) que desenvolvem a interface do usuário, garantindo que o site seja

responsivo e intuitivo.

- **Desenvolvedores de Backend:** Especialistas em TypeScript - React, APIs RESTful e integração com o banco de dados.
- **Designers UX/UI:** Responsáveis pelo design da interface, focando na experiência do usuário e garantindo que o site seja fácil de navegar e visualmente atraente.
- **Analistas de Qualidade (QA):** Encarregados de testar o sistema para garantir que ele funcione corretamente e sem bugs. Realizam testes funcionais, de usabilidade e de segurança.
- **Administradores de Banco de Dados:** Profissionais responsáveis pela configuração, manutenção e otimização do banco de dados, assegurando a integridade e a segurança dos dados.

6. Regras de Negócio

Registro de Aplicativos:

- Todos os softwares de gerenciamento incluídos na plataforma devem ser registrados com informações básicas: uma imagem, nome da ferramenta, link de redirecionamento para o site oficial e a categoria correspondente.
- Cada software deve ser categorizado em uma ou mais categorias específicas.

Pesquisa Avançada:

- A plataforma deve oferecer uma ferramenta de busca que permita aos usuários encontrar softwares com base em palavras-chave, categorias, funcionalidades específicas e outros critérios relevantes.
- A pesquisa deve retornar resultados classificados pela relevância e popularidade.

Filtros Personalizados:

- Os usuários devem poder aplicar filtros específicos para refinar os resultados da busca .

- Filtros devem ser facilmente ajustáveis e permitir múltiplas seleções.

7. Requisitos Do Sistema

7.1. Requisitos Funcionais

Mecanismo de Busca:

- Permitir que os usuários pesquisem por ferramentas de gerenciamento de software.
- A pesquisa deve ser feita de acordo com o nome da ferramenta
- Opções de filtro com a categoria para refinar os resultados da pesquisa.

Listagem de Ferramentas:

- Apresentar uma lista organizada das principais ferramentas de gerenciamento de software.
- Cada card de ferramenta deve ter uma imagem, nome, link de redirecionamento para o seu respectivo site e a categoria que a ferramenta se encaixa.

Responsividade e Navegação Intuitiva:

- Garantir que o site seja responsivo compatível tanto em dispositivos móveis quanto desktop.
- Navegação intuitiva e fácil de usar para que os usuários encontrem facilmente as informações que procuram.

7.2. Requisitos Não-Funcionais

Segurança:

- Garantir a segurança dos dados sensíveis, como informações de licença e acesso.

- Implementar medidas de autenticação e autorização robustas.

Desempenho:

- Garantir que o sistema seja responsivo e tenha tempos de resposta rápidos, mesmo com grande carga de usuários e ferramentas.

Escalabilidade:

- Permitir que o sistema cresça conforme o número de usuários e ferramentas aumentam.
- Garantir que o sistema possa lidar com picos de uso sem degradação significativa do desempenho.

Manutenção:

- Facilitar a manutenção do sistema, permitindo atualizações e correções de forma rápida e eficiente.

Usabilidade:

- Desenvolver uma interface de usuário intuitiva e amigável para facilitar a navegação e uso do sistema pelos usuários.

Disponibilidade:

- Garantir que o sistema esteja disponível e acessível sempre que necessário, minimizando o tempo de inatividade não planejado.

8. Protótipo

Objetivo:

- O protótipo da interface do sistema “TechForge” visa fornecer uma plataforma interativa para os usuários navegarem por várias ferramentas e tecnologias

de desenvolvimento de software. Seu principal objetivo é oferecer acesso fácil a diferentes categorias essenciais para projetos de desenvolvimento.

Navegação:

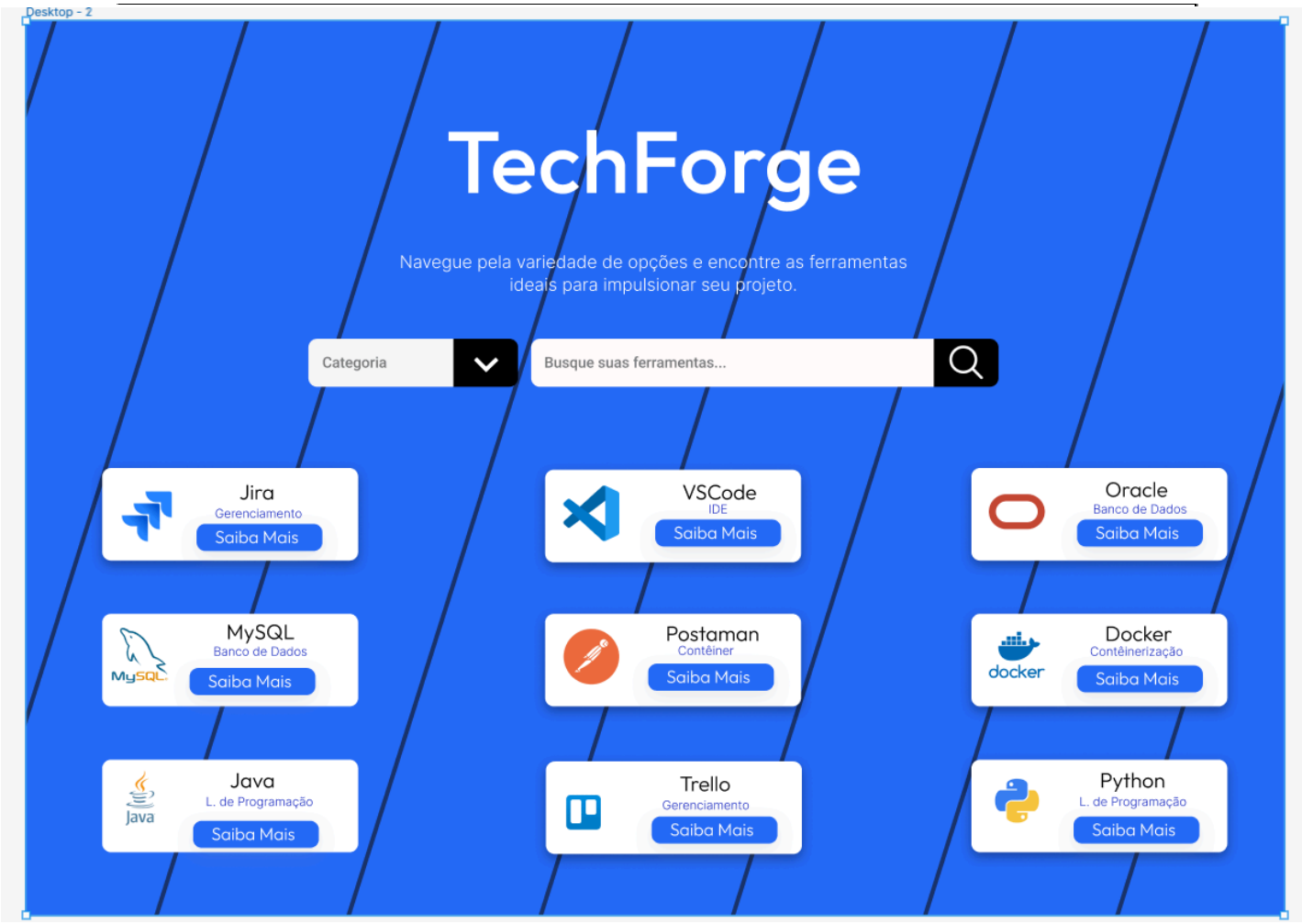
- Essa tela pode ser acessada a partir do painel principal do TechForge. Os usuários podem navegar entre os cards das ferramentas ou tecnologias clicando em uma das cartas de categoria apresentadas.

Detalhes da Tela:

- Cada card representa uma ferramenta específica e também a sua categoria, como gerenciamento de projetos, editores de código gerenciamento de bancos de dados , testes de API , contêineres , linguagens de programação e gerenciamento de tarefas .
- A barra de navegação superior inclui o logotipo do TechForge, indicando a identidade da marca, juntamente com uma função de pesquisa para acesso rápido a ferramentas ou informações dentro do TechForge.
- O foco central está na usabilidade, permitindo que os usuários encontrem suas ferramentas necessárias de forma eficiente.

Regras:

- As restrições de domínio incluem limitações de tamanho de campo apropriadas para nomes de exibição de tecnologias ou ferramentas.
- Os tipos de dados aceitos nos campos de pesquisa são strings, com valores padrão definidos com base em pesquisas comuns ou comportamento anterior do usuário.
- Os tipos de usuários permitidos podem variar; no entanto, normalmente desenvolvedores, gerentes de projeto e profissionais de TI utilizariam essa interface.



Link de acesso:

https://www.figma.com/design/RGgzLHXPo401ZYKiRkCPcv/Untitled?node-id=0-1&t=3YxKst6xiFr1jVHd-1

9. Ambiente de Desenvolvimento

Ferramentas de Desenvolvimento e Prototipação

- IDEs: Visual Studio
- Linguagem de Programação: TypeScript,
- Framework: React
- Bibliotecas:
    - "react": "^18.2.0",
    - "react-dom": "^18.2.0"

- “jsdom : "^24.0.0”
- “babel":"^7.24.5”
- “jest”: "^29.7.0”

- Prototipação: Figma.

Ferramentas de Versionamento

- Git - Sistema de controle de versão
- GitHub - Plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git

Ferramentas de Gerenciamento de Tarefas

- Notion - Plataforma de gerenciamento

10. Implementação

Componente dinâmico de Card:

O componente Card é um componente reutilizável e funcional que exibe informações de maneira organizada e estilizada. Ele é flexível e está sendo utilizado para renderizar as ferramentas de gerenciamento de software do nosso banco de dados. Ele é projetado para exibir um cartão informativo contendo uma imagem, nome, categoria e um link para mais informações. A seguir apresentaremos uma análise detalhada de cada parte do componente:

```tsx
interface CardProps {
  name: string,
  category: string,
  link: string,
  image: string
}


export function Card({name, category, link,image}:CardProps){
return(
  <div className="card">
    <img src={image} alt="Logo" />
    <div className="card-div" >
      <h2>{name}</h2>
      <p>{category}</p>
      <a href={link} target="_blank">Saiba Mais</a>
    </div>
  </div>
)
}
```

A interface CardProps define os tipos de dados que o componente Card espera receber como propriedades (props).Para usar o componente Card, você deve passar as propriedades necessárias (name, category, link, e image) ao chamá-lo. Isso renderizará um cartão com a imagem, nome, categoria e link especificados.

Página App.tsx:

O componente App é responsável por estruturar a página principal da aplicação. Ele exibe um título e subtítulo, fornece opções de navegação através de uma lista suspensa, uma barra de pesquisa, e renderiza uma lista de cartões dinâmicos usando o componente Card. Cada Card é criado com base nos dados fornecidos pelo array cards, garantindo que a interface seja flexível e fácil de atualizar ao modificar os dados subjacentes.

Arquivo cards.tsx:

O arquivo cards.tsx é um banco de dados local feito em Typescript que

exporta um array de objetos, onde cada objeto representa um cartão (card) com informações sobre diferentes ferramentas tecnológicas. Este array é utilizado no componente React App para renderizar uma lista de cartões informativos na interface do usuário.

Estrutura de Cada Objeto no Array

Cada objeto no array cards representa uma ferramenta e contém as seguintes propriedades:

- id: Um identificador único para a ferramenta.
- title: O nome da ferramenta.
- image: A URL da imagem associada à ferramenta.
- link: Um link para a página oficial ou de mais informações sobre a ferramenta.
- category: A categoria à qual a ferramenta pertence (por exemplo, "Gerenciamento", "Controle de Versão", "Teste", etc.).

Uso no Componente App:

- Renderização Dinâmica:
- O método *map* itera sobre o array cards.
- Para cada objeto filter no array, um componente Card é criado e renderizado, recebendo as propriedades name, image, link, e category com base nos valores do objeto.
- O componente Card exibe as informações da ferramenta, incluindo a imagem, título, categoria e um link para mais informações.

11. Testes

11.1. Teste de componentes

O teste de componentes verificará se os componentes estão sendo renderizados de forma correta com os tipos de dados necessários para o funcionamento esperado.

O resultado esperado para esse teste é a renderização do card com as informações de nome, link e imagem.

11.2. Casos de teste

Analisaremos passo a passo do código que está escrito em JavaScript, usando React e a biblioteca de testes **@testing-library/react,** onde verifica se os cards são renderizados corretamente na aplicação.

- Importações:

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import '@testing-library/jest-dom/extend-expect';
import App from '../App';
import cards from '../utils/data/cards';
```

React: Importamos a biblioteca React para criar componentes

Render e Screem: Utilizamos para importar funções da biblioteca de testes @testing-library/react. **Render** é usado para renderizar componentes React em um ambinte de teste e o **Screen** é usado para consultar elementos renderizados.

App: Serve para importar o componente principal da aplicação.

Cards: Importa a lista de cards a partir de um arquivo de dados.

- Interface CardProps:

```
interface CardProps {
  name: string;
  image: string;
  link: string;
  category: string;
}
```

Define uma interface TypeScript para as propriedades de um cartão. Isso é útil para garantir que os cartões recebam as propriedades corretas com os tipos de dados esperados.

- Mock do componente Card:

```
jest.mock('../components/Card', () => {
  return {
    Card: ({ name, image, link, category }: CardProps) => (
      <div data-testid="card">
        <h2>{name}</h2>
        <img src={image} alt={name} />
        <a href={link}>Link</a>
        <p>{category}</p>
      </div>
    )
  };
});
```

Jest.mock: Substitui o componente real **Card** por uma versão mock para testes. Isso é feito para controlar como o componente se comporta e para não depender de sua implementação real durante os testes.

Card.mock: Define um componente mock card que aceita name, image, link e category. como props e renderiza o diiv com esses dados. Cada div tem um atributo **data-testid** para facilitar a consulta durante os testes.

- Teste unitário:

```
test('renders all cards', () => {
  render(<App />);

  // Verifique se o número de cartões renderizados é igual ao número de cartões na
  lista
  const cardElements = screen.getAllByTestId('card');
  expect(cardElements).toHaveLength(cards.length);

  // Verifique se cada cartão tem os dados corretos
  cards.forEach(card => {
    expect(screen.getByText(card.title)).toBeInTheDocument();
    expect(screen.getByAltText(card.title)).toBeInTheDocument();
    // Use getAllByText para múltiplos elementos com o mesmo texto
    expect(screen.getAllByText(card.category).length).toBeGreaterThan(0);
  });
});
```

render(<App/>): Renderiza o componente 'App' para teste.

screen.getAllByTestId('card'): Obtém todos os elementos que tem o atributo 'data-testid = "card"'.

expect(cardElements).toHaveLength(cards.length): Verifica se o número de elementos renderizados é igual ao número de cartões la lista 'cards';

cards.forEach: Itera sobre cada card na lista e verifica os seguintes pontos:

- ❖ Se o texto do título do cartão está presente no documento.
- ❖ Se a imagem do cartão está presente no documento (verificada pelo atributo alt).
- ❖ Se a categoria do cartão está presente no documento, permitindo múltiplas ocorrências do mesmo texto.

OBS: Esse código assegura que todos os cartões são renderizados corretamente e que cada cartão contém os dados esperados (título, imagem, e categoria).

Analisaremos esse código de teste que verifica o comportamento do componente Card em uma aplicação React, usando a biblioteca de testes @testing-library/react e as extensões de asserções do Jest.

- Importação

```
// Card.test.tsx
import React from 'react';
import { render, screen } from '@testing-library/react';
import '@testing-library/jest-dom/extend-expect';
import { Card } from '../components/Card'; // Assumindo que o componente Card está no
mesmo diretório
```

React: Importa a biblioteca React, necessária para utilizar JSX.

Render e screen: Importa funções da biblioteca de testes @testing-library/react. render é usado para renderizar o componente em um ambiente de teste, e screen é usado para consultar elementos renderizados.

@testing-library/jest-dom/extend-expect': Importa extensões para asserções do Jest, como toBeInTheDocument.

Card: Importa o componente Card que será testado.

- Descrição de teste:

```
describe('Card Component', () => {
  const defaultProps = {
    name: 'Test Name',
    category: 'Test Category',
    link: 'https://example.com',
    image: 'https://via.placeholder.com/150',
  };
```

Describe: Agrupa os testes relacionados ao componente Card.

DefaultProps: Define um conjunto de propriedades padrão que serão passadas para o componente Card durante os testes.

- Primeiro Teste: Verificar a Renderização do Componente com Props Padrão

```
describe('Card Component', () => {
  test('renders the Card component with the given props', () => {
    render(<Card {...defaultProps} />);

    // Verifica se a imagem está renderizada corretamente
    const imgElement = screen.getByAltText('Logo');
    expect(imgElement).toHaveAttribute('src', defaultProps.image);

    // Verifica se o nome está renderizado corretamente
    const nameElement = screen.getByText(defaultProps.name);
    expect(nameElement).toBeInTheDocument();

    // Verifica se a categoria está renderizada corretamente
    const categoryElement = screen.getByText(defaultProps.category);
    expect(categoryElement).toBeInTheDocument();

    // Verifica se o link está renderizado corretamente
    const linkElement = screen.getByText('Saiba Mais');
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveAttribute('href', defaultProps.link);
    expect(linkElement).toHaveAttribute('target', '_blank');
  });
```

Define um teste que verifica se o componente Card é renderizado corretamente com as propriedades fornecidas.

render(<Card *{...defaultProps}* />): Renderiza o componente Card com as propriedades padrão.

screen.getByAltText('Logo'): Obtém o elemento de imagem pelo texto alternativo 'Logo'.

expect(imgElement).toHaveAttribute('src', defaultProps.image): Verifica se a imagem tem o atributo src igual ao valor esperado.

screen.getByText(defaultProps.name): Obtém o elemento que contém o nome fornecido.

expect(nameElement).toBeInTheDocument(): Verifica se o nome está presente no documento.

screen.getByText(defaultProps.category): Obtém o elemento que contém a categoria fornecida.

expect(categoryElement).toBeInTheDocument(): Verifica se a categoria está presente no documento.

screen.getByText('Saiba Mais'): Obtém o elemento de link pelo texto 'Saiba Mais'.

expect(linkElement).toHaveAttribute('href', defaultProps.link): Verifica se o link tem o atributo href correto.

expect(linkElement).toHaveAttribute('target', '_blank'): Verifica se o link abre em uma nova aba.

- Segundo Teste: Verificar a Renderização com Props Adicionais:

```
test('renders the Card component with additional props', () => {
  const additionalProps = {
    additionalInfo: 'Additional Info',
  };
  const combinedProps = { ...defaultProps, ...additionalProps };

  render(<Card {...combinedProps} />);

  // Verifica se a informação adicional está renderizada corretamente
  const additionalInfoElement = screen.getByText(additionalProps.additionalInfo);
  expect(additionalInfoElement).toBeInTheDocument();
});


});
```

additionalProps: Define propriedades adicionais que serão passadas para o componente.

combinedProps: Combina as propriedades padrão com as propriedades adicionais.

render(<Card *{...combinedProps}* />): Renderiza o componente Card com as propriedades combinadas.

screen.getByText(additionalProps.additionalInfo): Obtém o elemento que contém a informação adicional.

expect(additionalInfoElement).toBeInTheDocument(): Verifica se a informação adicional está presente no documento.

OBS: Este código realiza testes unitários no componente Card, verificando se ele é renderizado corretamente com diferentes conjuntos de propriedades. Ele assegura que todos os elementos importantes, como imagem, nome, categoria e link, estão presentes e têm os atributos corretos. Também testa a capacidade do componente de lidar com propriedades adicionais.

12. Implantação

Este capítulo tem como objetivo apresentar informações relevantes para a implantação e funcionamento do sistema.

12.1. Manual de Implantação

Pré-requisitos:

Antes de começar, certifique-se de que você tenha os seguintes softwares instalados em sua máquina:

- Node.js (recomendado: v14.x ou superior)

- npm (geralmente instalado junto com o Node.js)

Passo a Passo Windows / MacOS / Linux:

1. Clonar o Repositório

Primeiramente, clone o repositório do projeto para o seu computador. Abra o terminal e execute o seguinte comando:

```
git clone https://github.com/RobsonSantos3795/EquipeAgilGreen
```

2. Instalar as Dependências

Dentro do diretório do projeto, execute o comando para instalar todas as dependências necessárias:

```
npm install
```

3. Executar a Aplicação

Após a instalação das dependências, inicie a aplicação com o seguinte comando:

```
npm run start
```

Este comando irá iniciar o servidor de desenvolvimento e abrir a aplicação no seu navegador padrão. Geralmente, a aplicação será acessível em *http://localhost:3000* e em *http://localhost:5173*.

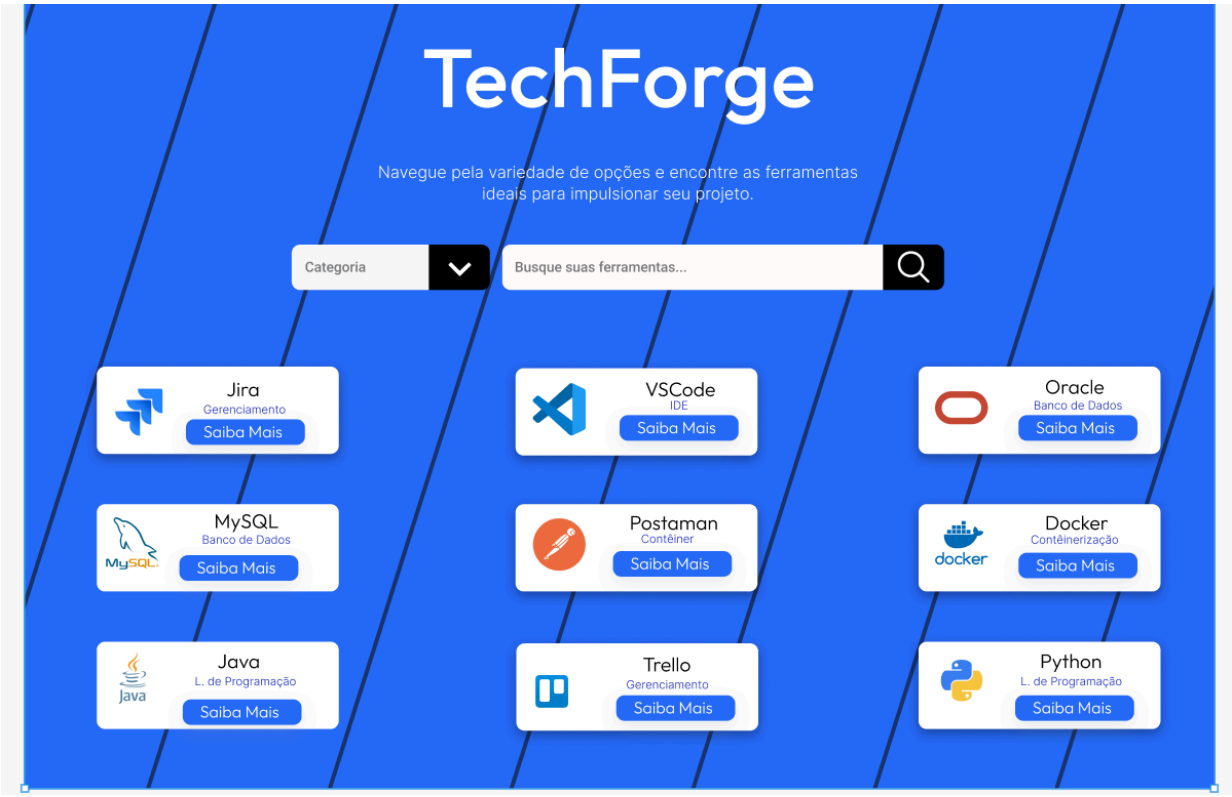
12.2. Manual do Usuário

Este manual fornecerá um guia passo a passo sobre como navegar e utilizar

todas as funcionalidades da nossa aplicação para encontrar as ferramentas ideais para impulsionar seu projeto.

1. Tela Inicial

Ao acessar o TechForge, você verá a tela inicial conforme a imagem a seguir:



2. Navegação pela Variedade de Opções

Abaixo do título "TechForge" e da descrição "Navegue pela variedade de opções e encontre as ferramentas ideais para impulsionar seu projeto", você encontrará duas seções principais de navegação: a barra de seleção de categoria e a barra de busca.

2.1. Seleção de Categoria

- Categoria: Utilize o menu suspenso para selecionar uma categoria específica de ferramentas. As opções de categorias podem incluir "Gerenciamento", "IDE", "Banco de Dados", "Contêiner", "L. de Programação", etc.
- Clique na seta para baixo para abrir o menu.

- Selecione a categoria desejada clicando nela.

2.2. Barra de Busca

- Busca: Use a barra de busca para encontrar ferramentas específicas.
- Clique na barra de busca e digite o nome da ferramenta que você está procurando.
- Clique no ícone de lupa ou pressione "Enter" para iniciar a busca.

3. Visualização de Ferramentas

Abaixo das seções de navegação, você verá a área onde as ferramentas são listadas em formato de cartões. Cada cartão representa uma ferramenta e inclui as seguintes informações:

- Nome da Ferramenta: O nome da ferramenta é exibido em destaque.
- Categoria: A categoria à qual a ferramenta pertence.
- Imagem: Um ícone ou logotipo da ferramenta.
- Link "Saiba Mais": Um botão que leva ao site oficial ou a uma página com mais informações sobre a ferramenta.

4. Interação com os Cartões de Ferramentas

4.1. Visualizar Detalhes da Ferramenta

Para obter mais informações sobre uma ferramenta específica, clique no botão "Saiba Mais" dentro do cartão correspondente. Isso abrirá uma nova aba no seu navegador com mais detalhes sobre a ferramenta.

5. Exemplos de Uso

5.1. Selecionar uma Categoria

1. Clique na seta para baixo ao lado de "Categoria".
2. Selecione "Gerenciamento" da lista.
3. A lista de ferramentas será filtrada para mostrar apenas as ferramentas

de gerenciamento.

5.2. Buscar uma Ferramenta Específica

1. Clique na barra de busca.
2. Digite "Postman".
3. Clique no ícone de lupa ou pressione "Enter".
4. A lista de ferramentas será filtrada para mostrar apenas o Postman.

5.3. Saber Mais Sobre uma Ferramenta

1. Encontre o cartão da ferramenta "Jira".
2. Clique no botão "Saiba Mais".
3. Uma nova aba será aberta no seu navegador com mais informações sobre o Jira.

6. Suporte e Ajuda

Se você encontrar algum problema ou tiver dúvidas sobre o uso do TechForge, entre em contato com o suporte.

13. Conclusões e Considerações Finais

Tech Forge

Versão 2.0

Autores:

- Beatriz Andrade Siquara
- Breno Pimentel de Almeida Miranda
- Luan Gabriel Santos Paim Dias
- Pablo Santana dos Santos
- Robson Graça dos Santos

- Yana Barreto Luiz Simina

Revisor: Prof. Felipe Oliveira

A plataforma Tech Forge 2.0 apresentou-se como uma solução inovadora e eficaz para a organização e seleção de ferramentas de gerenciamento de software. O projeto foi guiado pela necessidade crescente de centralizar informações e facilitar a tomada de decisão para profissionais e empresas que buscam as melhores ferramentas para suas necessidades específicas.

Aplicabilidade dos Resultados

A plataforma atendeu plenamente aos objetivos estabelecidos, proporcionando uma interface intuitiva e recursos avançados de busca e filtragem. A Tech Forge não só simplifica o processo de busca por ferramentas de gerenciamento de software, mas também oferece uma experiência de usuário satisfatória, possibilitando que os profissionais economizem tempo e recursos.

Principais Benefícios

- **Redução de Tempo e Esforço:** A plataforma facilita a pesquisa e comparação de ferramentas, otimizando o tempo gasto pelos usuários.
- **Aumento da Eficiência:** Ao possibilitar escolhas informadas, a plataforma contribui para operações mais eficientes e produtivas.
- **Interface Intuitiva:** O design amigável assegura uma navegação fácil, melhorando a usabilidade do sistema.

Limitações

Apesar dos avanços significativos, algumas limitações foram identificadas:

- **Atualização de Dados:** A manutenção constante da base de dados é crucial para garantir a relevância das informações.
- **Escalabilidade:** Embora o sistema tenha sido projetado para ser escalável, o

aumento substancial de usuários e ferramentas pode demandar ajustes e otimizações adicionais.

Considerações Finais

O desenvolvimento da Tech Forge 2.0 representa um passo significativo na direção de fornecer um recurso robusto e eficiente para profissionais e empresas que buscam melhorar seu gerenciamento de software. A plataforma não só cumpre com suas promessas iniciais, mas também abre caminho para futuras inovações e expansões.

O projeto foi um esforço conjunto dos autores e do revisor, cujo comprometimento e dedicação foram fundamentais para o sucesso alcançado. Agradecemos a todos os envolvidos e esperamos que a Tech Forge continue a evoluir e atender às necessidades de seus usuários de maneira ainda mais eficaz.