

Documentação do Projeto Hermes

Membros:

Pedro Carneiro, Rafael Teixeira, Vitoria Cardoso, Isabelle Maciel, Gabriel

1. Introdução

1.1 Apresentação

O projeto Hermes é uma iniciativa inovadora que visa criar uma plataforma web abrangente e intuitiva para facilitar a descoberta, organização e utilização de software livre. Com uma interface amigável e recursos avançados de pesquisa, o Hermes pretende ser o destino central para indivíduos e organizações que buscam soluções de software de código aberto.

Além de simplesmente localizar software livre, o Hermes se propõe a oferecer uma experiência completa aos usuários, fornecendo informações detalhadas sobre cada aplicativo, incluindo sua funcionalidade, requisitos de sistema, documentação, e até mesmo tutoriais e guias de uso. Essa abordagem holística visa capacitar os usuários a tomarem decisões informadas sobre quais softwares atenderão melhor às suas necessidades específicas.

1.2 Objetivos do Projeto

O projeto Hermes tem como objetivo promover a comunidade de software livre, incentivando colaborações, contribuições e feedbacks. Ao facilitar o acesso ao software livre e promover uma cultura de compartilhamento e transparência, o Hermes contribui para o crescimento e aprimoramento contínuo do ecossistema de código aberto.

Além disso, o Hermes pode incluir recursos adicionais, como integração com plataformas de desenvolvimento colaborativo, fóruns de discussão, e até mesmo funcionalidades de recomendação com base nas preferências e nas necessidades dos usuários.

2. Escopo do Projeto

O escopo deste projeto inclui:

- Desenvolvimento de um aplicativo web responsivo.
- Renderiza aplicativos/softwarewares que auxiliam no desenvolvimento e qualidade/efetividade de projetos.
- Testes unitários para garantir a robustez e funcionalidade do código.

3. Recursos Necessários

3.1 Equipe de Desenvolvimento

(**Front-End:** Irvin Marques, Bia Siquara, Ian Menezes. **Q.A:** Victor Alves, Irvin Marques. **Gerente de Projeto:** Irvin Marques)

3.2 Documentação e Pesquisa

Pedro Carneiro, Rafael Teixeira, Isabelle Maciel, Vitória Cardoso.

4 Requisitos Funcionais

4.1 Ferramentas de desenvolvimento

(**IDEs:** Visual Studio , **Frameworks:** Jest, Testing Library, **Biblioteca:** React, **Linguagem de programação:** JavaScript, **Prototipação:** Figma).

4.2 Ferramentas de versionamento

- Git (é um sistema de controle de versões distribuído)
- GitHub (é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git)
- GitDesktop (é um gerenciador de repositório de software baseado em Git)

4.3 Ferramentas de gerenciamento de tarefas

- Trello (plataforma de gerenciamento)
link: <https://trello.com/invite/accept-board>

4.4 Ferramentas de teste unitário

- Testing Library (frameworks)
- Jest (frameworks)

4.5 Ferramentas de teste de Caixa Preta

- Gtmetrix
- Domsignal

5. Requisitos não Funcionais

5.1 Desempenho

garantir que a aplicação seja responsiva e rápida com tempos de carregamentos mínimos

5.2 Usabilidade

priorizar a usabilidade e a experiência do usuário, com uma interface intuitiva, design limpo e acessibilidade adequada

5.3 Compatibilidade

garantir compatibilidade com uma variedade de navegadores web, sistemas operacionais e dispositivos, para garantir uma experiência consistente para todos os usuários

6. Estratégias de Testes

6.1 Estratégia de Testes Unitários

As estratégias de testes unitários incluem:

- **Identificação de Casos de Teste:** Identificar todos os casos de teste relevantes para as unidades de código.
- **Implementação de Testes Unitários:** Desenvolver testes unitários para cada unidade de código, cobrindo diferentes cenários.
- **Automação de Testes:** Automatizar os testes unitários para garantir que sejam executados regularmente durante o desenvolvimento.
- **Integração com Ferramentas de CI/CD:** Integrar os testes unitários ao pipeline de integração contínua/entrega contínua para garantir testes regulares e automáticos através do **GitHub Actions**.
- **Cobertura de Código:** Monitorar e manter a cobertura de código dos testes unitários para garantir uma cobertura adequada do código.
- **Revisão de Código:** Revisar os testes unitários como parte do processo de revisão de código para garantir sua eficácia e qualidade.

6.2 Estratégias de Testes de Caixa Preta

7. Responsabilidades

- **Equipe de Desenvolvimento:** Responsáveis pelo desenvolvimento do aplicativo web e auxílio na criação dos testes.
- **Equipe de Documentação e Pesquisa:** Responsáveis pela elaboração da documentação e pesquisa do Projeto Hermes.
- **Equipe de Testadores:** Responsáveis por desenvolver e executar os testes unitários, além de fornecer feedback sobre a qualidade do código.

8. Metodologia Ágil

8.1 Introdução

- É uma forma de conduzir projetos que busca dar maior rapidez aos processos e à conclusão de tarefas. Baseia-se em um fluxo de trabalho mais ágil, flexível, sem tantos obstáculos, com total interatividade

8.2 Modelo de Processo

8.2.1 Scrum

- É um framework de gerenciamento que utilizamos para nos auto-organizar e trabalhar em direção a um objetivo em comum. A estrutura descreve um conjunto de reuniões, ferramentas e funções para uma entrega eficiente de projetos.

Etapa	Duração Estimada	Data de Início	Data de Conclusão
Planejamento	1 dia	18/03	18/03
Design	2 dias	19/03	20/03
Desenvolvimento	9 semanas	21/03	—
Testes Unitários	2 semanas	27/03	—
Testes Integrados	2 semanas	—	—
Revisão e Lançamento	1 semana	—	—

9. Controle de Mudanças

Qualquer alteração no escopo, cronograma ou recursos do projeto deve ser submetida à aprovação do gerente de projeto antes de ser implementada.

Para o desenvolvimento do projeto, foram considerados diversos modelos de processo de software, incluindo cascata, prototipação, incremental e espiral. Após uma análise cuidadosa das características do projeto, foi escolhido o modelo incremental como o mais adequado.

10. Arquivo de Filtros

10.1 Descrição

- O filtro no componente Body permite aos usuários selecionar uma categoria específica de cards a serem exibidos na seção. Quando um filtro é selecionado, apenas os cards associados à categoria correspondente são mostrados, enquanto os outros são ocultados.

10.2 Funcionalidades

- **Estado:** "activeFilter" é o estado interno do componente React que mantém o filtro atualmente selecionado pelo usuário. Ele é inicializado com o título do primeiro filtro em "filterData".
- **Função de alteração do filtro:** Quando um usuário seleciona um filtro, a função "handleChangeActiveCard" é chamada. Esta função atualiza o estado "activeFilter" com o título do filtro selecionado e, em seguida, atualiza os cards ativos correspondentes à categoria selecionada.
- **Dados do filtro:** O filtro é baseado em um conjunto de dados "filterData", que contém informações sobre as categorias disponíveis. Cada objeto dentro de "filterData" inclui um título e um identificador único para representar uma categoria específica.

10.3 Blocos do código

```
const [activeFilter, setActiveFilter] = React.useState(filterData[0].title);  
// Estado para os cards ativos  
const [activeCards, setActiveCards] = React.useState(cards[filterData[0].id]);  
  
// Função para alterar o card ativo
```

```

function handleChangeActiveCard(filter) {
  setActiveFilter(filter.title)
  setActiveCards(cards[filter.id]);
}

return (
  <div className="bg-white relative ">
    <div className="navigation-wrapper z-10 transform -translate-y-[40px]">
      <div ref={sliderRef} className="keen-slider w-full overflow-hidden">
        {filterData.map((filter, index) => (
          <div key={index} className="keen-slider__slide min-w-fit max-w-fit">
            <Button
              active={activeFilter === filter.title}
              onClick={() => handleChangeActiveCard(filter)}
            >
              {filter.title}
            </Button>
          </div>
        ))}
      </div>
      <div className="flex gap-10 items-center justify-center">
        {loaded && instanceRef.current && (
          <>
            <Arrow
              left
              onClick={(e) =>
                e.stopPropagation() || instanceRef.current?.prev()
              }
              disabled={currentSlide === 0}
            />
            <Arrow
              onClick={(e) =>
                e.stopPropagation() || instanceRef.current?.next()
              }
              disabled={
                currentSlide ===
                instanceRef.current.track.details.slides.length - 1
              }
            />
          </>
        )}
      </div>
    </div>
  )}

```