UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL NASCIMENTO SANTOS

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE INICIALIZAÇÃO DE PROJETOS WEB

CAMPINA GRANDE - PB

2023

GABRIEL NASCIMENTO SANTOS

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE INICIALIZAÇÃO DE PROJETOS WEB

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Dalton Dario Serey Guerrero

CAMPINA GRANDE - PB

2023

GABRIEL NASCIMENTO SANTOS

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE INICIALIZAÇÃO DE PROJETOS WEB

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Dalton Dario Serey Guerrero

Orientador – UASC/CEEI/UFCG

Wilkerson de Lucena Andrade

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 28 de JUNHO de 2023.

CAMPINA GRANDE - PB

RESUMO

Com o crescente avanço do ecossistema web nos últimos anos, tem havido um surgimento constante de novos frameworks e bibliotecas. Entre eles, o ReactJS, lançado em 2013 pela Meta, destacou-se como uma poderosa ferramenta para a criação de interfaces interativas, complexas e eficientes, baseadas em componentes. Atualmente, o ReactJS possui uma comunidade ativa de desenvolvedores e conta com uma sólida documentação, consolidando-se como uma das principais tecnologias utilizadas no desenvolvimento web.

Devido à sua ampla adoção em grandes empresas de diversos setores ao redor do mundo e ao crescente uso em novas aplicações web, a Meta e outras empresas estão constantemente buscando maneiras de facilitar e agilizar o início de projetos em React. Como resultado, surgiram ferramentas como o create-react-app (desenvolvido pela própria Meta), Vite e Parcel. O objetivo deste trabalho é analisar e comparar essas bibliotecas, considerando diferentes parâmetros, como o tempo de instalação utilizando os gerenciadores de pacotes mais comuns, npm e yarn, facilidade de configuração do projeto, estrutura de pastas e arquivos criados, bem como a qualidade e legibilidade do código necessário para sua utilização.

Como contribuição, serão apresentadas sugestões para o aprimoramento dessas aplicações, levando em consideração os aspectos avaliados. O estudo visa oferecer insights valiosos para desenvolvedores que desejam iniciar projetos em React, auxiliando na escolha da melhor ferramenta e fornecendo recomendações para uma experiência mais eficiente e eficaz.

COMPARATIVE STUDY BETWEEN WEB PROJECT BUILD TOOLS

ABSTRACT

With the growing advancement of the web ecosystem in recent years, there has been a constant emergence of new frameworks and libraries. Among them, ReactJS, released in 2013 by Meta, stood out as a powerful tool for creating interactive, complex and efficient interfaces based on components. Currently, ReactJS has an active community of developers and has solid documentation, consolidating itself as one of the main technologies used in web development.

Due to its widespread adoption in large companies across industries around the world and its increasing use in new web applications, Meta and other companies are constantly looking for ways to make it easier and faster to start projects in React. As a result, tools such as create-react-app (developed by Meta itself), Vite and Parcel have emerged. The objective of this work is to analyze and compare these libraries, considering different parameters, such as installation time using the most common package managers, npm and yarn, ease of project configuration, structure of folders and files created, as well as the quality and readability of the code necessary for its use.

As a contribution, suggestions will be presented for the improvement of these applications, taking into account the evaluated aspects. The study aims to offer valuable insights for developers who want to start projects in React, helping them choose the best tool and providing recommendations for a more efficient and effective experience.

Estudo comparativo entre ferramentas de inicialização de projetos web

Gabriel Nascimento Santos
gabriel.nascimento.santos@ccc.ufcg.edu.br
Universidade Federal de Campina Grande (UFCG)
Campina Grande, Paraíba, BR

Dalton Dario Serey Guerrero
dalton@computacao.ufcg.edu.br
Universidade Federal de Campina Grande (UFCG)
Campina Grande, Paraíba, BR

ABSTRACT

With the growing advancement of the web ecosystem in recent years, there has been a constant emergence of new frameworks and libraries. Among them, ReactJS, released in 2013 by Meta, stood out as a powerful tool for creating interactive, complex and efficient interfaces based on components. Currently, ReactJS has an active community of developers and has solid documentation, consolidating itself as one of the main technologies used in web development.

Due to its widespread adoption in large companies across industries around the world and its increasing use in new web applications, Meta and other companies are constantly looking for ways to make it easier and faster to start projects in React. As a result, tools such as create-react-app (developed by Meta itself), Vite and Parcel have emerged. The objective of this work is to analyze and compare these libraries, considering different parameters, such as installation time using the most common package managers, npm and yarn, ease of project configuration, structure of folders and files created, as well as the quality and readability of the code necessary for its use.

As a contribution, suggestions will be presented for the improvement of these applications, taking into account the evaluated aspects. The study aims to offer valuable insights for developers who want to start projects in React, helping them choose the best tool and providing recommendations for a more efficient and effective experience.

RESUMO

Com o crescente avanço do ecossistema web nos últimos anos, tem havido um surgimento constante de novos frameworks e bibliotecas. Entre eles, o ReactJS, lançado em 2013 pela Meta, destacou-se como uma poderosa ferramenta para a criação de interfaces interativas, complexas e eficientes, baseadas em componentes. Atualmente, o ReactJS possui uma comunidade ativa de desenvolvedores e conta com uma sólida documentação, consolidando-se como uma das principais tecnologias utilizadas no desenvolvimento web.

Devido à sua ampla adoção em grandes empresas de diversos setores ao redor do mundo e ao crescente uso em novas aplicações web, a Meta e outras empresas estão constantemente buscando maneiras de facilitar e agilizar o início de projetos em React. Como resultado, surgiram ferramentas como o create-react-app (desenvolvido pela própria Meta), Vite e Parcel. O objetivo deste trabalho é analisar e comparar essas bibliotecas, considerando diferentes parâmetros, como o tempo de instalação utilizando os gerenciadores de pacotes mais comuns, npm e yarn, facilidade de configuração do projeto, estrutura de pastas e arquivos criados, bem como a qualidade e legibilidade do código necessário para sua utilização.

Como contribuição, serão apresentadas sugestões para o aprimoramento dessas aplicações, levando em consideração os aspectos avaliados. O estudo visa oferecer insights valiosos para desenvolvedores que desejam iniciar projetos em React, auxiliando na escolha da melhor ferramenta e fornecendo recomendações para uma experiência mais eficiente e eficaz.

Palavras-chave

Desenvolvimento web, bibliotecas, react

1. INTRODUÇÃO

A área de desenvolvimento web é conhecida por oferecer inúmeras oportunidades de emprego dentro do campo da programação. Consequentemente, essa área recebe um grande apoio da comunidade, refletido no número de bibliotecas que são constantemente atualizadas com novas funcionalidades. Segundo dados do npm trends, o React tem sido o framework mais utilizado no desenvolvimento web desde 2015, quando o npm começou a comparar dados relacionados ao número de downloads. Houve um breve período, em novembro de 2022, em que o React foi ultrapassado pelo Svelte e, em seguida, pelo Vue. No entanto, essa breve mudança não diminui a popularidade e a consistência do React ao longo dos anos.

Após sete anos no topo, o React não apenas amadureceu, mas todo o seu ecossistema também evoluiu, incluindo suas bibliotecas, que contam com um grande apoio da comunidade. Em dezembro de 2022, foi lançada a versão 4.0 do Vite, uma biblioteca para criação de projetos em React, juntamente com alguns números promissores para o seu futuro. A versão 3.0 do Vite foi lançada cinco meses antes, e o número de downloads semanais no npm era de 1 milhão. Esse número aumentou para 2,5 milhões desde o lançamento da versão 4.0. Além disso, de acordo com uma pesquisa realizada durante a Jamstack Conf, uma conferência de desenvolvimento, a utilização do Vite entre os desenvolvedores aumentou de 14% para 32%.

De acordo com a CNN, a demanda por profissionais da área de tecnologia cresceu mais de 670% em 2020. Como resultado, surgiram diversas ferramentas voltadas para esse campo. Este trabalho, no entanto, concentra-se nas bibliotecas para criação de projetos, mais especificamente as relacionadas ao React. Esse tipo de ferramenta tem como objetivo principal otimizar o tempo de

desenvolvimento, oferecendo um modelo inicial para o projeto e possibilitando a customização de acordo com diferentes necessidades.

Caso um desenvolvedor opte por iniciar um novo projeto em React sem o auxílio de uma biblioteca, será necessário realizar várias configurações manuais, como a criação de pastas, do arquivo package.json e a inicialização do git, apenas para citar algumas atividades. Caso o projeto seja em TypeScript, serão necessárias ainda mais configurações. Embora essa abordagem ofereça total liberdade ao desenvolvedor, ela não é prática em um ambiente com prazos e entregas. Por esse motivo, existem bibliotecas como o Parcel, que automatizam essas configurações iniciais.

O objetivo deste trabalho é analisar ferramentas como Vite, Parcel e create-react-app, algumas mais conhecidas do que outras dentro do ecossistema web mas todas relevantes. Pretendemos entender o funcionamento dessas ferramentas e destacar suas diferenças, a fim de realizar uma comparação entre elas e fornecer indicações sobre qual seria a melhor opção, levando em consideração os objetivos do desenvolvedor, o tamanho do projeto e o design de código. Com base nesse aprendizado, também esperamos fornecer um conjunto de sugestões para a melhoria dessas bibliotecas.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção vai dar uma visão geral sobre as diferentes bibliotecas estudadas neste trabalho, assim como também conceitos importantes para utilização de tais bibliotecas, por exemplo os gerenciadores de pacote.

Conceitos como tree shaking e lazy loading são abordados posteriormente no texto, e desempenham um papel importante na otimização de aplicações. Tree shaking é um processo realizado durante a fase de build do código que tem como objetivo remover código exportado que não é utilizado. Isso resulta em um bundle mais leve, eliminando partes desnecessárias do código. Com a remoção de código não utilizado, a aplicação tende a ter um melhor desempenho, uma vez que há menos código para ser processado e executado pelo navegador.

Por outro lado, o conceito de lazy loading está relacionado à otimização do carregamento de uma aplicação. Com o lazy loading, as partes mais pesadas e demoradas da aplicação, como grandes bibliotecas ou seções complexas, são carregadas apenas quando necessário, ou seja, sob demanda. Isso evita que o carregamento de toda a página seja comprometido, especialmente em dispositivos móveis ou em situações em que a conexão de internet é mais lenta. O lazy loading permite que o usuário tenha uma experiência de carregamento mais ágil, visto que apenas o conteúdo essencial é carregado inicialmente, e o restante é carregado conforme necessário.

2.1 Gerenciadores de pacote

Antes de prosseguirmos com a discussão das bibliotecas em si, é importante entender como utilizá-las. Para isso, é necessário um gerenciador de pacotes, que é uma ferramenta que permite o uso de pacotes desenvolvidos por outros programadores. Esses pacotes podem variar desde módulos contendo funções prontas para serem utilizadas em uma aplicação, evitando a necessidade de criar essas funcionalidades do zero, até frameworks que fornecem estruturas completas para o desenvolvimento de uma aplicação. Os pacotes são disponibilizados publicamente, permitindo que outras pessoas os utilizem.

Embora seja possível instalar pacotes manualmente, essa abordagem não é prática. Ela pode levar a esquecer de instalar dependências necessárias do pacote e demanda um tempo considerável. É nesse ponto que os gerenciadores de pacotes entram em cena. Eles facilitam o processo de versionamento, atualização, instalação e remoção dos pacotes. Ao serem chamados, eles leem um arquivo *package.json* localizado na pasta raiz do projeto, que contém uma lista de todos os pacotes utilizados e suas versões correspondentes. Além disso, esses gerenciadores também atualizam o arquivo JSON com os novos pacotes instalados e suas versões.

Existem várias opções de gerenciadores de pacotes disponíveis. O npm, é uma das opções mais comuns. O yarn se apresenta como uma alternativa mais rápida, enquanto o pnpm é uma opção mais recente que se propõe a ser ainda melhor e mais eficiente que seus concorrentes.

Apesar das outras opções disponíveis, o npm é por uma grande margem a ferramenta de gerenciamento de pacotes mais popular que temos, consequentemente é utilizada nos guias oficiais como uma das opções recomendadas pelos desenvolvedores por todas as bibliotecas estudadas neste trabalho, por esses motivos foi a ferramenta escolhida para realizar os testes e manter os resultados consistêntes.

2.1.1 Node Package Manager (NPM)

O npm desempenha um papel fundamental no desenvolvimento de projetos, proporcionando uma maneira eficiente de lidar com as dependências de um projeto e garantir a consistência do ambiente de desenvolvimento. Para compreender seu funcionamento, é importante explorar os princípios conceituais que sustentam essa ferramenta.

Ao executar o comando de instalação no terminal, todas as dependências listadas no arquivo package.json são buscadas no registro de pacotes do npm para realizar o download correspondente. É possível também fornecer flags para buscar apenas dependências e versões específicas, o que contribui para um controle mais preciso do ambiente de desenvolvimento.

Para compreender o funcionamento desse processo, é importante analisar a estrutura de arquivos típica de projetos que utilizam gerenciadores de pacote. A presença da pasta node_modules e do arquivo package-lock.json desempenha um papel crucial no controle das dependências instaladas. O primeiro armazena os pacotes baixados, enquanto o segundo garante a consistência das versões utilizadas.

No cerne do gerenciamento de dependências está o algoritmo utilizado pelos gerenciadores de pacote. Esse algoritmo verifica a existência da pasta node_modules e do arquivo package-lock.json. Em seguida, compara as árvores de dependências, adicionando as dependências ausentes e realizando ações como instalação, atualização e remoção. A priorização de instalações no topo da árvore é realizada por questões de desempenho e facilita futuras atualizações. O registro público do npm é uma base de dados amplamente utilizada no ecossistema do desenvolvimento JavaScript. Essa base de dados é alimentada por

desenvolvedores e empresas que contribuem com pacotes e bibliotecas para a comunidade.

No entanto, o npm oferece a flexibilidade de utilizar outros registros compatíveis, incluindo a opção de configurar um registro personalizado. Essa capacidade permite que os usuários adaptem o gerenciador de pacotes de acordo com suas necessidades específicas, como a utilização de repositórios internos ou privados.

A publicação de pacotes no registro oficial do npm é outra funcionalidade importante fornecida pelo gerenciador de pacotes. Ao definir as configurações adequadas no arquivo package.json, os desenvolvedores podem escolher se desejam disponibilizar seus pacotes para uso público ou mantê-los privados. Essa flexibilidade permite que projetos internos ou proprietários sejam gerenciados de forma eficaz, ao mesmo tempo em que promove a colaboração e o compartilhamento de bibliotecas na comunidade.

Além disso, o npm oferece recursos adicionais para facilitar o trabalho em equipe e a colaboração. Uma dessas funcionalidades é a criação de times, nos quais membros podem ser adicionados e ter permissões específicas para compartilhar bibliotecas entre si. Essa funcionalidade é particularmente útil em projetos maiores ou quando há a necessidade de controle de acesso a pacotes específicos.

Todas essas configurações e opções são definidas no arquivo package.json, que é a configuração central do projeto no npm. Nesse arquivo, além das configurações relacionadas ao registro e à publicação, é possível especificar a versão do pacote, a licença, as dependências e outras informações pertinentes ao projeto.

2.2 Create React App (CRA)

O Create React App (CRA) é um projeto desenvolvido pelo Facebook que tem como proposta gerar um boilerplate inicial para projetos em ReactJS. Ele visa fornecer uma única dependência sem a necessidade de configuração, permitindo aos desenvolvedores iniciar rapidamente projetos em React de forma simples e eficiente. O CRA também oferece flexibilidade, permitindo a personalização do projeto de acordo com as necessidades específicas do desenvolvedor.

A ideia é fornecer uma estrutura sólida o suficiente para que o usuário possa iniciar um projeto imediatamente, ao mesmo tempo em que oferece a flexibilidade necessária para personalização e customização do projeto.

Para utilizar o CRA, o usuário pode escolher entre os gerenciadores de pacote npm ou yarn. Ao ser executado no terminal, o CRA inicia uma command line interface que se comunica com o npm ou yarn via protocolo HTTPS. Durante o processo de inicialização, são baixadas as dependências e pacotes necessários do node_modules, com um progresso visual exibido na interface do CRA.

O CRA também realiza testes automatizados em sistemas operacionais populares para desktops/laptops, como Linux (Ubuntu), MacOS e Windows, verificando sua compatibilidade e interface em diferentes ambientes.

Durante a criação dos arquivos do projeto, o CRA inclui o arquivo .gitignore, que configura no Git quais arquivos e pastas não devem ser salvos no repositórtio, evitando fazer controle de versões de arquivos que podem ser gerados pelo ferramental a partir do código fonte. O diretório node_modules também é criado, contendo módulos de terceiros disponíveis no GitHub e no npm.js, que podem ser importados e utilizados no projeto.

O CRA faz uso de configurações em cache local, como o Babel, responsável por garantir compatibilidade com browsers mais antigos, o Jest para testes unitários, o ESLint para verificar o estilo do código com base em regras definidas pelo desenvolvedor, e o Webpack, um empacotador de módulos JavaScript amplamente utilizado .

Em resumo, o CRA simplifica o processo de criação e configuração de projetos em React, oferecendo uma estrutura inicial pronta para uso, mas também permitindo a personalização e customização do projeto conforme necessário. Ele é amplamente utilizado pela comunidade de desenvolvedores e possui uma documentação abrangente e uma comunidade ativa de suporte.

2.3 Vite

Vite é uma ferramenta de interface de linha de comando projetada para aprimorar a experiência de desenvolvimento em projetos web modernos. Desenvolvida com o objetivo de ser rápida e enxuta, a equipe por trás do Vite busca fornecer apenas o necessário. A ferramenta consiste em duas partes principais: um servidor de desenvolvimento para executar a aplicação durante o processo de criação e um comando de build que empacota e otimiza o código para a produção.

Uma das estratégias adotadas pelo Vite para melhorar a velocidade do servidor é a divisão dos módulos da aplicação em duas categorias distintas: dependências e código-fonte. As dependências são pacotes que não sofrem alterações frequentes durante o desenvolvimento e podem ser custosas em termos de desempenho. Para lidar com esse desafio, o Vite utiliza um pré-empacotamento dessas dependências, empregando uma ferramenta de construção escrita em Go, que é significativamente mais rápida do que as alternativas baseadas em JavaScript, oferecendo ganhos de velocidade de 10 a 100 vezes superiores.

Já em relação ao código-fonte, que necessita de transformação e edição constante, o servidor do Vite, ao executar a aplicação, envia o código para o navegador por meio de Módulos de JavaScript nativos. Isso permite que o Vite transforme apenas a parte do código solicitada pelo navegador, construindo-o sob demanda, em vez de processar a aplicação inteira de uma vez. Essa abordagem contribui para um desenvolvimento mais ágil e eficiente.

Outro benefício oferecido pelo Vite é a funcionalidade de Hot Reload, que permite atualizar o servidor de desenvolvimento automaticamente ao realizar alterações em um arquivo e salvá-lo. No entanto, o Vite observou que, à medida que a aplicação crescia, o Hot Reload padrão perdia desempenho. Para solucionar esse problema, o Vite utiliza módulos JavaScript nativos para recarregar apenas o módulo em questão na maioria dos casos, otimizando a performance durante o processo de desenvolvimento.

Embora o Vite empregue módulos JavaScript padrão para a maioria das tarefas, a geração do pacote final para produção ainda enfrenta alguns desafios em relação a problemas de rede ao utilizar imports aninhados. Para lidar com essa questão, o Vite utiliza estratégias como tree-shaking e lazy-loading, a fim de

melhorar o armazenamento em cache e otimizar a eficiência do pacote gerado para produção.

Em suma, o Vite se destaca como uma ferramenta que proporciona uma experiência de desenvolvimento mais ágil e eficiente para projetos web modernos. Seu enfoque na velocidade, divisão de dependências, utilização de módulos JavaScript nativos e funcionalidade de Hot Reload contribuem para um fluxo de trabalho mais produtivo, enquanto suas estratégias de otimização, como tree-shaking e lazy-loading, melhoram o desempenho do pacote final para produção.

2.4 Parcel

Uma opção estudada é o Parcel, uma ferramenta que se destaca por sua capacidade de inicializar projetos sem a necessidade de etapas de configuração adicionais. Basta executar um comando no terminal e o projeto está pronto para uso. O Parcel se beneficia do uso de múltiplos núcleos presentes nas CPUs modernas e do conceito de concorrência, aproveitando as threads da CPU para extrair o máximo desempenho e obter alta performance.

O compilador de JavaScript do Parcel é escrito em Rust, o que resulta em uma melhoria de velocidade de 10 a 20 vezes em comparação com outras ferramentas similares baseadas em JavaScript. Essa escolha de linguagem oferece vantagens significativas em termos de desempenho. Além disso, o Parcel utiliza recursos de cache, monitorando os arquivos em um nível mais baixo, mais próximo da máquina, o que permite uma resposta mais rápida quando uma alteração é feita. O Parcel também realiza a compressão e otimização das imagens utilizadas no projeto, reduzindo o tamanho dos arquivos e melhorando o desempenho.

Outra característica interessante é que o Parcel realiza a construção (build) somente dos arquivos que são solicitados pelo navegador, o que reduz consideravelmente o tempo necessário para inicializar o projeto. Essa abordagem se mostra eficiente ao evitar a necessidade de reconstruir todo o projeto a cada alteração. É evidente que o Parcel tem um forte foco em performance e muitas de suas funcionalidades são direcionadas para essa área. Além disso, a ferramenta é altamente extensível por meio de uma ampla variedade de plugins disponíveis, o que possibilita a personalização de acordo com as necessidades específicas do projeto.

Em resumo, o Parcel se destaca como uma ferramenta que simplifica a inicialização de projetos ao eliminar etapas de configuração adicionais. Seu aproveitamento dos recursos de multiprocessamento das CPUs modernas, o uso do compilador de JavaScript escrito em Rust, a utilização de cache e a construção seletiva de arquivos contribuem para um desempenho superior. O Parcel também se beneficia da compressão e otimização de imagens, além de sua extensibilidade por meio de plugins.

2.5 Popularidade

Outro aspecto interessante a ser analisado é a popularidade das ferramentas abordadas neste estudo. Para essa análise, foram considerados os dados do último ano, tanto do npm trends, que registra o número de downloads, quanto do Google Trends, que mostra a frequência em que o termo foi pesquisado. Através desses dados, podemos observar as tendências de popularidade ao longo dos últimos 12 meses.

No caso do Vite, é possível verificar um crescimento constante em ambos os gráficos de pesquisa apresentados na Figura 1 e Figura 2. Tanto o número de downloads registrados no npm quanto a quantidade de pesquisas realizadas no Google demonstram um aumento gradual de interesse pela ferramenta ao longo desse período.
Já o Create React App apresentou um pico de popularidade no mês de abril de 2023, porém, esse aumento foi observado apenas nos dados do npm trends. No Google Trends, não houve um aumento significativo de buscas pelo termo. Essa diferença pode indicar que o interesse pelo Create React App está mais relacionado ao seu uso específico dentro da comunidade de desenvolvedores, refletindo-se principalmente nos downloads registrados.

Por fim, o Parcel se destaca como uma ferramenta menos popular, com menor número de downloads e uma presença discreta nas pesquisas realizadas. Isso indica que, em comparação com o Vite e o Create React App, o Parcel ainda não conquistou um amplo reconhecimento e adoção pela comunidade de desenvolvedores.

As ferramentas selecionadas para estudo foram escolhidas levando em consideração diferentes fatores. O Vite foi escolhido devido à sua crescente popularidade e notoriedade no ecossistema de desenvolvimento web. O objetivo era investigar se essa ferramenta, que tem recebido reconhecimento, realmente oferece benefícios significativos e se é uma escolha viável para os desenvolvedores.

Por sua vez, o Create React App foi selecionado com base em dois motivos principais. Primeiro, por ser uma ferramenta específica para o framework React, permitindo uma comparação com outras ferramentas mais gerais disponíveis. Segundo, por ser um inicializador de projetos que já foi lançado a mais tempo, foi investigado se essa ferramenta ainda está alinhada com as demandas e tendências atuais do mercado.

Por fim, o Parcel foi incluído no estudo para dar espaço a ferramentas menos conhecidas e explorar possíveis surpresas. O objetivo era investigar se essa alternativa menos popular poderia oferecer benefícios únicos ou soluções inovadoras em relação às outras ferramentas mais amplamente utilizadas.

A escolha dessas ferramentas foi estratégica, visando obter uma visão abrangente e comparativa das diferentes opções disponíveis no contexto de desenvolvimento web. Ao explorar ferramentas populares, específicas e menos conhecidas, espera-se obter insights valiosos sobre o estado atual do ecossistema de ferramentas de desenvolvimento web e sua aplicabilidade em projetos reais.

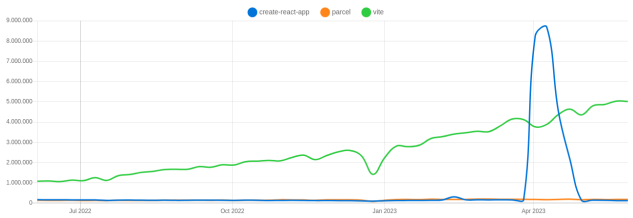

Figura 1: Comparação entre as bibliotecas analisadas segundo o npm trends

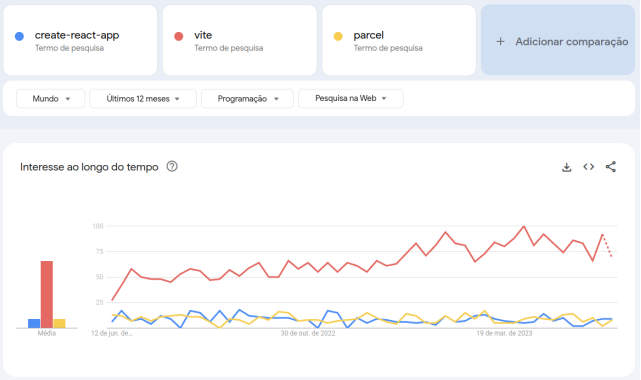

Figura 2: Comparação entre as bibliotecas analisadas segundo o google trends

3. METODOLOGIA

O presente trabalho consiste em uma análise comparativa de métricas entre diferentes bibliotecas, com o objetivo de compreender suas diferenças mais relevantes. Serão considerados aspectos como o tamanho da pasta inicial gerada e o nível de customização oferecido. Essa comparação é relevante para auxiliar os usuários na escolha de uma ferramenta mais flexível ou com arquivos iniciais menores, quando eles não têm certeza sobre qual opção selecionar.

A comparação será realizada por meio do sistema de arquivos do sistema operacional, permitindo uma análise precisa dos diferentes parâmetros. Para os comparativos de tempo, será utilizado uma estimativa baseada no número de perguntas feitas antes de cada configuração por parte da biblioteca.

O estudo será conduzido em um ambiente Linux, especificamente no Linux Mint 21.1 e com o npm versão 9.6.7 Com base nos dados coletados, serão elaborados gráficos para ilustrar de forma mais clara os pontos discutidos. Além disso, será criada uma lista de melhorias sugeridas, indicando áreas nas quais as bibliotecas podem focar para tornarem-se mais atrativas. Se possível, será aberta uma "issue" nos respectivos repositórios no GitHub, utilizando a tag de proposta, para compartilhar as sugestões encontradas.

3.1 Fluxos

A Figura 3 ilustra o fluxo de testes executado na biblioteca Vite, que apresentava mais ramificações da sua execução, levando à necessidade de um fluxograma para uma compreensão mais clara do processo. A representação visual por meio de um fluxograma tornou-se útil para evitar possíveis dúvidas e facilitar a análise do comportamento do código.

Por outro lado, as outras duas bibliotecas estudadas, Create React App e Parcel, possuíam fluxos de execução mais lineares. Portanto, não foi necessário utilizar um fluxograma para representar seus fluxos de teste. Essas bibliotecas seguiram um caminho mais direto e previsível durante a execução do código, tornando mais fácil compreender e analisar seu funcionamento sem a necessidade de uma representação visual adicional.

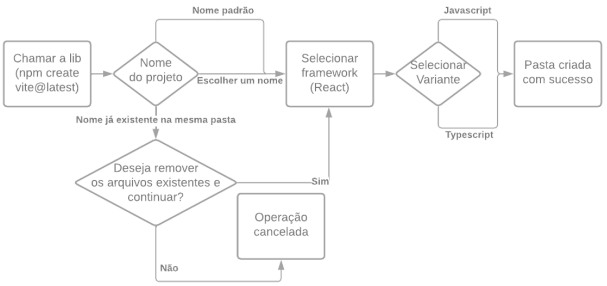

Figura 3: Processo de teste da biblioteca Vite

4. ESTUDO COMPARATIVO

Esta seção tem como objetivo avaliar o desempenho das ferramentas estudadas e realizar uma comparação entre elas. Serão apresentados gráficos comparativos e imagens ilustrativas que exemplificam o uso das ferramentas.

Para analisar a performance, foram utilizados parâmetros como tempo de inicialização do projeto, tempo de compilação, velocidade de carregamento das páginas e consumo de recursos. Foram realizados testes em diferentes cenários e tamanhos de projetos, a fim de obter uma visão abrangente do desempenho das ferramentas.

Os gráficos comparativos serão construídos com base nos dados coletados durante os testes. Eles permitirão visualizar as diferenças de desempenho entre as ferramentas em relação aos diferentes aspectos avaliados. Além disso, serão incluídas imagens que ilustram o uso das ferramentas em ação, demonstrando sua interface, recursos e funcionalidades.

Essa abordagem permitirá uma análise objetiva e fundamentada sobre a performance das ferramentas estudadas. Os gráficos e imagens fornecerão uma representação visual das diferenças e semelhanças entre as ferramentas, auxiliando na compreensão dos resultados obtidos.

É importante ressaltar que os resultados e conclusões obtidos nesta seção foram fundamentados nos testes realizados, levando em consideração as métricas e cenários estabelecidos. Essa análise comparativa contribuirá para a compreensão das vantagens e limitações de cada ferramenta, fornecendo subsídios para a seleção adequada da ferramenta mais adequada aos requisitos do projeto em questão.

4.1 Estrutura de arquivos

Nesta seção, apresentaremos a estrutura base dos arquivos gerados pela biblioteca, fornecendo uma visão geral da composição e quantidade de arquivos resultantes. Isso permitirá ter uma ideia do quão completa é a base de arquivos gerada ou se há presença de arquivos desnecessários.É importante ressaltar que a estrutura de arquivos pode variar dependendo da biblioteca específica e de suas configurações.

4.1.1 CRA

Ao utilizar o Create React App, é importante mencionar que a estrutura de diretórios gerada inicialmente pode conter alguns arquivos e pastas que podem ser considerados desnecessários, dependendo das necessidades do projeto. É comum que os

desenvolvedores acabem removendo parte desses arquivos para otimizar o projeto e reduzir o tamanho do diretório.

Um exemplo de diretório que muitas vezes é considerado desnecessário é o "node_modules". Essa pasta é criada para armazenar as dependências do projeto, ou seja, as bibliotecas e módulos externos utilizados no desenvolvimento. No entanto, em alguns casos, essa pasta pode ocupar bastante espaço em disco, principalmente quando existem muitas dependências instaladas. Isto pode ser um diferencial negativo para desenvolvedores que tem máquinas com recursos limitados.

Outro aspecto que pode ser mencionado é a presença de vários ícones e imagens relacionados ao React, que são fornecidos por padrão no Create React App. Esses arquivos incluem o logo do React e ícones utilizados em diferentes contextos. Embora sejam úteis para fins de demonstração e referência, nem sempre são necessários para um projeto em produção. Os desenvolvedores podem optar por remover esses arquivos extras para reduzir o tamanho do diretório e evitar a presença de conteúdo desnecessário, com o inconveniente de ser de forma manual.

```
node_modules
public
    |_      favicon.ico
            index.html
            logo192.png
            logo512.png
            manifest.json
            robots.txt
src
    |_      App.css
            App.js
            App.test.js
            index.css
            index.js
            logo.svg
            reportWebVitals.js
            setupTests.js
package.json
package-lock.json
README.md
```

Código 1: Árvore de arquivos gerados pelo Create React App

4.1.2 Vite

A estrutura de diretórios gerada pelo Vite é geralmente mais enxuta, contendo apenas os arquivos essenciais para a inicialização e execução da aplicação. Essa abordagem minimalista do Vite visa proporcionar um ambiente de desenvolvimento leve e eficiente, evitando a inclusão de arquivos desnecessários e focando no necessário para um desenvolvimento ágil e produtivo.

```
public
    |_ vite.svg
src
            |_ assets
                        |_ reacct.svg
            App.css
            App.jsx
            index.css
main.jsx
index.html
package.json
vite.config.js
```

Código 2: Árvore de arquivos gerados pelo Vite

4.1.3 Parcel

O Parcel requer que o usuário tenha um ponto de entrada predefinido para executar a aplicação. Por exemplo, um arquivo index.html. Ao executar o comando "npx parcel index.html", o Parcel realizará a compilação do sistema e iniciará a execução no endereço http://localhost:1234. Como resultado, será criada uma pasta chamada "dist", contendo o arquivo index.html fornecido como parâmetro, juntamente com dois arquivos adicionais: "Parcel.<hash>.js" e "Parcel.<hash>.js.map".

Esses arquivos adicionais gerados pelo Parcel são cruciais para o funcionamento correto da aplicação. O arquivo "Parcel.<hash>.js" contém o código JavaScript otimizado e empacotado pelo Parcel, enquanto o arquivo "Parcel.<hash>.js.map" é um mapeamento que permite depurar o código original durante o desenvolvimento.

Essa estrutura de arquivos gerada pelo Parcel é importante para garantir que a aplicação seja executada corretamente, facilitando o processo de desenvolvimento e fornecendo uma visão clara do código otimizado. A pasta "dist" e seus arquivos associados são prontos para serem implantados em um ambiente de produção, pois contêm o resultado final do processo de compilação realizado pelo Parcel.

```
dist
    |_ index.html
       Teste.<hash>.js
       Teste.<hash>.map
index.html
```

Código 3: Árvore de arquivos gerados pelo Parcel

4.2 Tamanho de arquivos gerados

Os gráficos abaixo ilustram o número de arquivos gerados e o espaço em disco ocupado pelas ferramentas estudadas. Na análise dos gráficos, o diretório node_modules do Create React App foi desconsiderado para facilitar a visualização, uma vez que apresentava números significativamente maiores em relação às demais variáveis.

Figura 3: Comparação entre o número de arquivos gerados pelas diferentes bibliotecas estudadas

Figura 4: Comparação entre o tamanho das pastas geradas pelas diferentes bibliotecas estudadas

	Itens gerados	Itens ocultos	Total em kB
CRA com node_modules	41.777	507	261.200
CRA sem node_modules	17	0	713,6
Vite	12	2	22,7
Parcel	3	0	49,7

Tabela 1: Comparação entre o número de arquivos gerados pelas diferentes bibliotecas estudadas

4.3 Tempo de configuração

Para avaliar a eficiência na inicialização de um projeto, é importante considerar o tempo necessário para que o projeto esteja em execução. No entanto, medir o tempo exato pode variar muito e até mesmo uma média não representaria com precisão o processo, uma vez que fatores como a velocidade de digitação do usuário e a escolha do nome do projeto podem influenciar.

Nesta seção, a análise será baseada na quantidade de perguntas feitas pelas bibliotecas durante a criação do projeto e no número de comandos executados no terminal. Essas métricas são utilizadas como um indicador da complexidade e rapidez no processo de inicialização.

No caso do Vite, o processo de inicialização do projeto envolve quatro perguntas básicas e cinco comandos no terminal, seguindo a abordagem do React. Essas perguntas incluem o nome do projeto a ser criado, o framework desejado e sua variante. Após fornecer as respostas, o usuário precisa apenas navegar para a pasta do projeto, executar o comando "npm install" para instalar as dependências necessárias e, em seguida, executar o comando para iniciar o projeto.

Por outro lado, ao criar um projeto com o create-react-app, o usuário pode especificar o nome do projeto diretamente como parâmetro no comando de criação. O processo de criação do projeto é um pouco mais demorado devido ao download das dependências necessárias ser maior. Após a conclusão, o usuário só precisa entrar no diretório do projeto e executar os comandos, totalizando três comandos para iniciar o desenvolvimento.

No caso do Parcel, a compilação direta é rápida, porém, é necessário fornecer um arquivo existente para ser processado, sendo utilizado apenas um comando. Após ter o arquivo, o comando em si é rápido e eficiente.

É importante destacar que, embora o tempo de inicialização possa variar dependendo de diversos fatores, como a velocidade da conexão com a internet e especificações do computador, tanto o Vite quanto o Parcel têm a vantagem de oferecer uma inicialização mais ágil em comparação com o Create React App. Isso se deve em parte ao pré-processamento de dependências do Vite e à abordagem mais enxuta do Parcel em relação ao número de perguntas e configurações necessárias.

4.4 Nível de customização

O Vite se destaca em termos de customização, oferecendo uma ampla variedade de opções ao iniciar um projeto. Isso inclui a possibilidade de escolher entre diferentes frameworks, bem como opções adicionais dentro de cada framework. Essa flexibilidade permite aos desenvolvedores adaptarem o projeto às suas necessidades específicas desde o início. Por exemplo, é possível selecionar um framework como Vue.js, React ou Svelte, e ainda ter a liberdade de escolher opções mais específicas dentro desses frameworks.

Por outro lado, o Parcel não possui um foco específico em um único framework e é agnóstico em relação à linguagem de programação. Isso significa que os desenvolvedores têm uma certa liberdade para escolher a linguagem que desejam utilizar. No entanto, essa customização precisa ser feita manualmente, pois o Parcel não oferece uma orientação direta nesse sentido.

Já o Create React App é exclusivamente voltado para o React, como sugere o nome. Ele foi incluído na pesquisa de forma intencional para avaliar as vantagens que uma biblioteca específica pode oferecer em comparação com opções mais gerais. No entanto, em termos de customização, o Create React App tem limitações, pois sua proposta é fornecer uma configuração

pré-definida e simplificada para projetos baseados no React, sem oferecer muitas opções adicionais.

5. TRABALHOS RELACIONADOS

Durante a pesquisa de literatura relacionada a este trabalho, foi constatado que existem poucos estudos semelhantes disponíveis. Os resultados encontrados se limitaram a textos que comparam os frameworks mais populares atualmente utilizados no desenvolvimento web, que foram os trabalhos *React.js vs. Next.js by Zerihun Dinku* [6] e *Comparative Analysis on Front-End Frameworks for Web Applications by Rishi Vyas* [7] . No entanto, esses estudos foram úteis para orientar a estrutura e as seções deste trabalho, bem como fornecer parâmetros de comparação e métricas relevantes a serem consideradas neste estudo.

Embora a pesquisa na área específica deste trabalho seja limitada, as referências consultadas no contexto mais amplo dos frameworks web forneceram uma base sólida para a análise e comparação das bibliotecas em questão. Essas literaturas auxiliaram na definição das abordagens metodológicas, bem como na identificação dos critérios de avaliação apropriados para medir e comparar as diferentes bibliotecas.

É importante ressaltar que a falta de estudos similares destacou a necessidade de realizar esta pesquisa, preenchendo uma lacuna na literatura e contribuindo para o conhecimento nessa área específica. Portanto, este trabalho busca fornecer insights valiosos e contribuições significativas para o campo do desenvolvimento web e a seleção de bibliotecas adequadas para projetos nessa área.

6. CONCLUSÕES

	Tamanho dos arquivos gerados	**Tempo de configuração**	**Nível de customização**
Create React App (CRA)	Gerou uma pasta consideravelmente maior que seus concorrêntes	Muito rápido já que é uma biblioteca focada em apenas um framework específico	Praticamente inexistente, pelos mesmos motivos citados no tempo de configuração
Vite	Apesar de ter gerados mais arquivos iniciais do que o Parcel, teve um tamanho menor mostrando sua eficiência e otimização em performance	Das bibliotecas estudadas foi a mais demorada por trazar mais opções de customização, então dependendo do contexto pode ser algo positivo ou negativo	Nível de customização bem alto, várias opções diferentes de frameworks que se pode trabalhar, e configurações diferentes dependendo do framework
Parcel	Um pouco superior ao Vite mas nada compromet ed or, ainda assim ocupa pouco espaço em disco inicialmente	Bem rápido, similar ao create-react-app	Pode ser utilizado com qualquer linguagem, mas não tem um passo a passo para as configuraçõe s como no Vite

Tabela 2: Tabele das conclusões finais do trabalho

Com base nas análises realizadas, é possível concluir que atualmente o Vite se destaca como a ferramenta líder entre as opções estudadas. Ele oferece uma ampla gama de funcionalidades, opções de personalização e apresenta um desempenho excepcional, que dificilmente apresenta falhas ou deficiências.

Por outro lado, o Create React App, embora seja uma ferramenta exclusiva para o React e diferenciada das outras duas em termos de contexto de uso, vem perdendo popularidade nos últimos tempos, em comparação com as novas ferramentas mais abrangentes e otimizadas disponíveis. Embora ainda seja uma opção válida, sua adoção tem diminuído gradualmente.

O Parcel, por sua vez, surge como uma agradável surpresa e merece maior reconhecimento por parte da comunidade de desenvolvedores. É uma excelente escolha para aqueles que ainda têm dúvidas sobre as configurações básicas ao iniciar um projeto, como a escolha entre JavaScript ou TypeScript. O Parcel oferece um conjunto sólido de recursos e é uma alternativa viável para desenvolvedores em busca de uma ferramenta eficiente e fácil de usar.

Quanto a trabalhos futuros, sugere-se a realização de uma análise mais aprofundada com ferramentas menos conhecidas pela comunidade. Isso permitiria explorar os prós e contras dessas ferramentas e descobrir "pérolas" ocultas que possam trazer mais visibilidade e reconhecimento para soluções que são eficientes, mas muitas vezes não recebem a devida atenção. Dessa forma, seria possível expandir as opções disponíveis e promover maior diversidade e inovação no ecossistema de ferramentas para desenvolvimento web, assim como sugerir melhorias diretamente para as bibliotecas estudadas, entrando em contato através de e-amails ou por meio do sistema de issues no Github.

7. EXPERIÊNCIA

Realizei um trabalho extremamente fascinante, uma vez que possuo um interesse na área de desenvolvimento frontend para web. Durante esse período, tive a oportunidade de conhecer novas ferramentas, como o Parcel, que me cativou imensamente, mesmo não sendo tão popular em comparação com outras opções. A partir dessa experiência, despertou-se em mim o desejo de explorar bibliotecas menos conhecidas em outros temas, que certamente possuem um enorme potencial, mas ainda não desfrutam de grande popularidade.

Já estava familiarizado com o Create React App há algum tempo, e foi interessante observar sua evolução até o estágio atual. Quanto ao Vite, eu já tinha conhecimento dessa ferramenta por sua crescente popularidade nos últimos tempos.

No entanto, um dos principais desafios foi encontrar materiais mais abrangentes que explicam de maneira menos casual o funcionamento interno dessas ferramentas, uma vez que as documentações tendem a focar mais nas funcionalidades oferecidas.

8. AGRADECIMENTOS

Primeiramente,, gostaria de agradecer a todos os professores do curso de Ciência da Computação da Universidade Federal de Campina Grande, por todo o conhecimento transmitido.

A minha mãe, Daniela Barros Nascimento, que sempre me guiou com bons conselhos e me deu todo suporte necessário para que eu podesse chegar a esse momento. Meu irmão, Guilherme Nascimento Oliveira, que posso contar sempre com ele.

Também agradeço minha namorada, Palloma Thalyta Araújo Reges, que me apoiou nos momentos mais difíceis durante a graduação e me fez acreditar que eu conseguia.

E aos meus grandes amigos que fiz ao longo do curso e agora levo para a vida como amigos de profissão também, Edson Wesley, Igor Franca, Lucas Abrantes, Gutemberg Filho, Jerônimo Jairo.

9. REFERÊNCIAS

[1] **Announcing Vite 4**. Disponível em: <https://vitejs.dev/blog/announcing-vite4.html>. Acesso em: 14 dez. 2022.

[2] **Procura por profissionais de tecnologia cresce 671% durante a pandemia**. Disponível em: <https://www.cnnbrasil.com.br/business/procura-por-profissionais-de-tecnologia-cresce-671-durante-a-pandemia/>.

[3] **@angular/core vs angular vs ember-source vs react vs svelte vs vue | npm trends**. Disponível em: <https://npmtrends.com/@angular/core-vs-angular-vs-ember-source-vs-react-vs-svelte-vs-vue>. Acesso em: 14 dez. 2022.

[4] **Mozilla on how package managers works** https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Understanding_client-side_tools/Package_management

[5] **NPM Trends** - https://npmtrends.com/npm-vs-pnpm-vs-yarn

[6] **React.js vs. Next.js - Zerihun Dinku** https://www.theseus.fi/bitstream/handle/10024/750122/Dinku_Zerihun.pdf?sequence=2

[7] **Comparative Analysis on Front-End Frameworks for Web Applications by Rishi Vyas** - https://d1wqtxts1xzle7.cloudfront.net/89814198/Comparative_Analysis_on_Front_End_Frameworks_for_Web_Applications-libre.pdf?1660741150=&response-content-disposition=inline%3B+filename%3DComparative_Analysis_on_Front_End_Framew.pdf&Expires=1686036799&Signature=EolH9UE49A0SeSQJWYXrQ74ONLVUtclC4gZraH3jPVsiP5QoJY3SlEFRuU9pXf7XOapySxUnu9vdp9d34SaICX2NHnPmkgevvxSjqPEW2S0lMPLXaItzI541QyYYxuIeotATZApmNWqk3jLM9-wV0ebVwxGYXShZmdOhK46t0eoK7UQZJwUaqsWq2Agpurer7ZBoOFYePsvZ6KwGDh8Jxyf9O3AgxyTLjvo7C91gA9vO9jOzJHEQ4oShQvxUZoDWDbz7HhhuP1wX4dGgZii4D84jsG4Mq7bIJT2mEoyLWGky48oLrz64A3JKdqFGhpzf30-K4dKON~zEWAjB8m0inA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[8] **Vite Documentation** - https://v3.vitejs.dev/guide/why.html

[9] **Parcel Documentation** - https://parceljs.org/

Sobre o autor:

Gabriel Nascimento Santos é graduando em Ciência da Computação na Universidade Federal de Campina Grande (UFCG) e atua como desenvolvedor de software frontend.

Dalton Dario Serey Guerrero. Professor.na UFCG. Orientador.