

**UNIFACS UNIVERSIDADE DE SALVADOR**

**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**DIEGO SILVA - 1272221315**

**MATEUS CATUREBA - 12722133037**

**MATHEUS LOBO - 1272221392**

**RAFAEL BASTOS - 1272225551**

**SISTEMAS DISTRIBUÍDOS E MOBILE**

**SALVADOR, DEZEMBRO DE 2023**

1. **Requerimentos de software:**

O software foi desenvolvido utilizando a linguagem Javascript, tanto no front-end quanto no back-end, através dos frameworks React (front-end) e Node.js com Express ( back-end). O banco de dados escolhido foi o PostgreSQL e para a comunicação foi usado uma API Rest. Para executar a aplicação, são necessários os seguintes softwares:

### **Para o Desenvolvimento do Back-End (Node.js):**

* Node.js:
  + Ambiente de execução para JavaScript no lado do servidor.
* npm (Node Package Manager):
  + Gerenciador de pacotes para instalação de bibliotecas e dependências.
  + Incluído com o Node.js.

### **Para o Desenvolvimento do Front-End (React):**

* Node.js:
  + Necessário para o desenvolvimento React, pois o npm é incluído.
* npm (Node Package Manager):
  + Gerenciador de pacotes para instalação de bibliotecas e dependências.
  + Incluído com o Node.js.
* Create React App (opcional):
  + Uma ferramenta para criar rapidamente projetos React pré configurados.
  + Instalação: npm install -g create-react-app

### **Banco de Dados :**

* Banco de Dados Relacional ( PostgreSQL)

### **Ferramentas Adicionais:**

* Editor de Código:
  + Visual Studio Code
* Git:
  + Sistema de controle de versão para rastreamento de alterações no código fonte.

Para instalar as dependências do Node.js e React, é necessário executar o seguinte comando nos respectivos diretórios do projeto:

* npm install

1. **Justificativa da escolha da tecnologia**

A escolha do Javascript e seus frameworks para a elaboração do projeto se deu pelos seguintes fatores:

* Coerência: Usar JavaScript tanto no lado do cliente (navegador) quanto no lado do servidor (Node.js) facilita a coesão e a compreensão do código.
* Assincronicidade: Node.js é conhecido por sua arquitetura orientada a eventos, o que permite que ele lide eficientemente com um grande número de conexões simultâneas. Isso é particularmente útil para aplicativos em tempo real, como chat e jogos online.
* Desempenho: O modelo de E/S não bloqueante do Node.js pode resultar em um desempenho aprimorado, pois permite que o servidor continue a lidar com outras solicitações enquanto espera operações de entrada/saída.
* Roteamento: O Express simplifica a definição de rotas para diferentes URLs e métodos HTTP.
* Middleware: O uso extensivo de middleware permite adicionar funcionalidades ao aplicativo de forma modular e reutilizável.
* O Express é conhecido por sua simplicidade e flexibilidade, permitindo que os desenvolvedores criem rapidamente APIs RESTful e aplicativos da web.
* Módulos e Pacotes: O Node Package Manager (NPM) oferece um vasto ecossistema de módulos e pacotes, permitindo que você aproveite bibliotecas existentes para acelerar o desenvolvimento.
* Facilidade de Construção: O Express facilita a construção de APIs RESTful devido à sua estrutura simplificada e orientada a middleware. Rotas e manipuladores podem ser configurados de maneira intuitiva.
* Suporte a Diversos Bancos de Dados: Node.js, em conjunto com bibliotecas como Mongoose para MongoDB ou Sequelize para bancos de dados relacionais, permite uma integração eficiente com diversos tipos de bancos de dados.
* React segue um modelo de desenvolvimento baseado em componentes. Isso significa que a interface do usuário é dividida em pequenos componentes reutilizáveis, tornando o código mais modular, fácil de entender e manter.
* React utiliza um modelo de programação reativa, o que significa que as alterações no estado do aplicativo são automaticamente refletidas na interface do usuário. Isso simplifica o gerenciamento do estado e torna as atualizações de UI mais eficientes.
* React usa um Virtual DOM para otimizar as atualizações na interface do usuário. Ele compara o estado atual do DOM com uma representação virtual em memória antes de efetuar as alterações no DOM real. Isso reduz o número de manipulações no DOM e melhora o desempenho.
* JSX é uma extensão de sintaxe que permite escrever marcação HTML dentro do JavaScript. Isso torna a criação de componentes mais intuitiva e facilita a visualização da estrutura da UI diretamente no código.
* O fluxo de dados em React é unidirecional, o que significa que os dados fluem em uma única direção, tornando o rastreamento de bugs e o entendimento do código mais fácil.
* React possui uma comunidade ativa de desenvolvedores e uma grande quantidade de bibliotecas e ferramentas de suporte, facilitando a resolução de problemas comuns e o desenvolvimento rápido.
* React pode ser facilmente integrado a outras bibliotecas e frameworks, permitindo que os desenvolvedores escolham as ferramentas certas para suas necessidades específicas.
* React suporta renderização no lado do servidor (Server-Side Rendering - SSR) e renderização no lado do cliente (Client-Side Rendering - CSR), tornando as aplicações React mais amigáveis para mecanismos de busca.
* React possui extensões para navegadores (React DevTools) que facilitam a depuração e o rastreamento do estado da aplicação.

Em resumo, o uso de React e Node.js apresentam diversas vantagens para o desenvolvimento de aplicações web modernas, proporcionando eficiência, desempenho e uma melhor experiência do usuário.

1. **Instruções para instalação e execução da aplicação.**

**Instalação do banco de dados:**

1. Baixar o PostgreSQL no site oficial: <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
2. Efetuar o download da versão desejada.
3. Executar o arquivo de instalação.
4. Seguir as instruções de instalação.
5. No campo senha, use: **62442.**
6. Caso seja solicitado um login, use **postgres**.
7. Finalizar a instalação.
8. No menu **Iniciar,**  pesquisar por **pgadmin** e executar.

**Instalação do VS Code:**

1. Baixar o VS Code no site oficial: <https://code.visualstudio.com/download>
2. Efetuar o download da versão desejada.
3. Executar o arquivo de instalação.
4. Seguir as instruções de instalação.
5. Finalizar a instalação.

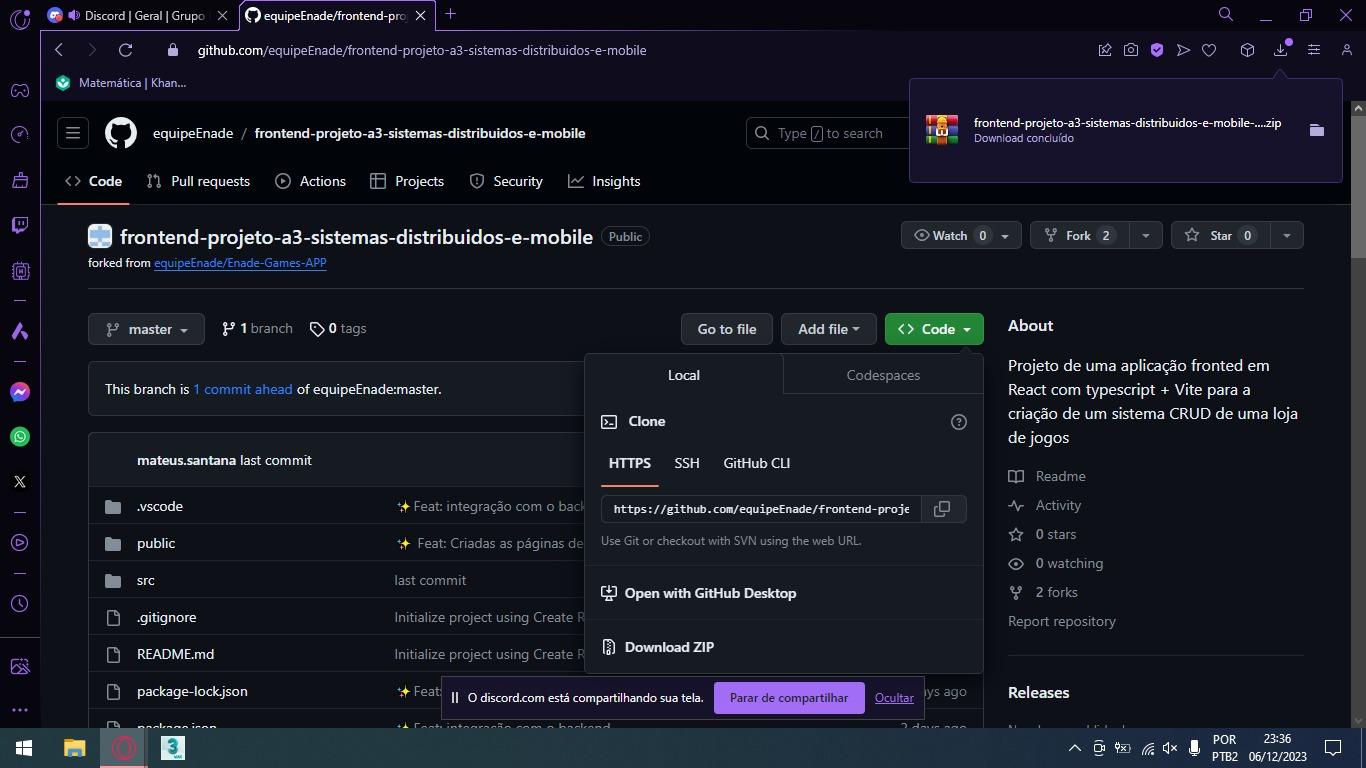
**Baixando arquivos do repositório do Github:**

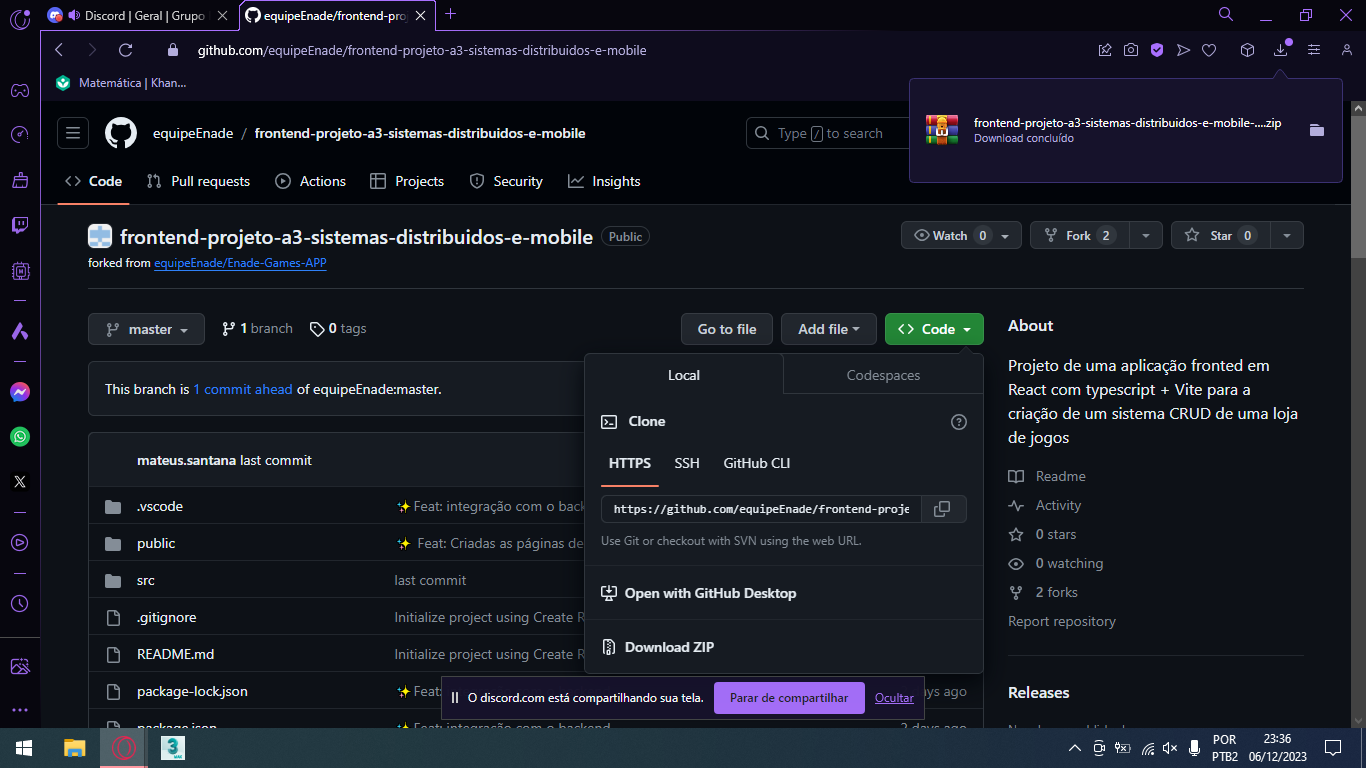
1. Acessar o repositório do projeto no Github através dos links:

**Back-end:** <https://github.com/equipeEnade/backend-projeto-a3-sistemas-distribuidos-e-mobile>

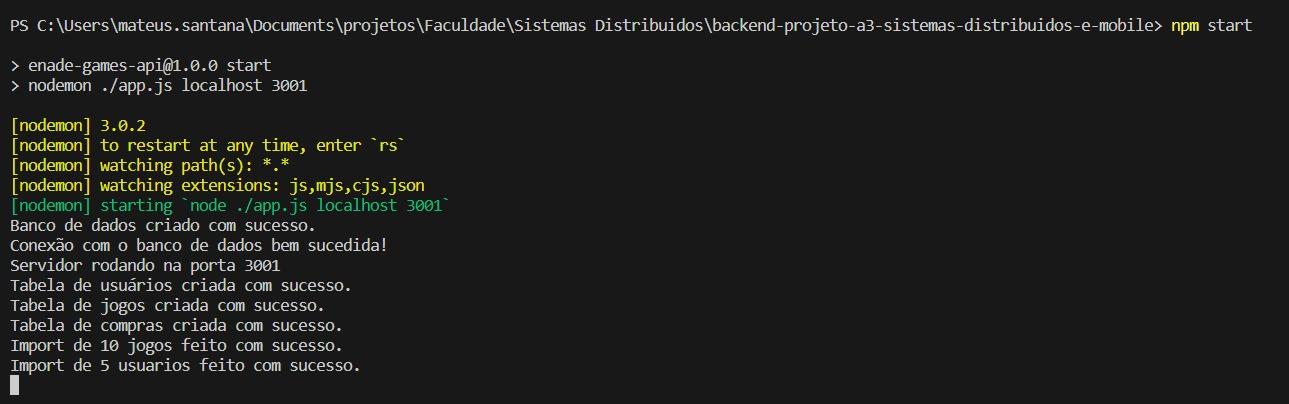
**Front-end:** <https://github.com/equipeEnade/frontend-projeto-a3-sistemas-distribuidos-e-mobile>

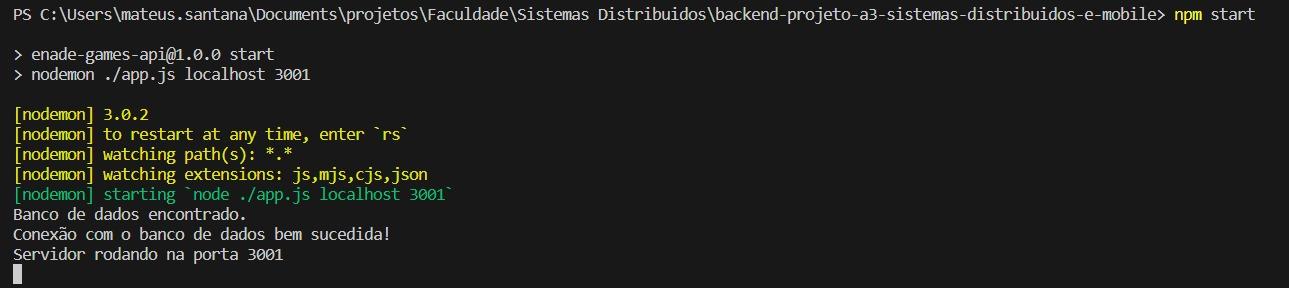
1. Criar uma pasta para receber os arquivos que serão baixados a seguir.
2. Proceder conforme o descrito para os dois repositórios:
3. Clicar no botão **CODE** e baixar como zip.





1. Extrair os arquivos na pasta criada no item b.
2. Abrir o VisualStudioCode
3. Já no VSCode File > Open Folder> selecione a pasta extraída no item anterior, repetir o mesmo processo para as duas pastas do repositório.
4. Após, acessar o terminal do VSCode (Ctrl + ” ).
5. Dentro do Terminal insira a instrução “ npm install”.



1. A seguir insira a instrução “npm start”
2. Seguir o mesmo procedimento para as duas pastas.
3. Ao concluir o processo do item ‘j’ seu navegador abrirá e a aplicação estará rodando.

**4. Apresentação e detalhamento da aplicação**

**Visão Geral da Arquitetura:**

1. Camada de Interface de Usuário (UI):

**Módulos:**

Página inicial

Página de Login

Página de Cadastro

Lista de Produtos

Detalhe do Produtos

Formulário de criação/atualização de Produtos

**2. Camada de Lógica de Aplicação (Backend):**

**Módulos:**

**Gestão de Produtos:**

Responsável por criar, atualizar, atribuir e excluir tarefas.

Interface com o banco de dados para armazenamento/retrieval de dados.

**Gestão de Usuários:**

Autenticação e autorização de usuários.

Permissões para diferentes papéis (administrador e usuário).

**Comunicação entre Módulos:**

Utiliza APIs REST para comunicação entre o frontend e o backend.

**3. Camada de Dados:**

**Banco de Dados:**

Armazena informações sobre produtos , usuários, e relacionamentos.

Utiliza o banco de dados PostgreSQL

**4. Camada de Segurança:**

**Autenticação:**

Utiliza requisições HTTP..

**Autorização:**

Baseada em funções, controla o acesso de usuários a diferentes partes do sistema.

**5. Camada de Log:**

**Registros de Auditoria:**

Registra eventos importantes, como venda de produtos, atualizações, e autenticação.

**6. Camada de Infraestrutura:**

**Servidores:**

Possibilidade de implantação em nuvem (por exemplo, AWS, Azure) para escalabilidade.

**Fluxo de Execução:**

O usuário acessa a Página Inicial.

O cliente faz uma requisição HTTP com o método get para o back-end, solicitando todos os Produtos

O back-end recebe essa solicitação na sua controller e redireciona para o devido método da service

A service redireciona para repository, que é a camada da aplicação back-end que tem acesso ao banco de dados, que faz a requisição para o banco de dados via query SQL, retorna para service, que depois retorna para controller uma lista com todos os produtos, e a controller retorna para o Cliente.

E assim acontece com todas as demais requisições.