

**DOCUMENTAÇÃO SOBRE O DESENVOLVIMENTO DO PROJETO
INTEGRADOR II**

1-Considerações iniciais

Com a crescente necessidade de criar sistemas informatizados para atender e gerenciar pessoas no contexto de clínicas de saúde e hospitais privados e públicos surgiu a ideia de satisfazer essa necessidade de uma maneira moderna e acessível às pessoas.

O grande problema acontece quando há uma enorme demora para agendar exames e consultas. Em quase sua totalidade de marcações são feitas de maneira presencial e o recebimento de laudos também.

A solução encontrada e trabalhada foi um "website", que visasse facilitar o agendamento, gerenciamento e recebimentos de exames (especificamente do exame de eletrocardiograma), um outro motivo que podemos citar são: o fácil acesso à plataforma para a população, manutenções mais sofisticadas e tecnologias bastantes utilizadas no mercado de trabalho, o que já facilita aos prestadores de serviço e a entidade que tem o sistema.

Inicialmente o projeto tem cunho acadêmico, mas fortes indícios de uma potencial solução, gerando acessibilidade, rapidez e administração pública ou privada, visto a grande deficiência das mesmas referente a inovação e administração do sistema de saúde implementado.

Os membros do referido projeto são: **ADALBERTO FELIPE PINHEIRO CHAVES, DIEFESSON DE SOUSA SILVA, LUIS FELIPE DOMINGOS VIEIRA TORRES, VICTOR GABRIEL MARTINS DE OLIVEIRA LOIOLA**. Todos alunos de 4º semestre do curso de ciência da computação na Universidade Federal do Ceará. O Professor da instituição, Marciel Barros, foi o orientador do projeto.

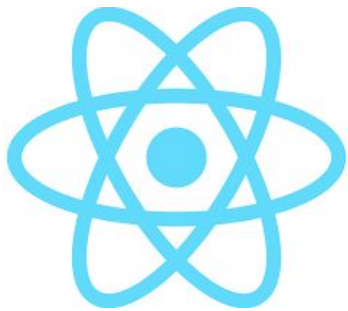
O período que decorreu este projeto foi da pandemia do covid-19, especificamente entre os anos de 2020 e 2021.

Nos próximos tópicos encontram-se detalhes técnicos e gerais de como o projeto foi implementado.

2-Tecnologias utilizadas

Responsável pela parte visual e de experiência do usuário.

O QUE É O REACT?



React é uma biblioteca Javascript para construção de interfaces de usuário (UI). Esta biblioteca surgiu em 2011, sendo utilizada pelo Facebook e, posteriormente, pelo Instagram. Em 2013, o código foi aberto para a comunidade, crescendo, assim, a sua popularidade.

Diferente de projetos em HTML, em que, criamos uma página em um único arquivo, o React faz diferente: ele faz uso do conceito de componentes.

Com o uso de componentes, o React pode separar o código em diferentes partes, ou seja, em arquivos diferentes (o que não poderia ser feito com o HTML). Cada uma destas partes se comportam como componentes reutilizáveis.

Por exemplo, em uma página comum da internet podemos ter um menu superior e um painel superior. Estes componentes, como o menu superior, poderiam ser reutilizados novamente em qualquer outra página do site.

COMO INSTALAR E USAR O REACT?

A seguir será descrito o passo-a-passo de instalar o React no Windows.

0.1: Antes de começar a instalação do React, é necessário verificar se você tem o Node instalado na sua máquina. Para saber isso, basta abrir o seu Prompt de Comando e digitar o comando a seguir:

```
Prompt de Comando
Microsoft Windows [versão 10.0.19042.804]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\lfeli>node --version
v14.15.4

C:\Users\lfeli>
```

Figura 1: Prompt de Comando do Windows

Caso apareça uma mensagem parecida com a acima, você já tem o Node instalado no seu computador e pode prosseguir. Caso contrário, você pode instalar o Node pelo link a seguir: [Node.js](https://nodejs.org/). É recomendado que você instale a versão LTS.

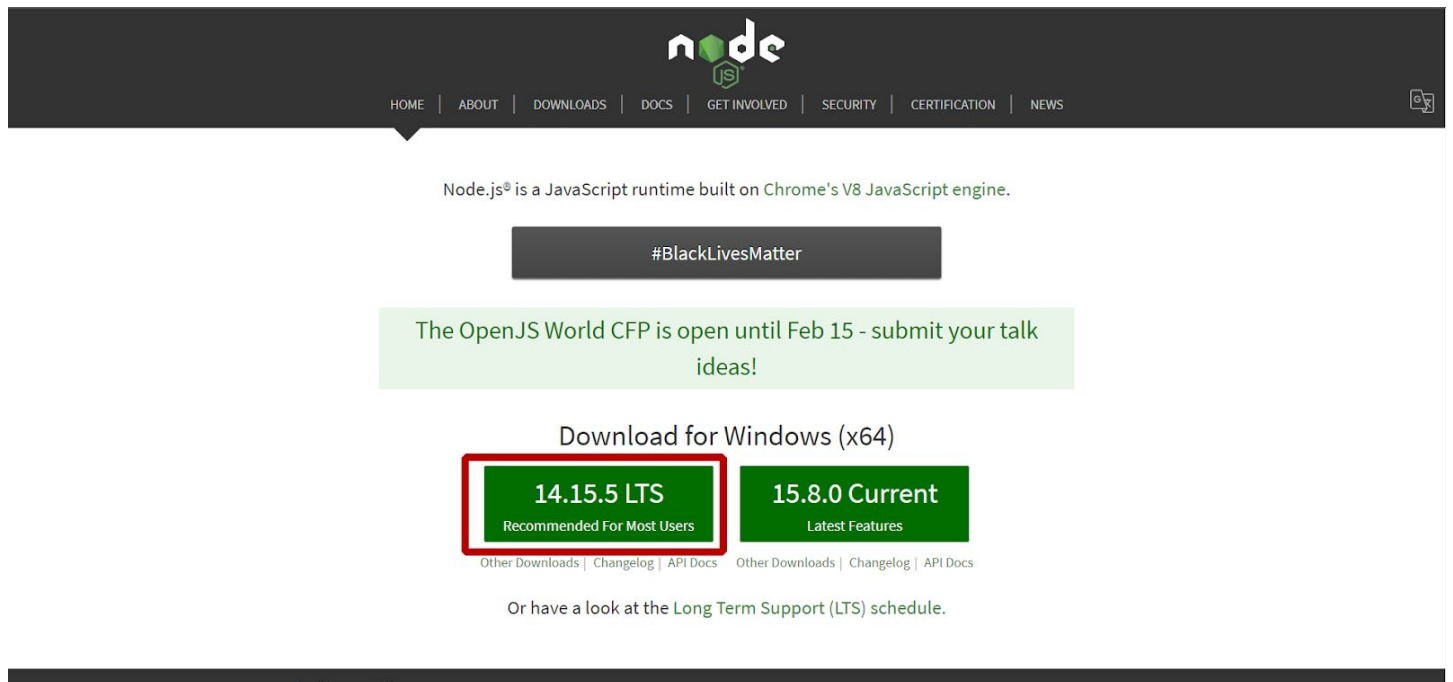


Figura 2: Página inicial do Node

1: Com tudo previamente configurado, agora temos de acessar um diretório do nosso computador para criar o projeto. Neste exemplo, criamos um projeto na pasta Documentos:

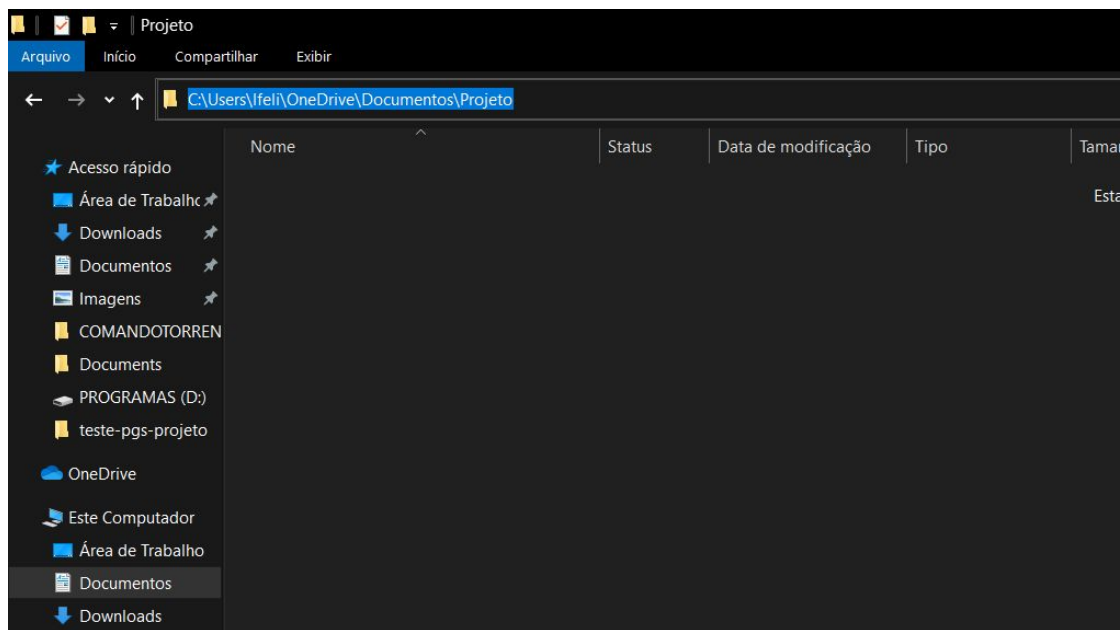


Figura 3: Diretório da pasta do projeto

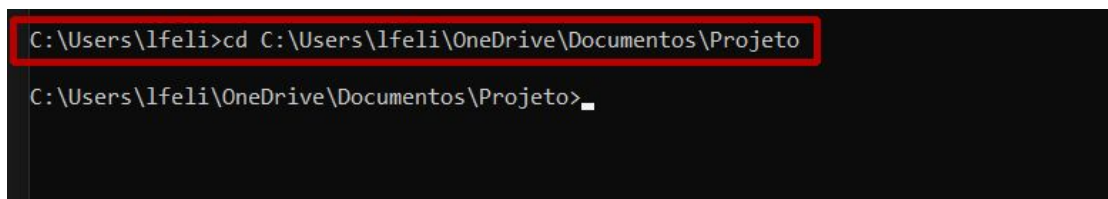


Figura 4: Acessando o diretório pelo Prompt de Comando

2: Agora, com o diretório acessado, usamos o seguinte comando `npx create-react-app nome-do-projeto` para fazer a instalação do projeto:

```
C:\Users\lfeli\OneDrive\Documentos\Projeto>npx create-react-app projeto-teste
```

Figura 5: Comando para instalar o projeto

2.1: Se tiver tudo ocorrido bem, aparecerá a mensagem a seguir:

```
npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back
  to using create-react-app.

We suggest that you begin by typing:

  cd projeto-teste
  npm start

Happy hacking!

C:\Users\lfeli\OneDrive\Documentos\Projeto>
```

Figura 6: Mensagem de conclusão de instalação

3. Pronto! Já temos um projeto em React criado. Agora, você pode acessar a pasta do projeto e executar o comando `npm start` para visualizá-lo.

```
C:\Users\lfeli\OneDrive\Documentos\Projeto>cd projeto-teste  
C:\Users\lfeli\OneDrive\Documentos\Projeto\projeto-teste>npm start
```

Figura 7: Comandos para fazer a visualização do projeto.

Será aberta uma nova aba no seu navegador, exibindo o estado atual do seu projeto, na porta 3000.

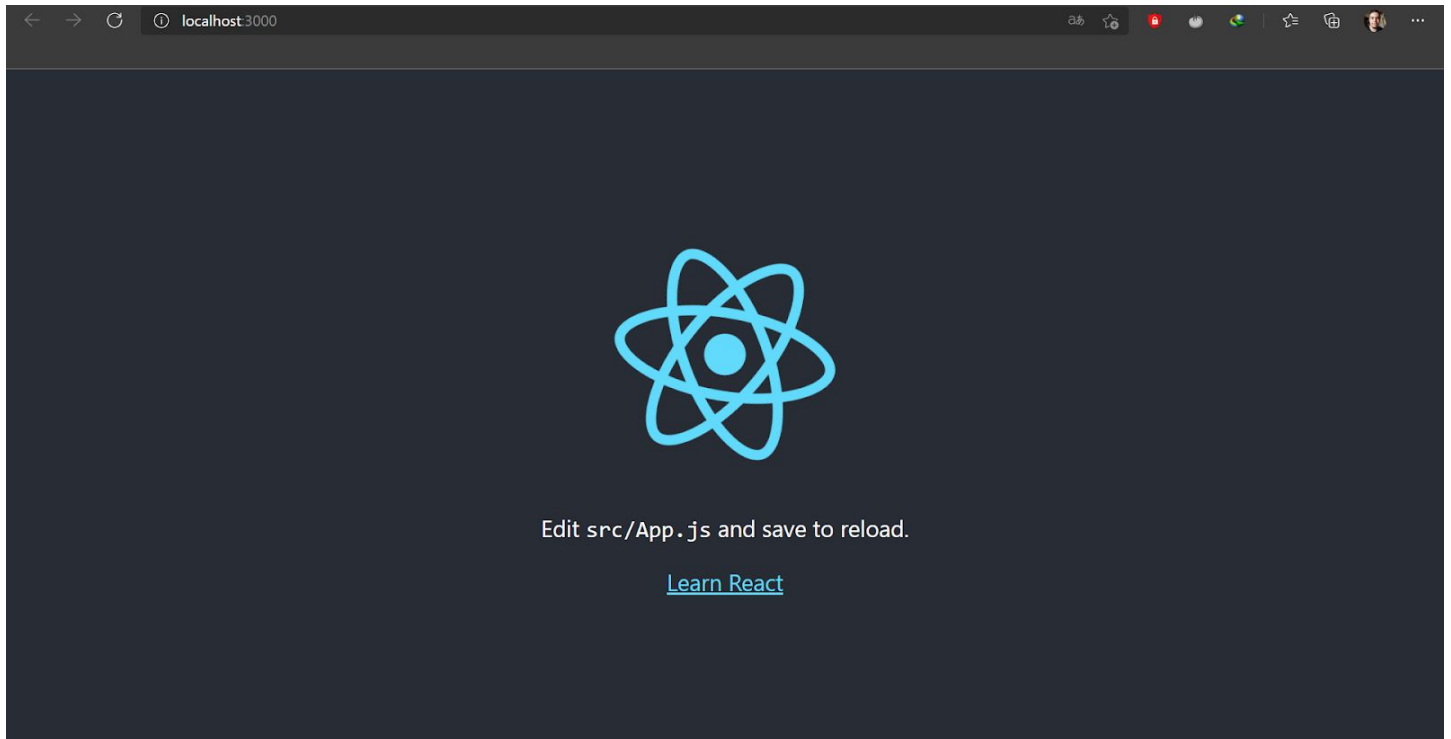


Figura 8: Página do projeto no navegador

Você também pode acessar a pasta com os arquivos do projeto, fazendo o uso de um editor como o Visual Studio Code.

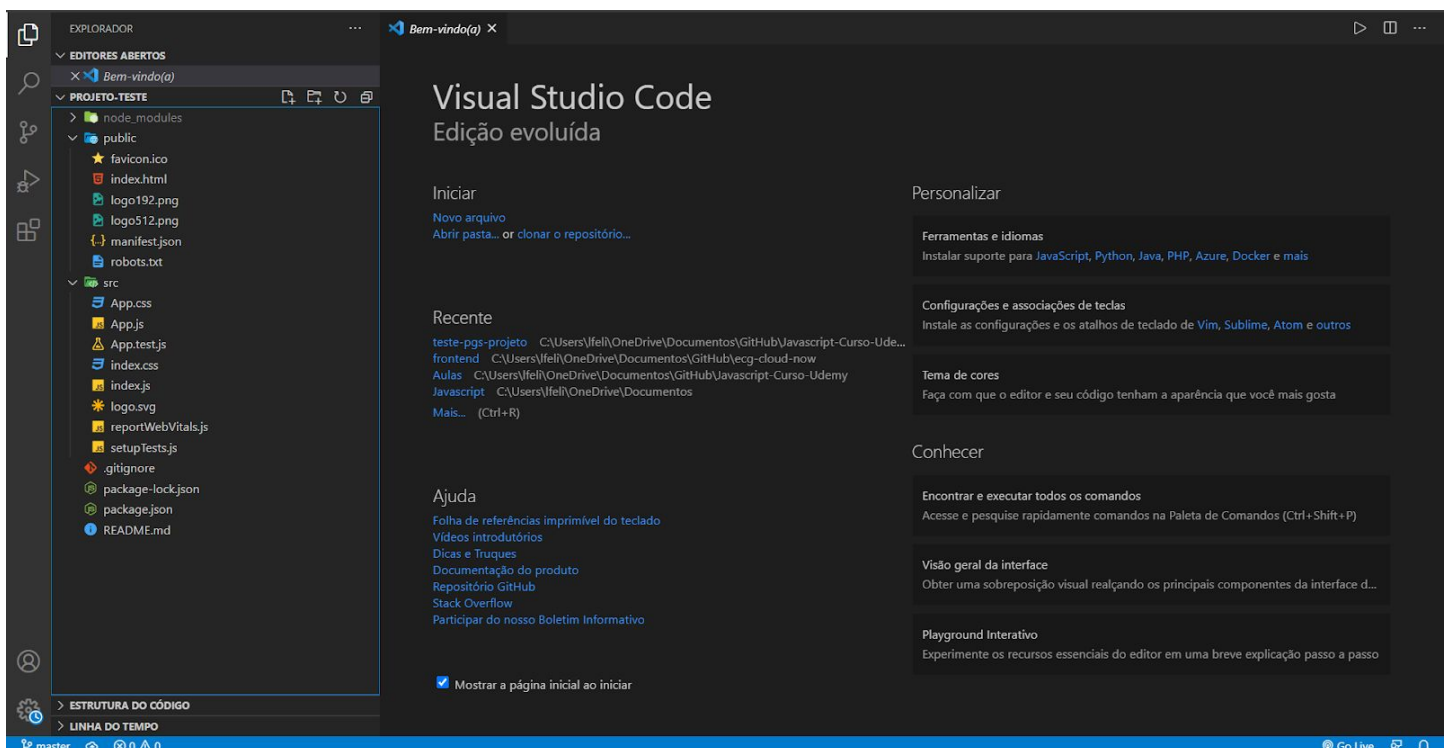


Figura 9: Visualização dos arquivos do projeto pelo Visual Studio Code

COMO O REACT FOI UTILIZADO EM NOSSO PROJETO?

Primeiramente, para iniciarmos o nosso projeto, excluímos todos os arquivos desnecessários, tanto da pasta *public* quanto da pasta *src*. Desse modo, ficamos apenas com os seguintes arquivos:

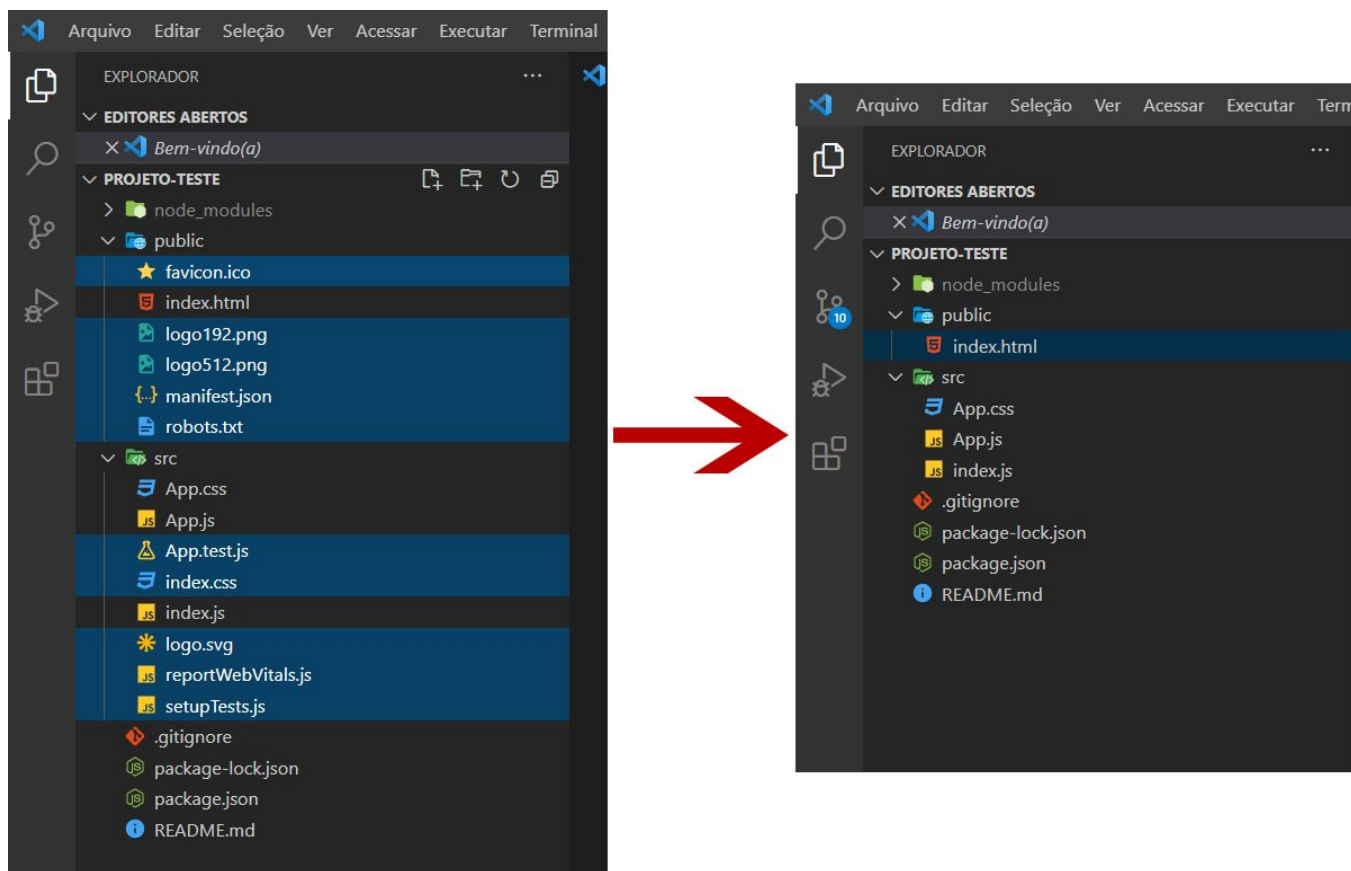


Figura 10: Projeto antes e depois da remoção de arquivos

Agora estamos, pela primeira vez, vendo um código em React. No arquivo “index.js”, removeremos algumas importações de bibliotecas desnecessárias e deixaremos que o arquivo fique responsável apenas por renderizar o que houver no arquivo “App.js”. Perceba que aqui, introduzimos o conceito de componentes, pois “App.js” será um componente, ou seja, um elemento da página “index.js”.

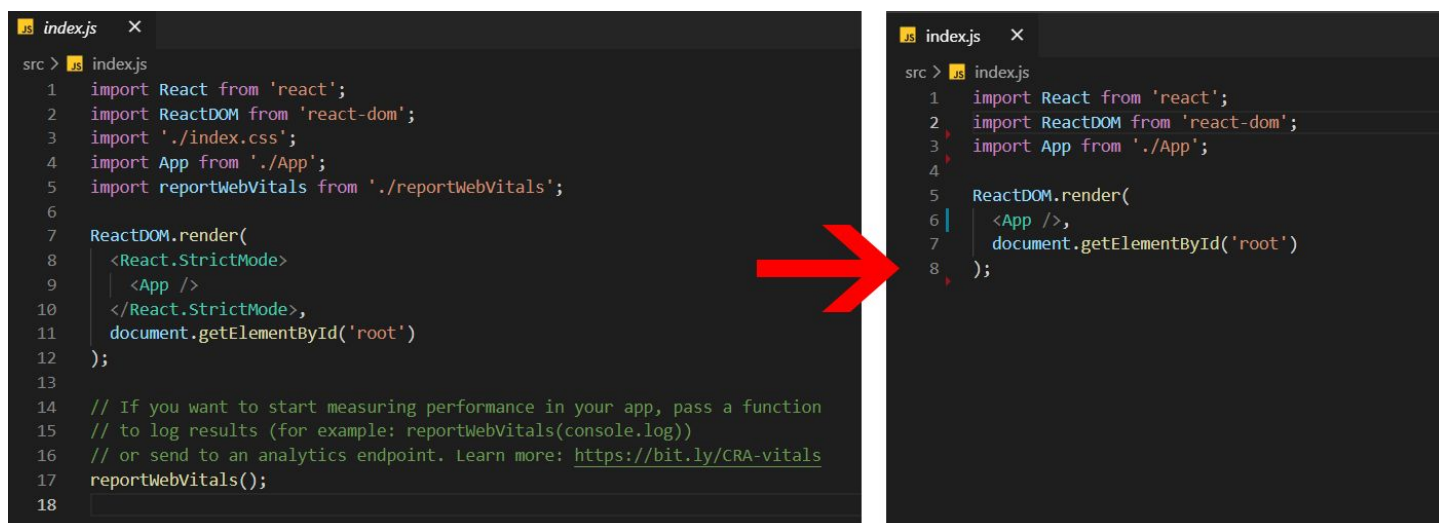


Figura 11: Antes e depois de configurar o arquivo “Index.js”.

Depois, partiremos para o arquivo “App.js” (aquele que nós acabamos de importar no “index.js”). Nele, importaremos apenas a biblioteca do React e o faremos retornar uma *arrow function*, que facilitará o nosso trabalho, caso queiramos exportar a função para outros arquivos.

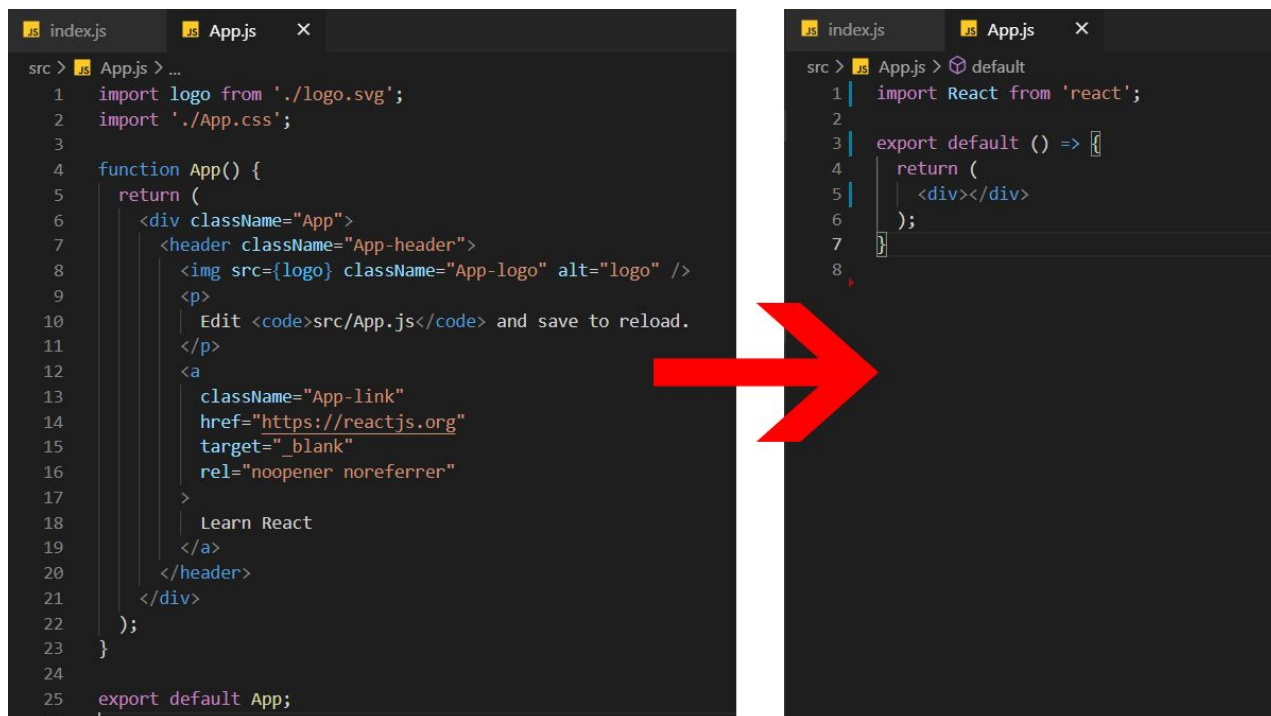


Figura 12: Antes e depois de configurar o arquivo “App.js”.

Por fim, no arquivo “index.html”, podemos apagar todos os comentários e links para outras imagens, deixando apenas o que for necessário, igualmente ao mostrado abaixo. Também podemos alterar a linguagem do nosso arquivo para “pt-BR” e trocar o nome do nosso projeto.

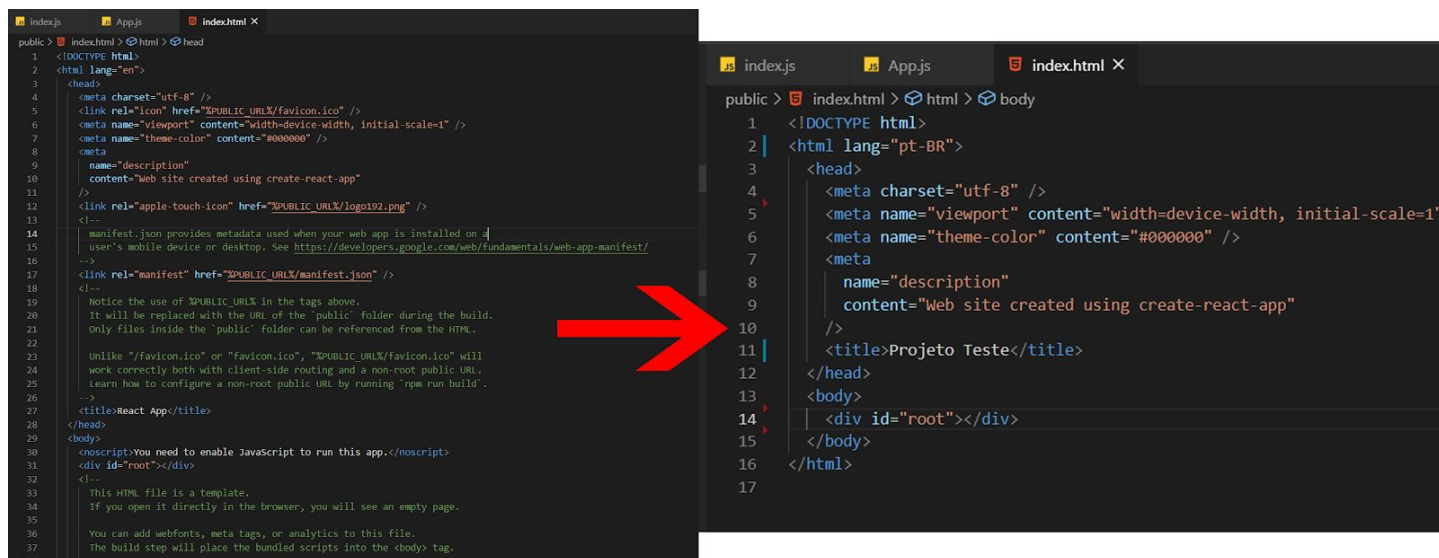


Figura 13: Arquivo “index.html” antes e após alterações.

Introduzindo o conceito de rotas

Em nosso projeto, temos a necessidade de criarmos várias páginas simultaneamente, como uma página para ser a *main*, outra para ser a página de *login* do médico, outra para ser a página de *login* do paciente, entre muitas outras que possam vir a surgir. Desse modo, surge a necessidade de

utilizarmos o conceito de Rotas ou *Routes* em nosso projeto que, basicamente, permitem interligarmos as várias páginas do projeto em diretórios diferentes, assim como ocorreria em uma página comum da internet, em que, poderíamos ter uma página inicial com um `nome`, e uma página de *login* como `nome/login`, por exemplo.

Antes de implementarmos qualquer coisa, é importante que criemos em nosso diretório “src” uma pasta que conterá todas as páginas que você deseja adicionar na página. No nosso caso, foram as páginas *main*, *login-médico* e *login-paciente*.

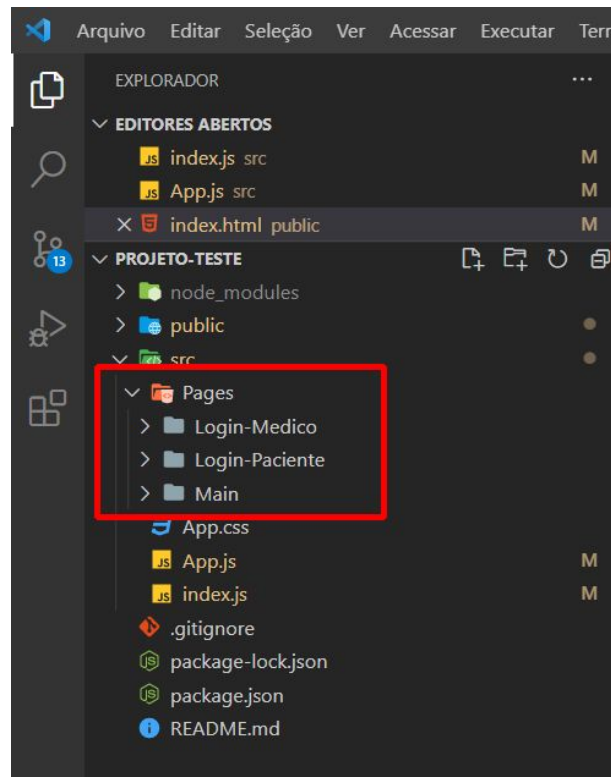


Figura 14: Criando uma pasta para as páginas do projeto

Com a organização feita anteriormente, podemos adicionar as páginas *index* de cada uma das páginas. Para isso, vamos utilizar a extensão de sintaxe para Javascript JSX.

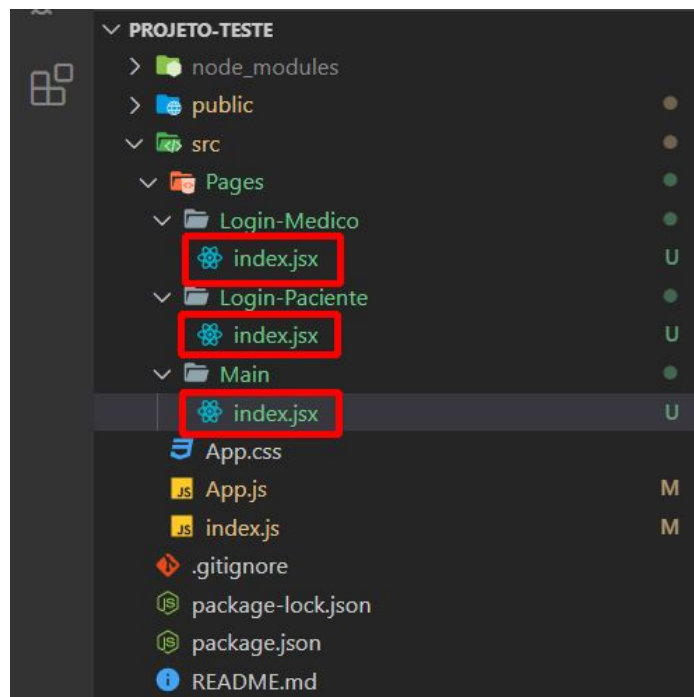


Figura 15: Criando arquivos index de cada página

O que o JSX é capaz de fazer?

Como dito anteriormente, O JSX é uma extensão de sintaxe para Javascript. O React não requer o uso do JSX. Porém, a maioria das pessoas acha prático como uma ajuda visual quando se está trabalhando com uma UI dentro do código em JavaScript. Ele permite ao React mostrar mensagens mais úteis de erro e aviso.

Porém, no momento, usaremos esta extensão apenas para adicionar informações básicas em cada página.

```
src > Pages > Main > index.jsx
1  import React from 'react';
2
3  function Main () {
4    return (
5      <div>Página Inicial</div>
6    );
7  }
8
9  export default Main;
```

```
src > Pages > Login-Paciente > index.jsx > [default]
1  import React from 'react';
2
3  function LoginPaciente () {
4    return (
5      <div>Login Paciente</div>
6    );
7  }
8
9  export default LoginPaciente;
```

```
src > Pages > Login-Medico > index.jsx > [default]
1  import React from 'react';
2
3  function LoginMedico () {
4    return (
5      <div>Login Médico</div>
6    );
7  }
8
9  export default LoginMedico;
```

Figura 16: Código básico de cada página.

Para o próximo passo, é necessário que instalemos o módulo para manipular rotas. Podemos fazer isso, basta que abramos um Prompt de Comando no diretório do projeto e o comando `npm install --save react-router-dom` seja executado, como mostrado a seguir.

```
C:\Users\lfeli>cd C:\Users\lfeli\OneDrive\Documentos\Projeto\projeto-teste  
C:\Users\lfeli\OneDrive\Documentos\Projeto\projeto-teste>npm install --save react-router-dom
```

Figura 17: Comandos para instalar o módulo de Rotas

O próximo passo é criar um arquivo, na pasta `src` com o nome de “Routes.js”.

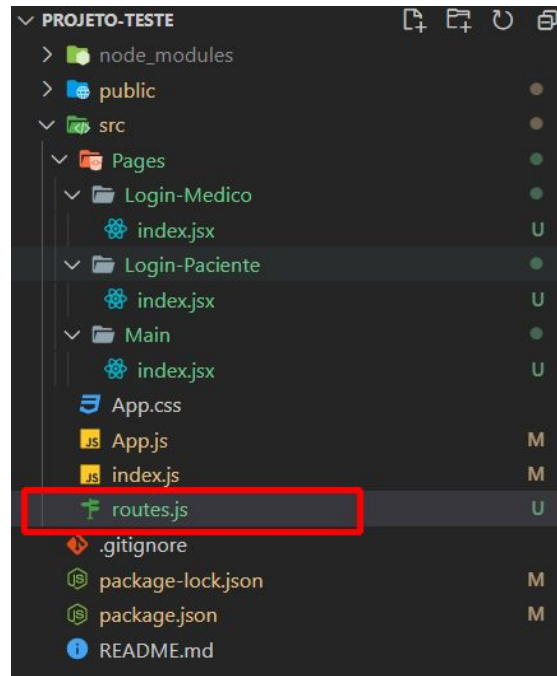


Figura 18: Criando arquivo “Routes.js” em `src`

Neste arquivo, além do React, iremos importar três componentes da biblioteca ‘react-router-dom’. Também iremos importar cada uma das páginas que queremos colocar como diretório na página e criaremos as suas respectivas rotas, da forma como foi implementado abaixo.

```

src > routes.js > [?] default
1  import React from 'react';
2  import {BrowserRouter, Switch, Route} from 'react-router-dom';
3
4  import Main from './Pages/Main';
5  import LoginMedico from './Pages/Login-Medico';
6  import LoginPaciente from './Pages/Login-Paciente';
7
8  function Routes() {
9      return (
10         <BrowserRouter>
11             <Switch>
12                 <Route path="/" exact component={Main}/>
13                 <Route path="/login-medico" component={LoginMedico}/>
14                 <Route path="/login-paciente" component={LoginPaciente}/>
15             </Switch>
16         </BrowserRouter>
17     );
18 }
19
20 export default Routes;

```

Figura 19: Implementando o arquivo de Routes

Perceba que usamos “exact” para a configuração da rota da página Main, pois, caso não o utilizássemos, teríamos um erro.

Por fim, importamos e configuramos o arquivo de Rotas no “App.js”, como mostrado abaixo:

```

src > JS App.js > ...
1  import React from 'react';
2  import Routes from './routes';
3
4  export default () => {
5      return (
6          <div>
7              <Routes/>
8          </div>
9      );
10 };
11

```

Figura 20: Importando arquivo de rotas no arquivo App

E desse modo, terminamos a configuração de rotas em nosso projeto.

Podemos ainda, criar diretórios para armazenar dados específicos do projeto, como imagens e outros componentes. Para isso, basta que nós importe nos arquivos corretos, como pode ser visto abaixo.

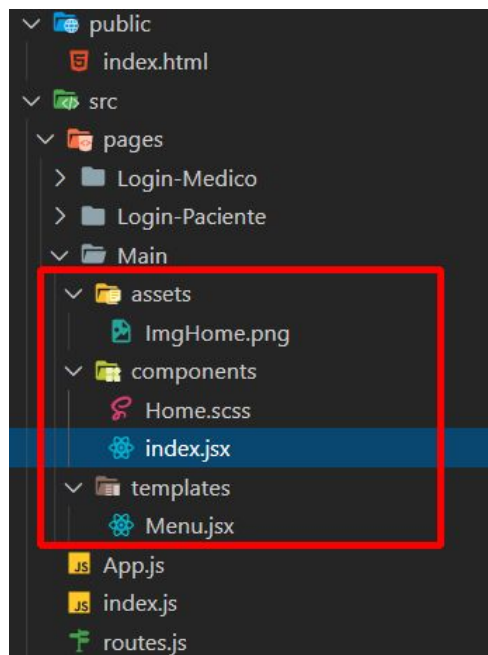


Figura 21: Dividindo os arquivos de uma pasta em diretórios distintos.

```
import React from 'react'
import imgHome from '../assets/ImgHome.png'
import './Home.scss'
```

Figura 22: Importando os arquivos no arquivo principal.

O que é um arquivo SCSS?



Em nosso projeto, utilizamos arquivos do tipo .scss para fazer o estilo de nossas páginas. Para entender como eles funcionam, precisamos, primeiramente, conhecer um pouco mais sobre o Sass. O Sass é uma extensão da linguagem CSS, porém, como a própria linguagem se declara, o Sass é um CSS com superpoderes.

SCSS Sass

```
.button {
  padding: 3px 10px;
  font-size: 12px;
  border-radius: 3px;
  border: 1px solid #e1e4e8;
}
```

CSS

```
.button {
  padding: 3px 10px;
  font-size: 12px;
  border-radius: 3px;
  border: 1px solid #e1e4e8;
}
```

Figura 23: Comparação de sintaxe do Sass com o CSS 1.

Como você pode ver, pela imagem anterior, o Sass e o CSS têm uma sintaxe semelhante em todos os sentidos apresentados ao CSS. Mas, o que a faz diferente? Vamos ver uma das coisas da qual o Sass é capaz de fazer a seguir.

SCSS Sass

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

CSS

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Figura 24: Comparação de sintaxe do Sass com o CSS 2.

Como você deve ter percebido, o Sass torna a nossa vida mais fácil, pois nos poupa de ter de repetir os mesmos seletores várias vezes. Agora, podemos escrever as regras de um estilo dentro de outro. É claro que o Sass oferece inúmeras outras facilidades para quem vai trabalhar com a estilização, porém, optamos por apresentar, apenas para exemplificar, apenas uma das coisas das quais ele é capaz de fazer para que tenhamos uma noção de como aplicá-lo em um projeto como o nosso.

Como instalar o SASS?

Instalar o SASS é fácil. Basta acessarmos o diretório de nosso projeto e executar o comando `npm install -g sass`, como mostrado a seguir.

```
Prompt de Comando
Microsoft Windows [versão 10.0.19042.804]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\lfeli>cd C:\Users\lfeli\OneDrive\Documentos\Projeto\projeto-teste

C:\Users\lfeli\OneDrive\Documentos\Projeto\projeto-teste>npm install -g sass
```

Figura 25: Instalação do SASS

Finalizada a instalação, já podemos utilizar o SASS em nosso projeto.

Como o SASS foi utilizado em nosso projeto?

Em nosso projeto, fizemos uso de arquivos Sass para fazer o estilo das diferentes páginas criadas.

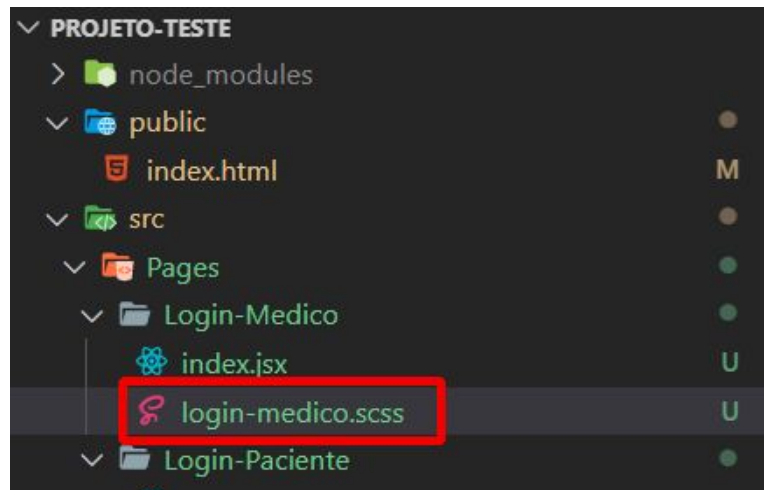


Figura 26: Exibindo arquivo .scss de uma página.

Finalmente, para adicionar o arquivo .scss no arquivo principal, bastou fazer a sua importação, na forma padrão de exportação de arquivos no React.

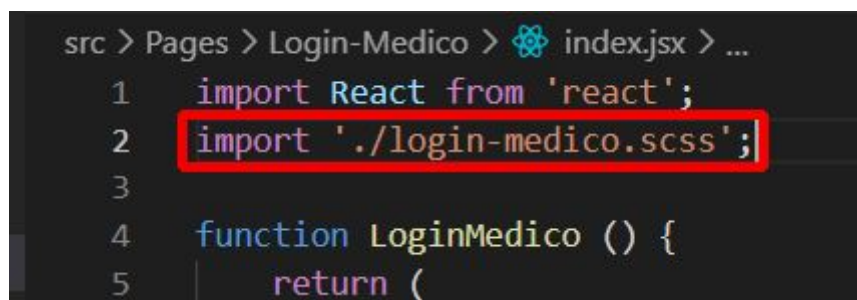


Figura 27: Importando arquivo .scss no arquivo principal.

FRONTEND

O frontend foi desenvolvido utilizando react e algumas outras dependências foram colocadas para ocorrer a conexão com o que foi desenvolvido no backend, para essa conexão ocorrer foi utilizado o axios.

O que é o axios?



Axios é um cliente HTTP baseado em Promissas para fazer requisições. Esse cliente “http” pode ser utilizado tanto no navegador quanto no node.js. No nosso sistema foi utilizado as funções de get, que serve para pegar dados que estão no backend por meio do formato “json”, post que serve para criar um novo objeto e a função delete que serve para deletar um objeto que está contido no banco de dados.

Um exemplo dessas funções está nesse trecho de código:

```
c > service > consulta.js > default > listByTime
1  import api from './api'
2
3  export default {
4    list: () => {
5      return api.get(process.env.REACT_APP_ENDPOINT_LIST_APPOINTMENTS);
6    },
7
8    listByTime: (id, date) => {
9      return api.get(process.env.REACT_APP_ENDPOINT_LIST_TIMES_APPOINTMENTS + id + "&date=" + date);
10   },
11
12   byId: (id) => {
13     return api.get(process.env.REACT_APP_ENDPOINT_BY_ID_APPOINTMENT + id);
14   },
15
16   save: (medicId, patientId, time) => {
17     return api.post(process.env.REACT_APP_ENDPOINT_CREATE_APPOINTMENT, {medicId, patientId, time}).
18     then().catch();
19   },
20
21   delete: (id) => {
22     return api.delete(process.env.REACT_APP_ENDPOINT_DELETE_APPOINTMENT + id);
23   },
24 }
```

No qual “api” é uma importação do axios, da seguinte forma: `axios.create({ baseUrl })`. E a baseUrl é uma url que vai ser o padrão do nosso projeto, no caso, nossa url padrão está presente em <http://ec2-3-138-120-34.us-east-2.compute.amazonaws.com/>, e baseado nessa url é feita as demais requisições usando os endpoints. Endpoints vai ser alguma função criada no backend em formato json onde as requisições são feitas com o auxílio do axios.

Além disso, para uma maior segurança do nosso sistema foi utilizado o .env onde ele está sendo referenciado na imagem acima por meio do “process.env” e os demais textos da sessão, são os endpoints que estão sendo referenciados neste arquivo .env.

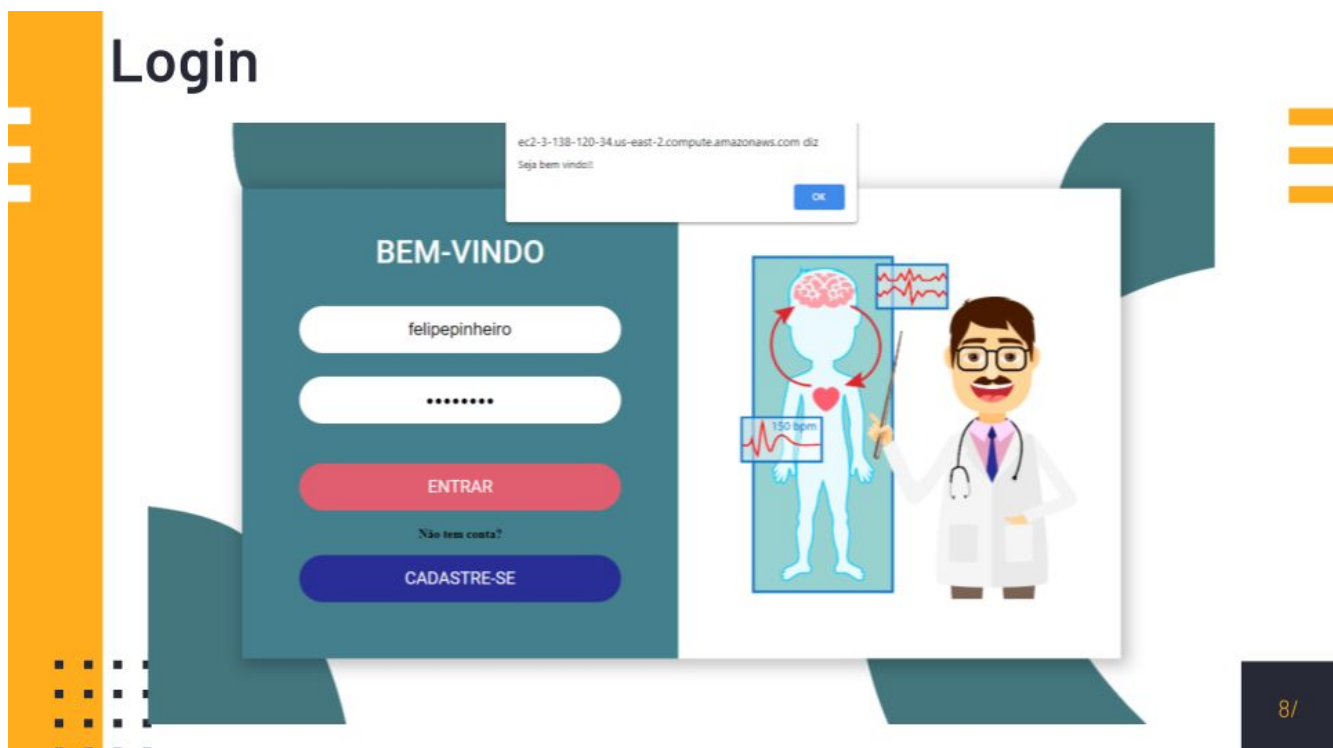
Telas criadas

- Tela de home principal

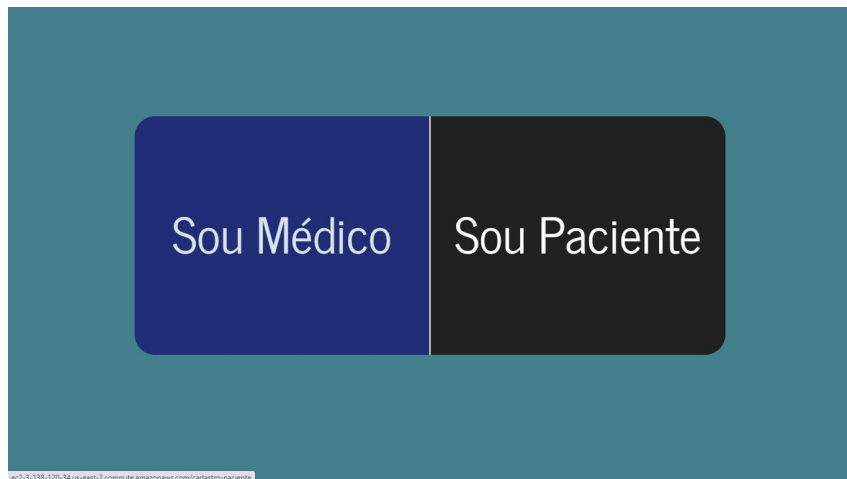


Essa tela se subdivide em 3 e os dados presentes nela são todos estáticos pois não é necessário requisições nessa página. Nela têm as telas de sobre nós que vai ser percorrido um pouco sobre o nosso projeto. E por último tem a tela de “nossa equipe” que vai mostrar os integrantes e suas funções.

Login



Na tela de login podemos ter acesso às consultas e exames inserindo credenciais válidas. Caso o paciente ou até mesmo o médico não tenha seu acesso no sistema encontra-se também uma tela para realizar o cadastro de acordo com o perfil de médico ou paciente. Veja as imagens abaixo:



Tipo de usuário para o cadastro

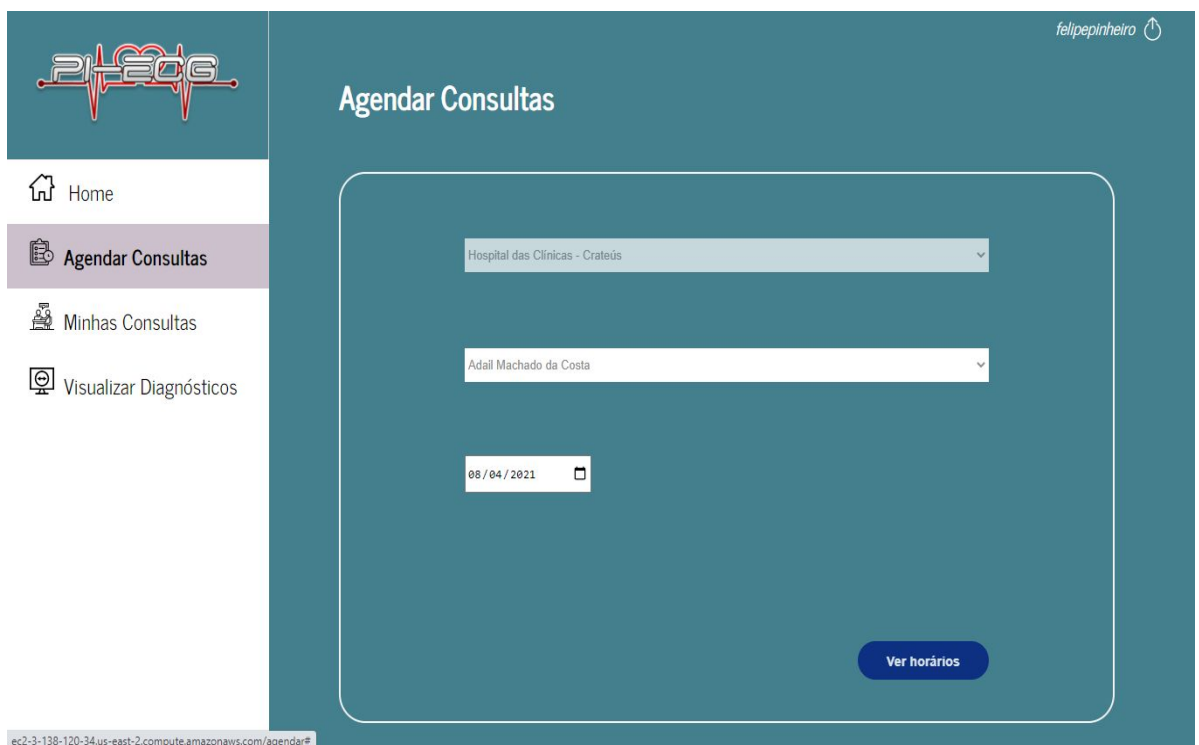
A imagem mostra uma tela de cadastro com o título 'Cadastro' em azul. O formulário contém os seguintes campos: 'Nome Completo' (preenchido com 'Adalberto Felipe Pinheiro Chaves'), 'CPF' (com máscara '***-**-**'), 'Número' (preenchido com '7846454223'), 'Cidade' (preenchido com 'Guaraciaba do Norte'), 'Estado' (preenchido com 'CE'), 'Endereço' (com subcampos 'Rua...' e 'Bairro...'), 'Email' (preenchido com 'felipe.afp27@gmail.com'), 'Nome de usuário' (preenchido com 'felipepinheiro'), 'Senha' (com máscara '*****') e 'Confirmar Senha' (com máscara '*****'). No final do formulário, há dois botões: 'Cadastrar' e 'Já possui uma conta? Fazer Login'.

Tela de cadastro

Finalizando a área dedicada ao paciente, temos as tela de início (home), agenda consultas e ver as consultas já feitas pelo paciente. Observe as respectivas imagens abaixo.



Tela de início

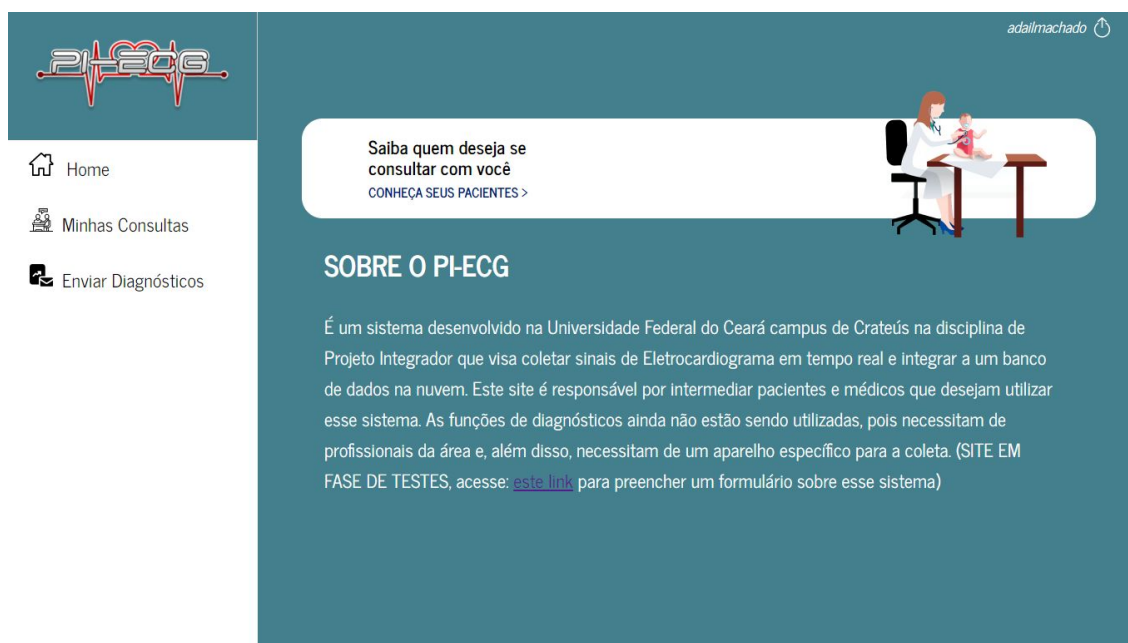


Data e médico a ser feito a consulta




Horários disponíveis para aquele dia selecionado

As páginas do médico são semelhantes ao do paciente, visto que possui algumas alterações a fim de cumprir as regras de negócios, tais como verificar as consultas que foram marcadas para ele e a quem marcou(paciente), também temos o detalhe de que o médico pode apagar(cancelar) consulta por um motivo genérico nessa versão do projeto. Nas próximas linhas podemos observar.



Tela de início do médico



Home

Minhas Consultas

Enviar Diagnósticos


adailmachado

Minhas consultas

Data	Hora	Local	
1/4	8:00	Hospital das Clínicas - Crateús	
19/3	7:00	Hospital das Clínicas - Crateús	
15/4	9:30	Hospital das Clínicas - Crateús	
8/4	9:30	Hospital das Clínicas - Crateús	

ec2-3-138-120-34.us-east-2.compute.amazonaws.com/minhas-consultas-medico#

Lista de consultas



Home

Minhas Consultas

Enviar Diagnósticos

adailmachado

Consulta

ID DA CONSULTA: #109

PACIENTE: Adalberto Felipe Pinheiro Chaves

LOCAL: Hospital das Clínicas - Crateús

TIPO DE CONSULTA: Exame de Eletrocardiograma

STATUS DA CONSULTA: Pendente

Cancelar consulta

ec2-3-138-120-34.us-east-2.compute.amazonaws.com/detalhes-consulta-medico#

Detalhes da consulta selecionada

Assim finalizamos a abordagem das telas com mais relevância do sistema.

Backend

Responsável pela a integração com o banco e por prover as funcionalidades base do sistema.

Tecnologias utilizadas

Flask	Microframework para desenvolvimento de serviços web baseados em HTTP
Swagger	Conjunto de ferramentas para a documentação e teste de APIs HTTP baseado em OpenAPI 3
Docker	Conjunto de ferramentas para gerenciamento de contêineres
ECS	Plataforma para implantação de serviços contidos em contêineres

Objetivos

Gerenciamento de gráficos de ECG, pacientes, médicos e consultas, assim como o controle do registro e autenticação de usuários da plataforma.

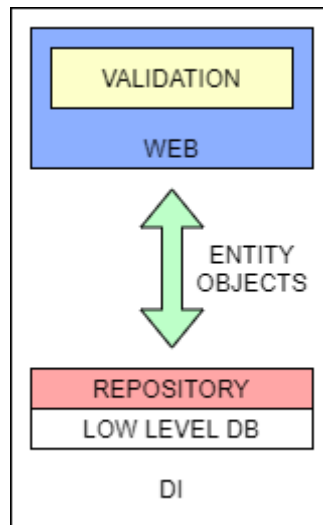
Funcionalidade

As informações são disponibilizadas através de uma API RESTful com os seguintes recursos:

/sample	Gerenciamento de amostras de ECG
/user	Gerenciamento de usuários
/session	Criação de sessões e controle de acesso
/appointment	Gerenciamento de consultas

Informações detalhadas sobre esses recursos podem ser obtidas no formato OpenAPI 3 na wiki do backend, disponível em <https://github.com/diefesson/ecg-cloud-now/tree/wiki> (requer um visualizador compatível, ex: <https://swagger.io/tools/swagger-ui>)

Estrutura e implementação



A aplicação foi desenvolvida em Python e estruturada de acordo com o conceito Domain Driven Design (DDD), separando a lógica de aplicação da persistência de recursos.

As configurações são obtidas através de variáveis de ambiente e em segundo caso arquivos de ambiente (.env), que é uma estratégia compatível e amplamente utilizada na plataforma Docker por ser fácil de configurar e tornar os detalhes da implementação transparentes para a implantação.

A escolha do Flask se deu pela fácil integração e capacidade de extensibilidade, além de ampla documentação disponível online.

Próximos passos

- Integração com a coleta em tempo real.
- Autenticação padronizada baseada em JWT.
- Separação da lógica de controle e web.
- ORM e versionamento dos esquemas de banco de dados.
- Gerenciamento e auxílio na emissão de diagnósticos.
- Análise de anomalias de ECG.
- Suporte a formatos de arquivo padrões para importação e exportação ECG

4-Banco de dados

Responsável pelo armazenamento e segurança das informações.

Com os diversos tipos de bancos de dados existentes para o setor de desenvolvimento de sistemas é necessário fazer pesquisas para achar a melhor infraestrutura para o projeto.

Como opções temos várias, dentre elas se destacam MySQL, MongoDB, MariaDB e SQLite. Podemos encontrar bancos relacionais e não relacionais. A escolha foi baseada tanto na necessidades, e na implementação, consequentemente a curva de aprendizagem, manuseios, manutenções. Ficou definido a escolha do MySQL, pois trata-se de um banco relacional e de fácil uso, além do mais, utilizamos quando cursamos a disciplina de Banco de Dados. Logo, sendo o mais acessível aos membros da equipe, pois já não se tratava de algo novo, o que resultou em ganho de tempo.

É importante destacar, que a implementação do banco de dados se deu por meio da plataforma RDS que encontrada facilmente nos serviços da Amazon Web Services, também conhecido como AWS. Além de possuir um custo financeiro baixo e com um certo nível de dados gratuito, também nos permite uma grande acessibilidade tanto a nível de projeto, quanto de confiabilidade, velocidade e segura.

Pois a plataforma é rigidamente atualizada e contém toda equipe de engenheiros para resolver falhas do serviço e oferecer o suporte necessário ao clientes. O mais importante é que elevemos o nosso amadurecimento a outro patamar superior, pois foi utilizado um serviço de nuvem, o que conta muito a nível de projeto, tanto na disponibilidade dos dados como também no uso pelo time, já que equipe do projeto encontrava-se em regime de “home office”.

Tabelas criadas:



figura X(tabelas encontradas no banco de dados do projeto).

Resumidamente o banco de dados guarda as principais informações que cumprem os requisitos da solução proposta. Podemos encontrar a tabela para guardar os usuários(user), onde temos colunas como: nome, senha, endereço, telefone e etc. Informações básicas que todas as pessoas possuem. O sistema possui também uma tabela para guardar os laudos, visto que ter e manter um banco de dados otimizados e com as devidas “normalizações” é uma “mina de ouro” para futuros estudos de ciência de dados e pesquisas.

REFERÊNCIAS:

1. Site oficial do React, <https://pt-br.reactjs.org/>
2. Tutorial de como usar rotas no React, https://www.youtube.com/watch?v=ZQUyJid_EAM
3. Documentação sobre o Sass, <https://sass-lang.com/documentation/style-rules>
4. Como instalar Sass, <https://sass-lang.com/install>
5. O que é a AWS, <https://aws.amazon.com/pt/what-is-aws/>
6. O que é o RDS, <https://aws.amazon.com/pt/rds/>