



SISTEMA DE AGENDAMENTO ONLINE DE CONSULTAS

Trabalho de Conclusão de Curso

Engenharia da Computação

Davi da Mota Nogueira

Orientador: Prof. Edison de Queiroz Albuquerque



UNIVERSIDADE
DE PERNAMBUCO

**Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação**

DAVI DA MOTA NOGUEIRA

**SISTEMA DE AGENDAMENTO ONLINE
DE CONSULTAS**

Monografia apresentada como requisito parcial para Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Outubro de 2022.

Nogueira, Davi da Mota

Sistema de Agendamento Online de Consultas/ Davi da Mota
Nogueira.– Recife - PE, 2022.

x, 44 f. : il. ; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de
Computação) Universidade de Pernambuco, Escola Politécnica de
Pernambuco, Recife, 2022.

Orientador: Prof. Dr. Edison de Queiroz Albuquerque.

Inclui referências.

1. Aplicação *Web*. 2. Marcar Consultas. 3. Containerização. I.
Sistema de Agendamento Online de Consultas para o
NUTES-CISAM. II. Albuquerque, Edison de Queiroz. III. Universidade
de Pernambuco.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 21/10/2022, às 14h00min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **DAVI DA MOTA NOGUEIRA**, orientado(a) pelo(a) professor(a) **EDISON DE QUEIROZ ALBUQUERQUE**, sob título Sistema De Agendamento Online De Consultas, a banca composta pelos professores:

LUIS CARLOS DE SOUSA MENEZES (PRESIDENTE)

EDISON DE QUEIROZ ALBUQUERQUE (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

Aprovada Aprovada com Restrições* Reprovada

e foi-lhe atribuída nota: 8,0 (oit)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

Luis Carlos de Sousa Menezes
AVALIADOR 1: Prof (a) **LUIS CARLOS DE SOUSA MENEZES**

Edison de Queiroz Albuquerque
AVALIADOR 2: Prof (a) **EDISON DE QUEIROZ ALBUQUERQUE**

AVALIADOR 3: Prof (a)

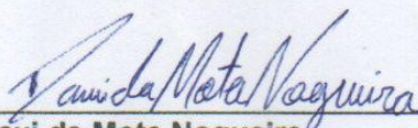
* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Dedico este trabalho aos meus familiares e amigos.

Autorização de publicação de PFC

Eu, **Davi da Mota Nogueira** autor(a) do projeto de final de curso intitulado: **Sistema De Agendamento Online De Consultas**; autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

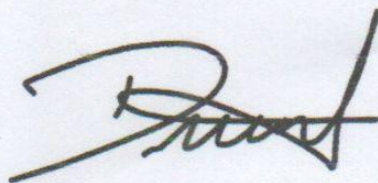


Davi da Mota Nogueira



Orientador(a): **Edison de Queiroz Albuquerque**

Coorientador(a):



Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 21/10/2022

Resumo

O sistema médico é muito importante para o cuidado com as pessoas, o Centro Integrado de Saúde Amaury de Medeiros (CISAM) em especial, cuida da parte materna da saúde. Como um sistema público de saúde, até uma simples consulta por teleconferência pode dificultar a vida das pessoas, pois algumas vezes elas precisam sair de suas residências e enfrentar dificuldades só para fazer um cadastro no hospital. Este projeto visa ajudar as todas as pessoas, com um sistema de marcação de consulta online, utilizando ferramentas amplamente utilizadas pelo mercado de trabalho, como *JavaScript*, *Node.js*, *Hypertext Markup Language* (HTML) e *Cascading Style Sheets* (CSS). O trabalho visa a troca de uma planilha que os funcionários do CISAM precisam preencher por telefone com o paciente, como resultados foi feita uma aplicação *web* para obter uma lista de pacientes e marcação de consultas, uma estrutura padronizada e organizada para desenvolvimento futuro, escalabilidade e um forma de colocar o projeto em produção a partir de containerização.

Palavras-chave: Aplicação, Projeto, Marcar, Pacientes, Contêiner, *Frontend*, *Backend*, Consulta, Banco.

Abstract

The medical system is very important for the care of people, the Integrated Health Center Amaury de Medeiros (CISAM) in particular, takes care of the maternal part of health. As a public health system, even a simple consultation by teleconference can make people's lives difficult, as they sometimes have to leave their homes and face difficulties just to register at the hospital. This project aims to help everyone with an online query markup system, using tools widely used by the job market, such as JavaScript, Node.js, Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). The work aims to exchange a spreadsheet that CISAM employees need to fill in with the patient over the phone, as a result a web application was made to obtain a list of patients and schedule appointments, a standardized and organized structure for future development, scalability and a way to put the project into production from containerization.

Keywords: Application, Project, Bookmark, Patients, Container, Frontend, Backend, Consultation, Bank.

Lista de ilustrações

Figura 1. Tabela de equivalência de funções do CRUD com SQL	13
Figura 2. Relação entre as funções do CRUD com do HTTP	13
Figura 3. Diagrama do funcionamento da arquitetura MVC	15
Figura 4. Comparação entre uso de <i>containers</i> e de VM	17
Figura 5. Modelo Lógico do Banco de Dados para Marcar Consultas	22
Figura 6. Página de <i>login</i> pensada no <i>Figma</i>	23
Figura 7. Página dos pacientes pensada no <i>Figma</i>	24
Figura 8. Caixa para marcar consulta e consulta confirmada	25
Figura 9. Página sobre os detalhes do paciente escolhido	26
Figura 10. Página para editar dados do paciente e caixa de sucesso de edição	28
Figura 11. Disposição das pastas do projeto	30
Figura 12. Página de <i>login</i> produzida	31
Figura 13. Página de pacientes produzida	31
Figura 14. Caixa para marcação de consulta, por paciente	32
Figura 15. Página de edição dos dados de paciente produzida	33
Figura 16. Disposição final das pastas	34
Figura 17. <i>Dockerfile</i> escrito para aplicação	35
Figura 18. Arquivo de pacotes de dependências do <i>Node.js</i> para a aplicação	36
Figura 19. Arquivo <i>Docker Compose</i> , necessário para criar mais de um <i>container</i> de uma vez	37
Figura 20. Conteúdo do arquivo <i>.env</i> , para organizar variáveis	38

Lista de abreviaturas e siglas

API – *Application Programming Interface*

CISAM – Centro Integrado de Saúde Amaury de Medeiros

CRUD – *Create Read Update Delete*

CSS – *Cascading Style Sheets*

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Transfer Protocol*

MOR – Mapeamento Objeto-Relacional

MVC – *Model-View-Controller*

NUTES – Núcleo de Teleconferência em Medicina

REST – *Representational State Transfer*

SGBD – Sistema de Gerenciamento de Banco de Dados

SQL – *Standard Query Language*

TFD – Tratamento Fora de Domicílio

UI – *User Interface*

UPE – Universidade de Pernambuco

URI – *Universal Resource Identifier*

VM – *Virtual Machine*

WWW – *World Wide Web*

Sumário

1.	Introdução	10
2.	Fundamentação Teórica	12
2.1.	Aplicação Web	12
2.2.	Create Read Update Delete (CRUD)xRepresentational state Transfer (REST)	12
2.3.	Model View Controller (MVC)	14
2.4.	Node.js e Linguagens	15
2.5.	Banco de Dados	16
2.6.	Containerização	16
3.	Materiais e Métodos	18
3.1.	Transformação do Atual	18
3.2.	Ferramentas	18
3.2.1.	Backend	18
3.2.2.	Frontend	18
3.2.3.	Banco de Dados e Containers	19
3.3.	Metodologia	19
4.	Resultados	21
4.1.	Análise e Design	21
4.2.	Desenvolvimento o Frontend	22
4.3.	O Backend	29
4.4.	Implementação dos Containers	34
5.	Conclusão e Trabalhos Futuros	39
	Referências	40

1. Introdução

O Centro Integrado de Saúde Amaury de Medeiros, também conhecido como CISAM-UPE, foi inaugurado em 23 de Janeiro de 1946, com o intuito de ser a maternidade da Encruzilhada e desde sua criação, o local vem passando por atualizações no atendimento, ofertando cada vez mais serviços, como pré-operatório, pós-operatório, serviço de humanização, ensino para estudantes da UPE, pediatria e em 2014 além da reabertura da Sala de Vacinação e a Implantação do Projeto Doula Comunitária Voluntária, surgiu a instalação do Núcleo de Teleconferência em Medicina, chamado de NUTES[1].

O núcleo é a área de principal interesse neste projeto, pois ele tem como objetivo, desenvolver ações no apoio à assistência em saúde[2]. Uma área que cresceu nessa pandemia foi a teleconsulta, mas mesmo depois de dois anos do início da pandemia, e as pessoas retomando à sua nova rotina, a teleconsulta foi vista com bons olhos para a agilidade e disponibilidade de médicos sem que as pessoas precisarem se deslocar, pois há o motivo de idade, distância e conveniência.

Mas o grande problema que ocorre no CISAM é o fato de que para agendar alguma consulta, mesmo que não seja uma teleconsulta, é necessário que a pessoa ou responsável compareça ao hospital na Encruzilhada, mesmo quem reside no agreste ou sertão, tem que pegar um transporte disponibilizado pela prefeitura, viajar por horas, pegar fila para agendar uma consulta, quando consegue, pois há situações que mesmo depois de viajar por horas, não consegue nem agendar a consulta desejada.

Complementando isso, algumas tele-consultas são marcadas por telefone com os funcionários do NUTES, tendo a complicação da disponibilidade da pessoa com o telefone, o sinal, e a quantidade de páginas que o funcionário precisa acessar, para marcar a consulta, pois além do mesmo ter que completar os campos do paciente, é preciso consultar a agenda dos médicos para poder marcar, sendo bem inconveniente e demorado para o funcionário e o paciente. Além do fato que nos dias atuais é preciso mudar para sistemas que refletem o avanço constante[3]. Por isso que a utilização de uma simples aplicação *web*, para os pacientes e para os

funcionários marcarem as consultas, são necessárias para resolver esses problemas.

2. Fundamentação Teórica

2.1. Aplicação Web

A importância da teleconsulta é uma opção muito importante principalmente para pessoas que vivem em municípios mais remotos e com IDH mais baixo[4]. Por isso, para resolver um problema como esse, a solução de Jing Bai[5], professora do departamento de engenharia elétrica da China, foi desenvolver um sistema no *World Wide Web* (WWW) é boa, pois consegue disponibilizar um serviço completo sem se deslocar da sua residência. Sem um sistema desse, o paciente precisa se deslocar a fim de chegar ao hospital, mesmo que seja só para um cadastro no sistema, elas precisam enfrentar dificuldades com o transporte do Tratamento Fora de Domicílio (TFD), pois o clientelismo praticado por vereadores e outros políticos com intuito de barganha política, a desinformação e uma legislação inconsistente, ajudando o aumento do número de clientelismo[6].

Logo, a proposta para solucionar o problema apresentado é a construção de dois websites: um voltado para os pacientes se cadastrarem, e outro para os funcionários acessarem e marcarem as consultas requisitadas pelos pacientes. Para a conexão dos dois será utilizado o mesmo banco de dados para armazenar as informações dos pacientes e suas respectivas consultas.

Primeiramente, para saber como será o funcionamento dos websites tem que se saber como será dada a comunicação do servidor com o público. O protocolo de comunicação para acesso ao *World Wide Web* (WWW) será o *Hypertext Transfer Protocol* (HTTP), o mais utilizado no mundo inteiro[7].

Esse protocolo é baseado em requisições na qual são compostas por mensagens enviadas por clientes à aplicação, onde o mesmo retorna uma mensagem de resposta[8]. Normalmente, o método de acesso utilizado para o cliente enviar as requisições são os navegadores, enquanto a aplicação será mantida e processada em um servidor.

2.2. Create Read Update Delete (CRUD) x Representational state Transfer (REST)

O *REpresentational State Transfer* (REST) é um tipo de arquitetura de solução *web* na qual utiliza operações baseadas nos cabeçalhos HTTP, onde há um conjunto de operações que o desenvolvedor pode colocar na sua aplicação, onde

por meio de *Universal Resource Identifier* (URI), é possível a comunicação entre sistemas mais complexos[9].

Há também o *Create Read Update Delete* (CRUD), um conjunto de operações que interagem diretamente com o banco de dados. Essas funções só é uma maneira de explicar que o programa pode manipular dados, com um leve ajuste de configurações, é possível mapear as operações do CRUD com os cabeçalhos do HTTP, na Figura 1, é possível ver uma tabela com as descrições e equivalências das funções de manipulação. Sendo a primeira, de criação de entidade(s), a segunda de leitura da(s) entidade(s), em seguida a função de atualização e por último, a exclusão de uma ou mais entidades da tabela[10].

Figura 1. Tabela de equivalência de funções do CRUD com SQL.

NAME	DESCRIPTION	SQL EQUIVALENT
Create	Adds one or more new entries	Insert
Read	Retrieves entries that match certain criteria (if there are any)	Select
Update	Changes specific fields in existing entries	Update
Delete	Entirely removes one or more existing entries	Delete

Fonte: <https://nordicapis.com/crud-vs-rest-whats-the-difference/>

É muito utilizado o termo CRUD para designar aplicações que contém essas funções de manipulação de dados, mas há uma grande diferença entre os dois termos, citados anteriormente, pois enquanto um é só um acrônimo das funções representadas, o REST é um estilo de arquitetura de *software*, comumente usada para produções de *web Application Programming Interfaces* (APIs), sendo um termo com definições mais complexas. Na Figura 2 é possível entender o porquê as pessoas se confundem[10].

Figura 2. Relação entre as funções do CRUD com do HTTP.

CRUD	HTTP
CREATE	POST/PUT
READ	GET
UPDATE	PUT/POST/PATCH
DELETE	DELETE

Fonte: <https://nordicapis.com/crud-vs-rest-whats-the-difference/>

Portanto, as duas aplicações propostas neste trabalho serão aplicações RESTful, pois são sistemas compatíveis com REST e que desmembram as responsabilidades do cliente e do servidor[9].

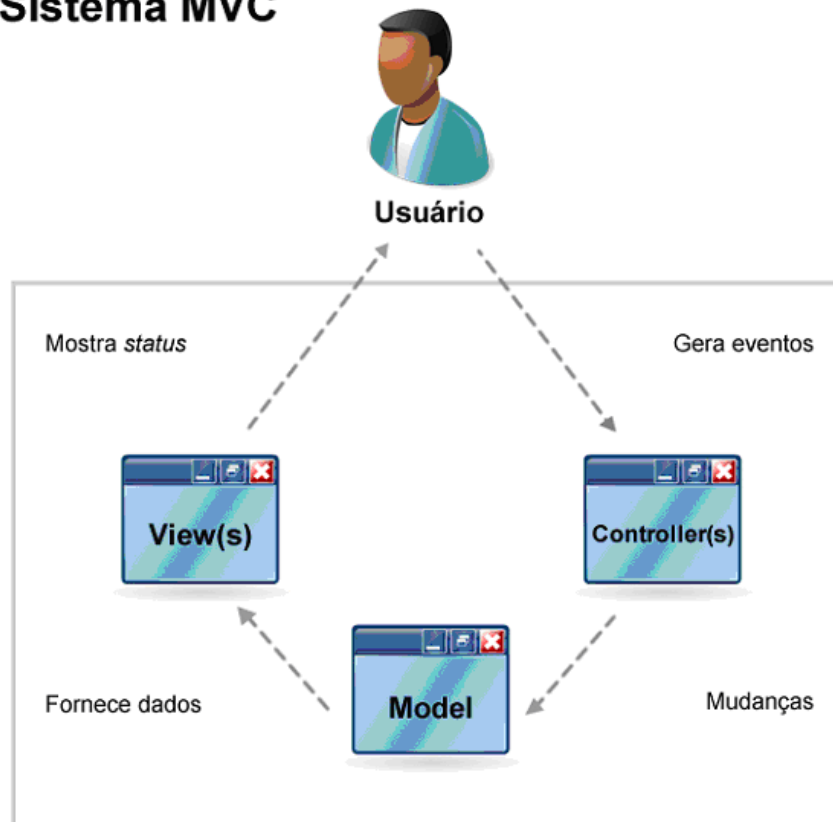
2.3. Model View Controller (MVC)

O *Model-View-Controller* (MVC) é um padrão de arquitetura de estruturação de projeto de *software* comumente sendo orientado a objetos. Nela estão bem definidas as separações de *frontend* (parte do software que controla a parte visual, que interage com usuário), *backend* (seção responsável por estruturar a aplicação e de apoiar as informações que o usuário necessitar), e as regras de negócio. O modelo é relacionado às regras de negócio, pois ela relaciona e configura o banco de dados e a aplicação desenvolvida. A visualização é comumente associada com páginas de *User Interfaces* (UIs) e *frontend*, mas as *views* podem não ser gráficas, só escritas[11].

O controle é um objeto que permite a manipulação das *views*, a partir do que o usuário escolher, como John Deacon[11] explica de forma simplificada, que o *controller* maneja os *inputs* enquanto as *views* gerenciam os *outputs* para o usuário. Na Figura 3, pode-se observar de maneira diagramática, o funcionamento da arquitetura.

Figura 3. Diagrama do funcionamento da arquitetura MVC.

Sistema MVC



Fonte: <http://melissalobo.blogspot.com/2014/08/o-que-e-mvc.html>

2.4. Node.js e Linguagens

A linguagem de *backend* e *frontend* (parcialmente) utilizada para o projeto foi *JavaScript* - pois como o propósito do desenvolvimento das aplicações é de aperfeiçoar e escalonar o trabalho já feito, é de extrema importância utilizar a linguagem mais usada para desenvolvimento web[12][13]. Por isso, a fim de tornar o desenvolvimento contínuo passando para outro grupo de desenvolvedores do NUTES-CISAM, a base de linguagem do projeto é HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) e *JavaScript*, apenas a última é usada para construção do *backend*.

Para facilitar o desenvolvimento do *backend* foi utilizado a ferramenta *Node.js*, um *runtime* de *JavaScript* (*runtime* pode ser entendido como tempo de execução, mas, nesse caso, funciona como uma máquina virtual que gerencia programa na linguagem) que tem uma performance incrível e encaixa perfeitamente com o projeto, pois ela é facilmente escalável[14].

Sua performance é caracterizada pela sua capacidade de concorrência e assincronicidade, pois é possível manusear várias conexões de forma concorrente. Além disso, a ferramenta prioriza os cabeçalhos HTTP, por onde ocorre o funcionamento do projeto, por isso que *Node.js* é amplamente utilizado para desenvolvimento web[15].

Para complementar o uso do *Node.js*, foi implementado ao projeto outras ferramentas, duas delas sendo *frameworks*, a *Express*¹, utilizada para prover de forma reduzida e flexível, um conjunto de recursos robustos para o desenvolvimento de aplicações *web* e móveis. Outro *framework* é o *Bootstrap*², bastante utilizado para desenvolvimento *frontend*, pois há uma facilidade enorme com essa ferramenta, já é disponibilizado uma caixa de ferramenta com vários artifícios para produção de interfaces uniformes, ainda tendo possibilidade de customização própria dos recursos.

O projeto faz uso de uma linguagem de *template*, a *Handlebars*³, isso significa que é uma linguagem baseada e tendo recursos de outra, que na ocasião dessa ferramenta é o HTML, mas há como adicionar funções do próprio *Handlebars*, que facilmente podem exercer operações mais complexas na apresentação do *frontend*.

2.5. Banco de Dados

Para o armazenamento dos dados ser compatível com toda a estrutura do projeto relacional e como uma exigência do CISAM, a linguagem disponível e de melhor desempenho é a *Standard Query Language* (SQL), na qual o Sistema de Gerenciamento de Banco de Dados (SGBD) escolhido foi o MySQL, a segunda ferramenta mais utilizada no mundo[16].

Como dito anteriormente, como será um projeto de desenvolvimento contínuo, foi preciso utilizar um Mapeamento Objeto-Relacional (MOR) que fosse compatível com o MySQL, dessa maneira foi escolhido a ferramenta *Sequelize*⁴, para facilitar as interações entre o projeto e o banco de dados.

2.6. Containerização

Algo que foi bastante requisitado, foi a necessidade de colocar o projeto em *containers*, com o intuito de manejar várias versões do projeto de forma leve e

¹ <https://expressjs.com/pt-br/>

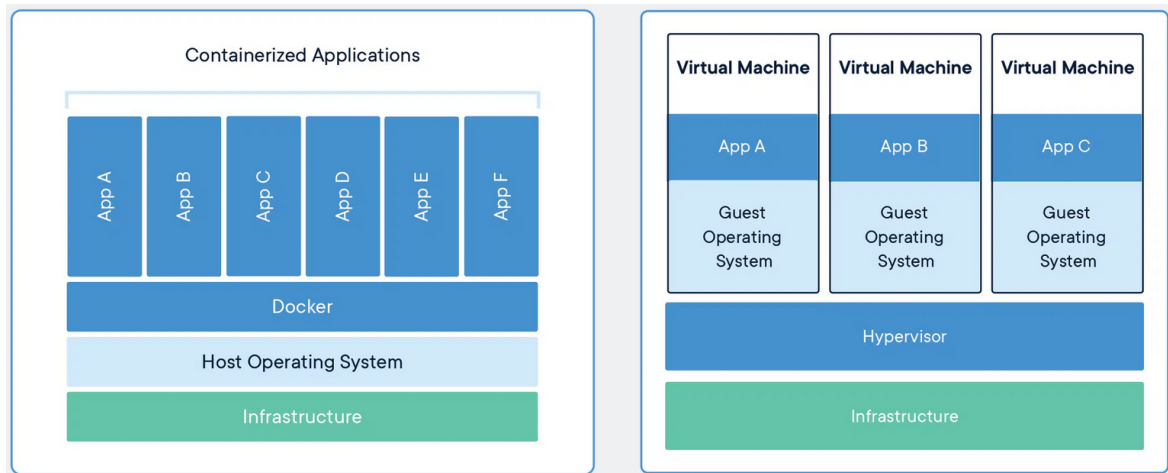
² <https://getbootstrap.com/>

³ <https://handlebarsjs.com/>

⁴ <https://sequelize.org/>

independente. Além disso, é possível manter vários *containers* de aplicações diferentes em uma mesma infraestrutura, dessa maneira, reforçando a segurança e facilitando a manutenção. Na Figura 4 é possível ver a diferença entre a utilização de *containers* e de *Virtual Machines* (VM), pois só há um sistema operacional, e o Docker para controlar possíveis mudanças nas aplicações[17].

Figura 4. Comparação entre uso de *containers* e de VM.



Fonte: <https://www.docker.com/resources/what-container/>

3. Materiais e Métodos

A solução proposta neste trabalho visa a implementação de uma aplicação *web*, para os funcionários do NUTES-CISAM marcarem as consultas dos pacientes cadastrados em um banco de dados do centro médico.

3.1. Transformação do Atual

Atualmente, o funcionamento das teleconsultas, já citada no capítulo 1, é feita de maneira nada conveniente, a pessoa precisa sair de sua residência para conseguir se cadastrar no sistema, para depois, por telefonema, marcar uma teleconsulta com o funcionário.

O método utilizado é ineficaz, devido ao tempo desperdiçado pelos pacientes e funcionários, pois é preciso completar uma planilha de 23 colunas com as informações do solicitante, sendo as 3 últimas realmente demandante do funcionário, sendo: a escolha do médico, especialidade do atendimento e a data da consulta, tudo isso sendo feito por telefone.

Por isso que o projeto tem duas partes, uma para o paciente colocar seus dados no banco, a partir de outra aplicação *web*, e outra para o funcionário pegar esses dados e conseguir marcar a consulta com mais rapidez e eficiência, podendo marcar mais pacientes, e dedicar seu tempo em alguma outra área também. Assim como o paciente não terá o estresse das TFD visto no tópico 2.1.

Portanto o entendimento da planilha foi essencial para a produção, e possível automação de um processo repetitivo e tendencioso.

3.2. Ferramentas

As ferramentas que serão usadas, já foram ditas no tópico 2, mas neste subcapítulo será organizado onde cada uma será alocada.

3.2.1. Backend

Para o processo que controla as funcionalidades por trás das aplicações, serão usadas as ferramentas *Node.js* para gerar o servidor, junto com o *Express* para criar e organizar as rotas facilmente, dessa maneira podemos afirmar que o *JavaScript* é o *backend* da aplicação projetada.

3.2.2. Frontend

Como o *frontend* é a parte que interage diretamente com a interface visualizada pelos usuários, as ferramentas utilizadas para compor esse trabalho são

as *Handlebars* e *Bootstrap*, a primeira sendo utilizada para conexão do *backend* com o *frontend*, além de ajudar na criação das páginas, onde a segunda ferramenta normaliza e faz a maior parte da criação da UI.

3.2.3. Banco de Dados e *Containers*

Como já dito anteriormente, para harmonizar com a aplicação relacional orientada à objetos, a linguagem escolhida para o armazenamento dos dados foi a SQL, usando o gerenciador MySQL. Também foi utilizado o *Sequelize* para facilitar a comunicação entre a aplicação e o banco de dados.

A infraestrutura da produção da aplicação, juntamente com o banco de dados, será utilizado o *Docker* para formar *container* do banco de dados e um *backup* juntos, em outro será armazenada a aplicação, e entre elas haverá uma rede para conexão e conversa dos *containers*. Tudo isso será implementado em um servidor físico localizado na diretoria da UPE, provisoriamente.

3.3. Metodologia

Para o entendimento e planejamento da solução do problema, foi utilizado a metodologia científica, de forma que foi feita a observação do problema, as hipóteses para solução, implicações e experimentos. Juntamente com a metodologia ágil para a produção da aplicação *web*, pois além de ser o que as empresas usam no mercado de trabalho, é uma forma de produção de uma aplicação melhor e mais aceita pelo cliente (NUTES-CISAM), pois o *feedback* do produto é mais rápido e preciso.

Então foi pensado na solução, depois foi feito os modelos do banco de dados e quais tabelas e atributos iria conter baseado na planilha que é utilizada atualmente, em seguida foi feita a ideia das páginas utilizando a ferramenta *Figma*⁵, um processo muito importante para o desenvolvimento do *frontend* e o *backend*, depois da aprovação do cliente, a produção do aplicativo começou, passando por vários alinhamentos que aconteciam semanalmente, para parecer com um processo ágil de desenvolvimento, aprovações e erros.

Com o *frontend* e *backend* da aplicação feita, o projeto avançou para uma fase de testes e implementação infraestrutural no servidor da UPE, sendo feito pela

⁵ <https://www.figma.com/>

ferramenta dita anteriormente, o *Docker*, por sua habilidade de containerização e simplicidade de controle de manutenção.

A aplicação usará as quatro operações básicas do CRUD: o *Create*, *Read*, *Update* e *Delete*, pois serão criadas as consultas dos pacientes, irá ver quem são os pacientes, de acordo com a qualificação do funcionário do CISAM poderá atualizar dados cadastrais do paciente e cancelar consultas, dessa forma deletando o dado de consulta. Essas funções só poderão ser feitas a partir da comunicação REST que disponibiliza ações HTTP - *GET*, *POST* E *DELETE* - acionadas por meio de URIs, que realizarão as operações CRUD informadas.

Só tem três operações HTTP porque o *POST* serve tanto para criar quanto para atualizar dados, o *GET* é a função de leitura do dado requisitado, podendo voltar todos os dados de uma tabela específica ou um dado específico, enquanto o *DELETE* serve para deletar os dados específicos ou todos os dados, que é algo perigoso, então não existirá a função de deletar tudo.

Para ser mais específico, serão feitas as seguintes operações:

- `app.get('/')`;
- `app.get('/home')`;
- `app.get('/pacientes')`;
- `app.post('/editar')`;
- `app.post('/marcar-consulta')`;
- `app.get('/resumo-paciente')`;
- `app.delete('/deletar-consulta')`;

A utilização do termo *app*, é em função da biblioteca *express.js* que serve para demonstrar que o sistema de software é uma aplicação *web*. Também se observa as rotas URIs que fazem a ligação com as funções do CRUD, assim como as operações HTTP.

4. Resultados

Agora será mostrado o desenvolvimento da aplicação utilizando as metodologias mostradas na seção 3.3, aprofundando na escolha do design, aprovação da modelagem dos dados, a produção do *backend* com as criações de rotas e limites de manipulação, o *frontend* ligando com o *backend* e a produção de *containers*, juntamente com as ligações a serem implementados ao servidor.

4.1. Análise e Design

Sabendo que o objetivo do projeto é trocar a planilha que o funcionário do NUTES-CISAM tem que preencher, a primeira análise feita, foi a observação do trabalho atual para marcar uma consulta. Observado a ação, foi idealizado dois projetos para satisfazer as necessidades da planilha, o primeiro que não foi abordado no documento, é o de registro do paciente no banco de dados do NUTES, podendo ser feito por um *site* que vai substituir o atual, assim, não precisando ter um funcionário à disposição para fornecer os dados por telefone.

O outro projeto (foco deste documento), é a utilização da tabela pacientes, para mostrar aos funcionários, quais as pessoas que esperam por uma consulta a ser marcada. Dessa maneira foi feito um modelo lógico do banco de dados, para saber quais atributos cada tabela possuiria, e quais os atores mais importantes para a situação.

Dessa maneira, na Figura 5, se tem em disposição o modelo mencionado anteriormente, nele é possível observar que o foco principal é a consulta, o paciente é o segundo foco, pois não haveria consulta sem o paciente, assim como o médico, mas nesse caso foi colocado apenas a especialidade médica, pois a disponibilidade dos médicos é feita em outro software que não pôde ser configurado ou manipulado, o funcionário precisa consultar por fora.

Pode-se ver também que há a tabela de usuário, na qual é relacionada ao funcionário que marcará as consultas, é necessário para a gerência e controle de quem está marcando as consultas, e caso aconteça algo inesperado, um erro com a consulta, os coordenadores poderão ter acesso ao funcionário que marcou a consulta.

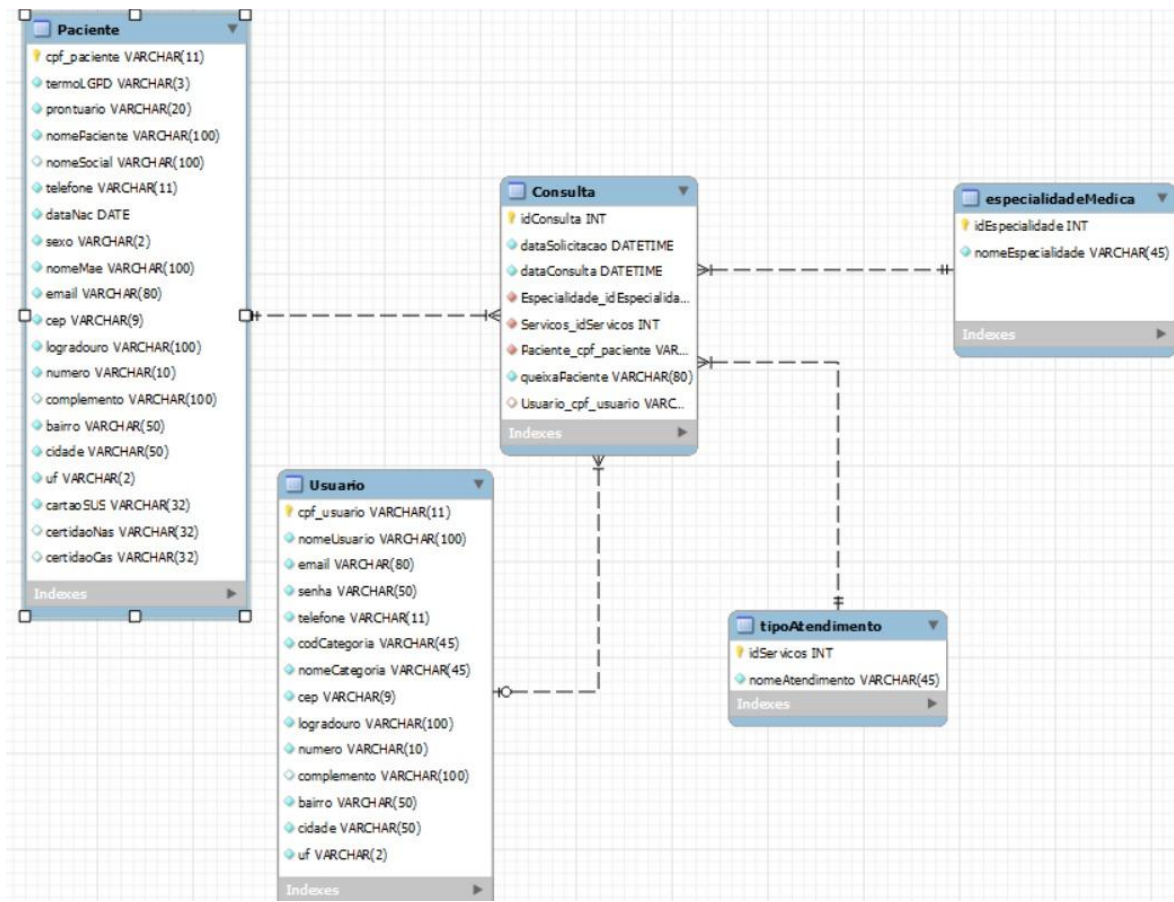


Figura 5. Modelo Lógico do Banco de Dados para Marcar Consultas.

Também é possível observar que são poucos os atributos que podem ser nulos, mas isso pode mudar no futuro, pois essa modelagem foi feita se baseando no caminho ideal, na qual a pessoa tem todas as informações necessárias. A mudança que pode acontecer com maior probabilidade é o fato do paciente não possuir um prontuário, logo, alterando o atributo à aceitar valores nulos, mas isso acarretaria em outras mudanças nas regras de negócio.

4.2. Desenvolvimento o *Frontend*

Com o modelo lógico feito e aprovado, o design da aplicação *web* foi desenvolvida utilizando a ferramenta *Figma*. Foram feitas 4 páginas base para o projeto, nas Figuras 6, 7, 8, 9 e 10, destacam-se as imagens. Na Figura 6 é possível observar a página de *login*, contendo um cabeçalho, remetendo a possibilidade de voltar ao *site* aberto do NUTES-CISAM, assim, a ideia de que a área do funcionário marcar consulta, fizesse parte do *site* aberto às pessoas.

Figura 6. Página de *login* pensada no *Figma*.

Identifique-se no NUTES CISAM com:

CPF
Digite seu CPF

Senha
Digite sua Senha

Entrar

[Esqueceu a Senha?](#)

Contato
(81) 3182-7780
telessaude.cisam@upe.br

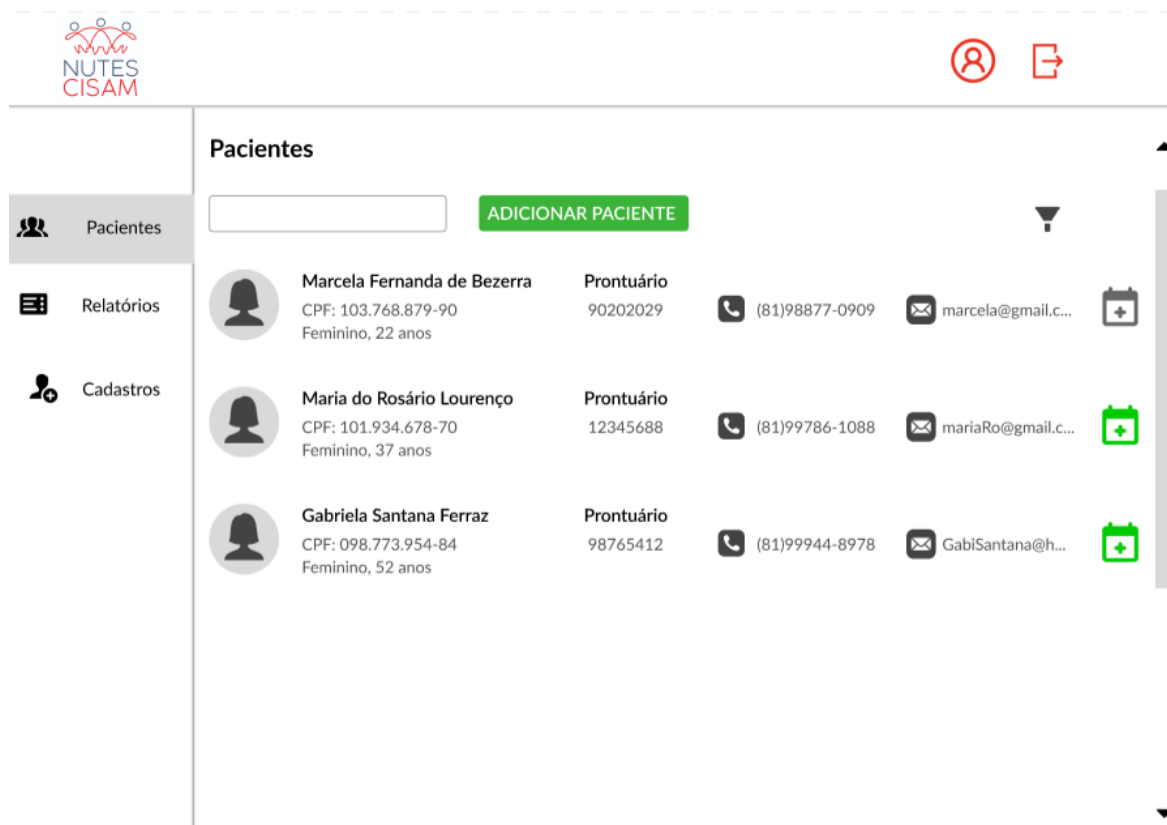
Redes sociais
f i t o

Nutes Teleassistência Teleconsulta Eventos Notícias Contato

Fonte: De autoria própria.

A Figura 7 mostra a página que aparece logo depois da autenticação do funcionário, onde é possível observar várias possíveis funções, por exemplo:

- Sair da sessão (*logout*);
- Abrir o perfil do funcionário;
- Acessar outras páginas de interesse;
- Observar os pacientes;
- Filtrar os pacientes;
- Adicionar paciente (função prevista só para administradores);
- Marcar consulta do paciente;
- Clicar no paciente para obter mais informações;
- Verificar quais pacientes ainda tem consulta a ser marcada.

Figura 7. Página dos pacientes pensada no *Figma*.

Fonte: De autoria própria.

Complementando a Figura 7, a Figura 8 mostra a caixa de texto que surge após selecionar a opção do calendário possível de marcar consulta, representado pela cor verde ao lado do paciente. Nesta caixa de texto é preciso preencher os campos com informações necessárias para os funcionários saberem qual dia e hora colocar, é preciso consultar uma outra tabela, que não está envolvida com o banco do projeto deste documento, por regra de negócio que é mantido dessa maneira.

Depois de ter consultado a tabela de disponibilidade médica e preenchido as informações, o funcionário poderá salvar essa consulta no banco de dados, mais precisamente a tabela de consultas. Quando confirmada a validação feita pelo *backend*, aparecerá uma caixa de texto informando sobre o sucesso da marcação de consulta.

Figura 8. Caixa para marcar consulta e consulta confirmada.

DEFINA A DATA DA CONSULTA:

Dia Mês Ano

12 Novembro ▼ 2022

Horário

08:00 ▼

Cancelar **SALVAR**

Email enviado com sucesso!

✓

Fonte: De autoria própria.

Logo a Figura 9 mostra como é mostrado os detalhes do paciente, clicando no nome ou imagem que aparece na lista da Figura 7, nela há várias seções com informações do paciente escolhido, assim como últimos atendimentos, tendo três marcações, consultas feitas, consultas canceladas e consulta marcada. Há também uma seção destinada à serviços consumados, ainda não tem uma definição sobre a regra do negócio desses serviços, pois o CISAM faz parte do Sistema Único de Saúde (SUS), logo, ter uma seção de serviços baseado em cobrança ainda está para definir.

À direita há dois botões que levam à outras páginas, um para marcar consultas, logo outro método de marcação de consultas, mas também tem outro botão que foi retirado de produção, o botão de prontuário, por causa de outra regra de negócio onde a edição e criação de prontuário é por outro sistema que não pode ser implementado ao projeto.

Figura 9. Página sobre os detalhes do paciente escolhido.

NUTES CISAM

Marcela Fernanda de Bezerra **Prontuário** 90202029 **MARCAR CONSULTA**
 CPF: 103.768.879-90 **PRONTUÁRIO**
 Feminino, 22 anos
 (81)98877-0909 marcela@gmail.com

RESUMO **DADOS CADASTRAIS**

Últimos Atendimentos

Status	Data	Horário	Médico	Especialidade	Plano
●	31/05/2022	08:00	Laura Olinda Bregieiro Fer...	Ginecologista e Obstetra	SUS
●	03/12/2021	13:00	Simone Angélica Leite de ...	Ginecologista e Obstetra	SUS
✓	15/10/2021	08:40	Simone Angélica Leite de ...	Ginecologista e Obstetra	SUS

<< 1 2 ... 13 >>

Últimos Serviços

Situação	Data	Serviço	Convênio	Valor
Aguardando gerar Lote	31/05/2022	Consulta	SUS	R\$ 0,00
Aguardando gerar Lote	03/12/2021	Consulta	SUS	R\$ 0,00
Aguardando gerar Lote	15/10/2021	Consulta	SUS	R\$ 0,00

<< 1 2 ... 13 >>

Fonte: De autoria própria.


A edição dos dados do paciente é feita na tela de edição, representada pela Figura 10, na qual só quem tem permissão de administrador pode alterar esses dados, isso será identificado por um atributo na tabela de usuário no banco de dados, um atributo binário para identificar se o funcionário é ou não um administrador. Percebe-se que a página de edição é uma aba da página de detalhes do paciente, mostrado na Figura 9, é possível identificar duas abas, sendo uma de



resumo, outra com dados cadastrais, aí será possível modificar os dados do paciente escolhido.

Depois de modificado e verificado as informações, aperta-se o botão de salvar, concluindo a ação e da mesma forma que em marcação de consultas, quando se salva dados, aparece uma caixa informando ao funcionário que a mudança foi deferida e já está salvo no banco de dados.


Essas são as páginas pensadas para o projeto, mas na aba da esquerda é possível ver que ainda há duas páginas que não foram mostradas, relatórios e cadastros, elas foram apenas mencionadas, sem muito foco com regras de negócio, e o motivo maior de não desenvolvê-las, é por causa do interesse maior no momento, fazer o sistema para marcar consultas e colocar no ar.

Figura 10. Página para editar dados do paciente e caixa de sucesso de edição.








←

 **Marcela Fernanda de Bezerra**

CPF: 103.768.879-90

Feminino, 22 anos

 (81)98877-0909  marcela@gmail.com

Prontuário

90202029

MARCAR CONSULTA

PRONTUÁRIO

RESUMO

DADOS CADASTRAIS

INFORMAÇÕES DO PACIENTE

Nome	Nome da Mãe
<input type="text" value="Marcela Fernanda de Bezerra"/>	<input type="text" value="Catarina de Bezerra Fonseca"/>
CPF	Data de nascimento
<input type="text" value="103.768.879-90"/>	<input type="text" value="30/04/1990"/>
Idade gestacional ao nascer	Sexo
<input type="text" value="Semanas"/>	Feminino ▼
Email	Celular
<input type="text" value="marcelafbezerra@upe.br"/>	+55 <input type="text" value="(81) 98877-0909"/>
Observações	
<input type="text" value="Paciente teste"/>	

ENDEREÇO

CEP	Rua	Número
<input type="text" value="52121-321"/>	<input type="text" value="Rua"/>	<input type="text" value="99"/>
Cidade	Bairro	Complemento
<input type="text" value="Recife"/>	<input type="text" value="Bairro"/>	<input type="text" value="A"/>
		Estado
		Pernambuco ▼

CARTEIRA DO PACIENTE

Convênio	Número Carteira	Validade
<input type="text" value="SUS"/>	<input type="text" value="1234567"/>	<input type="text" value="00/00/0000"/>

Cancelar **SALVAR**

Dados atualizados com sucesso!



OK

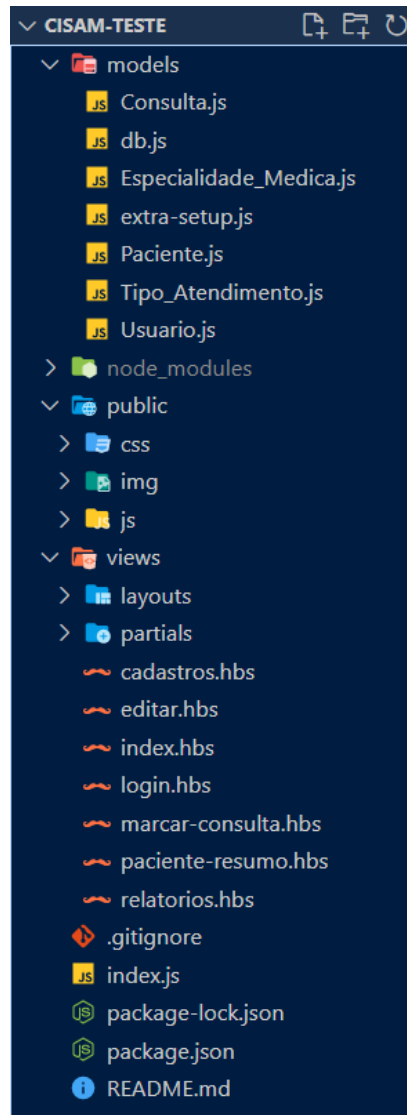
Fonte: De autoria própria.

4.3. O Backend

A parte lógica do projeto foi desenvolvida com base nas ideias das telas mostradas anteriormente, primeiramente foi desenvolvida a organização das pastas no projeto, exposta na Figura 11. É possível analisar que a arquitetura MVC está sendo bem aplicada, onde a pasta de *models* contém os modelos relacionados ao banco de dados e o funcionamento deles, sendo *db.js*, o arquivo de configuração geral do banco de dados.

A pasta *views*, armazena as páginas que aparecem na tela do usuário, nesse caso, o funcionário do NUTES-CISAM, como é usado a ferramenta *Handlebars*, é possível fazer cabeçalhos, rodapés e barra de navegação, sem repetição de código, utilizando *layouts*. Como a entrega principal do projeto é pequena em relação a outros projetos de aplicação *web*, a parte de *controller*, ficou junta com a página principal de funcionamento de projetos em *Node.js*, no arquivo *index.js*, nela encontra-se vários caminhos e requisições de todo o projeto.

Figura 11. Disposição das pastas do projeto.



Fonte: De autoria própria.

Nas Figuras 12 e 13, é possível observar como ficou a página de *login* e a da listagem de pacientes, nota-se que a utilização da ferramenta *Bootstrap*, consegue organizar e normalizar certos aspectos do desenvolvimento, como estilos dos botões, fonte das letras e disposição dos itens na página. A página de *login* ficou mais simples, pelo fato de não ser integrada ao novo *site* do NUTES-CISAM. A listagem dos pacientes foi retirada a foto da pessoa, para conservar mais a integridade e segurança do paciente, também pelo fato do sistema não requer o armazenamento de fotos, na qual poderia ocupar muito espaço no banco de dados.

Figura 12. Página de *login* produzida.




Login

Fonte: De autoria própria.

A página apresentada na Figura 13 demonstra a função de deletar o paciente, caso seja necessário, por motivo do tempo de produção, foi retirado outras informações sobre os pacientes, só deixando o essencial, o número de prontuário, nome e cpf, para obter mais informações, é necessário ver os detalhes do paciente.

Figura 13. Página de pacientes produzida.



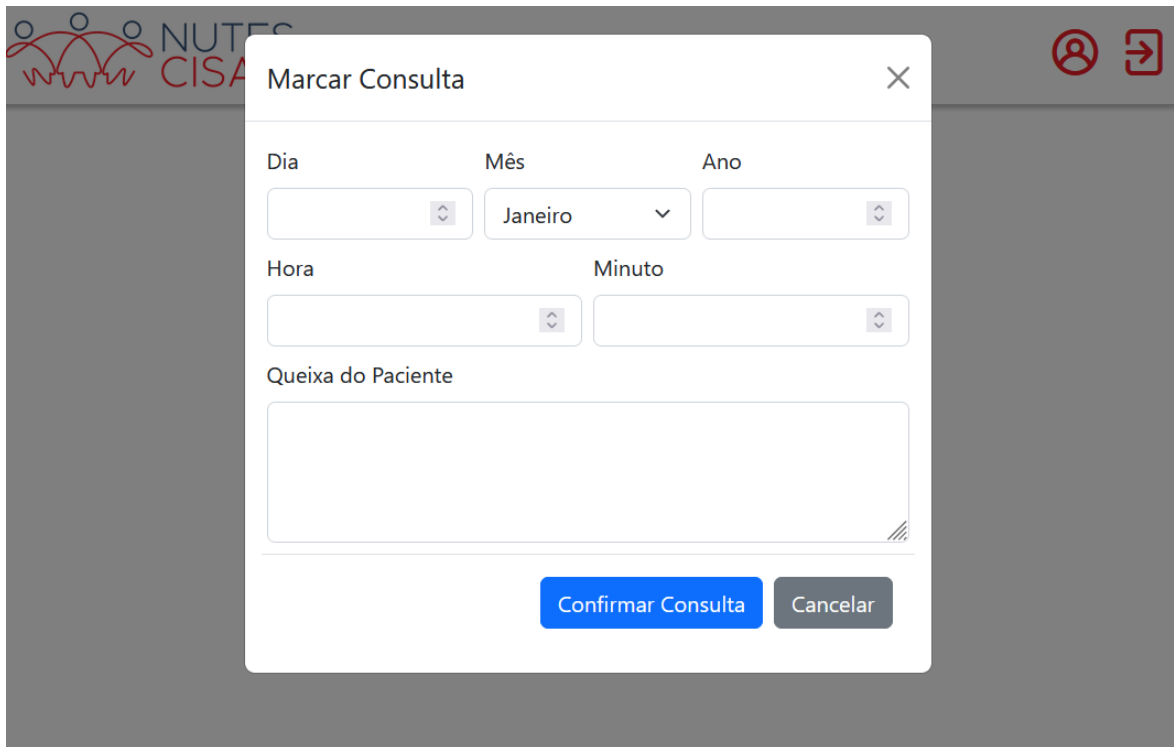
Pacientes		Nome	CPF	Ações
90202029	Marcela Fernanda de Bezerra	10376887990	<input type="button" value="Editar"/> <input type="button" value="Deletar"/> <input type="button" value="Calendário"/>	
12345678	Roberto Levi Silveira	13893980210	<input type="button" value="Editar"/> <input type="button" value="Deletar"/> <input type="button" value="Calendário"/>	
65412398	Andrea Marlene Fogaça	19106870490	<input type="button" value="Editar"/> <input type="button" value="Deletar"/> <input type="button" value="Calendário"/>	
21452129	Lucas Otávio João Brito	74966343798	<input type="button" value="Editar"/> <input type="button" value="Deletar"/> <input type="button" value="Calendário"/>	
49635703	Ayla Louise Milena Aragão	78256997427	<input type="button" value="Editar"/> <input type="button" value="Deletar"/> <input type="button" value="Calendário"/>	
87654321	Cauã Cauã Augusto Porto	90844608475	<input type="button" value="Editar"/> <input type="button" value="Deletar"/> <input type="button" value="Calendário"/>	

Fonte: De autoria própria.

Ao apertar o botão do calendário ao lado do paciente mostrado na Figura 13, aparece a caixa para marcar a consulta do paciente, exibido na Figura 14. Nela

pode-se acompanhar a escolha do dia, mês, ano, hora, minuto e a queixa que o paciente sente.

Figura 14. Caixa para marcação de consulta, por paciente.



A imagem mostra uma interface de usuário para marcar uma consulta. No topo, há um cabeçalho com o texto "NUTEC CISA" e ícones de perfil e seta. O formulário principal, intitulado "Marcar Consulta", contém os seguintes elementos:

- Campos de seleção para "Dia", "Mês" (com "Janeiro" selecionado) e "Ano".
- Campos de seleção para "Hora" e "Minuto".
- Um campo de texto para "Queixa do Paciente".
- Dois botões de ação: "Confirmar Consulta" (em azul) e "Cancelar" (em cinza).

Fonte: De autoria própria.

Figura 15. Página de edição dos dados de paciente produzida.

NUTES CISAM

Marcela Fernanda de Bezerra Prontuário: 90202029 CPF: 10376887990

MARCAR CONSULTA
PRONTUÁRIO

DADOS CADASTRAIS RESUMO

INFORMAÇÕES DO PACIENTE

Nome: Marcela Fernanda de Bezerra Nome da Mãe: Marcela Bezerra Peixoto

CPF: 10376887990 Data de Nascimento: 08 / 09 / 2000

Idade Gestacional: Digite a idade gestacional Sexo: Feminino

Email: marcela@gmail.com Celular: 81988770909

Observações: Digite as Observações

ENDEREÇO

CEP: 53020555 Rua: Rua bla, Bairro ble, Cidade1, PE

Número: 800 Cidade: Cidade1

Bairro: Bairro ble Complemento:

Estado: PE

CARTEIRA DO PACIENTE

Convênio: SUS Número Carteira: 123456789101112

Validade: 20/08/2022

SALVAR Cancelar

Fonte: De autoria própria.

Na Figura 15 tem-se a página de edição do paciente, observa-se que atualmente é possível modificar todos os dados referentes à paciente, tudo é dividido em seções para melhor interpretação de quem estiver modificando.

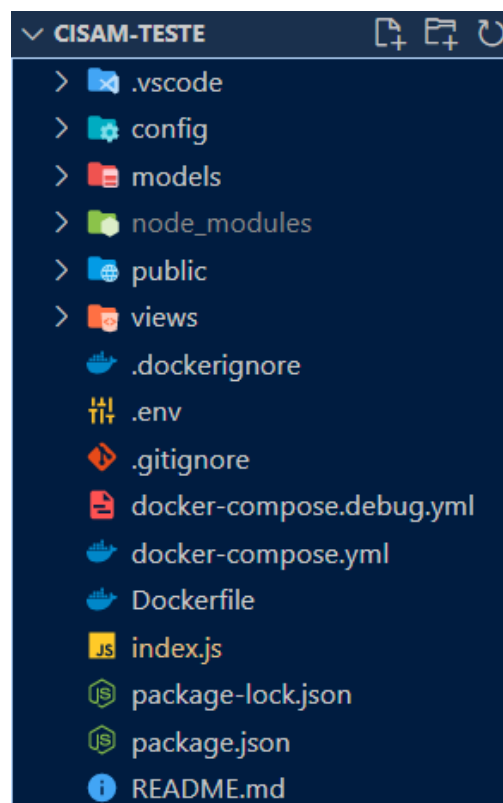
Todas as páginas precisam de uma rota para o servidor fornecer as páginas e as informações delas, há pelo menos 8 rotas disponíveis, sendo 3 dessas rotas ligadas à função *POST*, logo, são rotas para mudança de pacientes, exclusão de pacientes e a de marcação de consultas, outras rotas utilizam a função *GET*, pois só é de requisição para mostrar a página ao usuário.

Existem dois tipos de validação dos dados escritos pelos usuários, há a validação feita no *frontend*, utilizando o HTML de forma correta para cada tipo de dados, mas é possível burlar isso alterando no navegador da pessoa, trocando o tipo de dado recebido por outro qualquer, pensando nisso, foi feita uma validação no próprio *backend* para cada tipo dado que precisasse de validação, por isso, mesmo que o usuário tente burlar o tipo de dado, modificando o HTML no navegador, a segunda validação não deixará que o dado passe para o banco de dados.

4.4. Implementação dos *Containers*

Com o intuito de implementar o projeto no servidor da UPE, utilizando a ferramenta *Docker*, que transforma a aplicação em *containers*, dois para ser mais preciso: o *container* do banco de dados e outro para a aplicação. É preciso modificar um pouco a disposição das pastas, e adicionar outros arquivos para configuração e *containerização*. A Figura 16, mostra como ficou a disposição final para produção.

Figura 16. Disposição final das pastas.



Fonte: De autoria própria.

Um dos arquivos mais importantes para a produção de *containers*, é o *Dockerfile*, dependendo do projeto, pode haver mais de um *Dockerfile*, pois ele é responsável pelo tipo do *container* que vai ser criado, na Figura 17, é possível ver a

versão do *Node.js* sendo utilizado, a organização das pastas é muito importante, pois o que o *container* vai criar se baseia no conteúdo das pastas, na Figura 17, é preciso informar qual o diretório do projeto.

Ademais, é necessário copiar o arquivo de dependências, mostrado na Figura 18, é preciso executar os comandos para o funcionamento, o *npm install* serve para instalar as dependências que o arquivo *package.json* informa, o próximo passo é copiar todo o diretório da aplicação, e por fim, inicializar o projeto, utilizando a função *npm start*.

Figura 17. *Dockerfile* escrito para aplicação.

```
FROM node:14
WORKDIR /
COPY package.json .
RUN npm install
COPY . .
CMD npm start
```

Fonte: De autoria própria.

O conteúdo do arquivo *package.json*, mostrado na Figura 18, é possível constatar a seção priorizada para dependências do projeto, incluindo o nome da ferramenta utilizada e a versão dela, o acento circunflexo antes do número de versão, significa que o projeto suporta as versões superiores que a mostrada.

Também consta que há uma organização do projeto no arquivo, mostrando o nome do projeto (importante, pois pode mudar o nome do *container* quando for criado), versão, arquivo principal, autor e *scripts*, que servem para a aplicação e funcionamento do projeto, a função vista anteriormente, *npm start*, busca neste arquivo a interpretação.

Figura 18. Arquivo de pacotes de dependências do *Node.js* para a aplicação.

```
{
  "name": "Consulta-CISAM",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "author": "Davi Nogueira",
  "license": "ISC",
  "dependencies": {
    "ember-truth-helpers": "^3.1.1",
    "express": "^4.18.1",
    "express-handlebars": "^6.0.6",
    "mysql2": "^2.3.3",
    "nodemon": "^2.0.19",
    "sequelize": "^6.21.3",
    "dotenv": "^10.0.0"
  }
}
```

Fonte: De autoria própria.

A Figura 19 mostra o arquivo mais importante para a produção de *containers*, o *docker compose*, nele nota-se que é um arquivo extenso contendo versão do *Docker Compose*, serviços, que cada um é uma imagem isolada para se formar um *container*, agregando esses serviços e por último o *volumes* é responsável por manter os dados ativos, se a aplicação precise reiniciar.

Há várias funções para a criação dos serviços, é possível fazer dois *Dockerfiles*: um para o banco de dados e outro para a aplicação *web*, mas nesse caso não foi preciso, pois a própria ferramenta *Docker*, tem imagens online que podem ser usadas, tem a função para a aplicação de restart do serviço, o ambiente, localizado no arquivo *.env*, no qual o conteúdo é mostrado na Figura 20, as portas para conexão do serviço em *container*. Dependendo da diferença do tipo de serviço, podem haver diferentes tipos de funções.

Figura 19. Arquivo *Docker compose*, necessário para criar mais de um *container* de uma vez.

```
version: '3.8'

services:
  mysql:
    image: mysql:5.7
    restart: unless-stopped
    env_file: ./env
    environment:
      - MYSQL_ROOT_PASSWORD=$MYSQLDB_ROOT_PASSWORD
      - MYSQL_DATABASE=$MYSQLDB_DATABASE
    ports:
      - $MYSQLDB_LOCAL_PORT:$MYSQLDB_DOCKER_PORT
    volumes:
      - db:/var/lib/mysql
  teste2:
    image: teste2
    depends_on:
      - mysql
    build:
      context: .
      dockerfile: ./Dockerfile
    restart: unless-stopped
    env_file: ./env
    ports:
      - $NODE_LOCAL_PORT:$NODE_DOCKER_PORT
    environment:
      - DB_HOST=mysql
      - DB_USER=$MYSQLDB_USER
      - DB_PASSWORD=$MYSQLDB_ROOT_PASSWORD
      - DB_NAME=$MYSQLDB_DATABASE
      - DB_PORT=$MYSQLDB_DOCKER_PORT
    stdin_open: true
    tty: true

volumes:
  mysql:
```

Fonte: De autoria própria.

Figura 20. Conteúdo do arquivo `.env`, para organizar variáveis.

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=123456
DB_NAME=cisam
DB_PORT=3306

MYSQLDB_USER=root
MYSQLDB_ROOT_PASSWORD=123456
MYSQLDB_DATABASE=cisam
MYSQLDB_LOCAL_PORT=3307
MYSQLDB_DOCKER_PORT=3306
NODE_LOCAL_PORT=6868
NODE_DOCKER_PORT=8080
```

Fonte: De autoria própria.

O arquivo `.env`, é comparado à utilização de *Enums*, pois são variáveis que podem ser utilizadas globalmente pelo projeto, desde que esclareça a dependência da ferramenta e arquivo `.env`, e observa-se o quão é importante a decisão de usar essa ferramenta de organização, pois as Figuras 18, 19 e o `index.js`, precisam modificar seus arquivos para suportar as variáveis.

5. Conclusão e Trabalhos Futuros

Com o que foi expresso neste trabalho, é possível compreender a gravidade da situação que os pacientes do CISAM enfrentam, ter que sair de suas residências e enfrentar obstáculos para receber tão pouco de volta. Uma simples aplicação *web*, utilizando ferramentas usadas em abundância por empresas do mercado de trabalho, consegue melhorar a vida dos pacientes assim como dos funcionários do NUTES-CISAM.

Esse projeto faz parte de uma reestruturação completa do NUTES, por isso, pode ser considerado um trabalho completo de *fullstack*, pois foi feito o *frontend* assim como o *backend*, além da coleta de informações para se construir o projeto, também inclui-se o trabalho de pós-produção, de como será implementada a solução exposta. O maior objetivo do projeto é a listagem de pacientes, pois já ajuda o funcionário do NUTES-CISAM, com o tempo que passaria no telefone, e o objetivo de marcar consultas dos pacientes.

Com a utilização de ferramentas de controle e organização, é possível uma escalabilidade imensa, pois arquivos de configurações estão localizados em arquivos particulares. É esperado uma melhoria no futuro, pois ainda tem implementação de outras páginas a ser feita, assim como adicionar esta aplicação no site aberto do NUTES-CISAM.

Referências

- [1] UPE. **CISAM - Centro Universitário Integrado de Saúde Amaury de Medeiros**. Disponível em: <<http://www.upe.br/uh-cisam.html>> Acesso em: 26 de setembro de 2022.
- [2] NUTES-CISAM. **Quem Somos**. Disponível em: <<https://sites.google.com/upe.br/nutes-cisam/#h.1easpem0wr9p>> Acesso em: 26 de setembro de 2022.
- [3] HARALD G.; MEHDI J.; RENÉ R.; GEORG T. **Software Evolution Observations based on product release history**. *Proceedings of the International Conference on Software Maintenance (ICSM'97)*, 1997.
- [4] B.M. ALKMIM, Maria; S. MARCOLINO, Milena. **Factors Associated with the Use of a Teleconsultation System in Brazilian Primary Care**. *Revista de Telemedicine and e-Health*, v.21, n.6, jun. 2015.
- [5] BAI, Jing; ZHANG, Yonghong; DAI, Bing. **Design and development of an interactive medical teleconsultation system over the World Wide Web**. Em *IEEE Transactions on Information Technology in Biomedicine*, vol. 2, no. 2, p. 74-79, Junho 1998.
- [6] DIAS VIEIRA, Anaclaudia; APARECIDA NERY, Maria. **O programa de tratamento fora de domicílio - TFD e os desafios para o acesso pelos usuários do SUS numa perspectiva de direito**. Em: III Congresso de Serviço Social do IMIP, IV Jornada de Serviço Social do IMIP, Volume 1, 2017, Recife. p. 75-77
- [7] A. Mah, Bruce. **An Empirical Model of HTTP Network Traffic**. *Proceedings of INFOCOM '97, Kobe-Japão*, vol.2, p. 592-600, abril. 1997.
- [8] ETF Datatracker. **Hypertext Transfer Protocol -- HTTP/1.1**. Disponível em: <<https://datatracker.ietf.org/doc/html/rfc2616>>. Acesso em: 28 de setembro de 2022.
- [9] IETF Datatracker. **Constrained RESTful Environments (CoRE) Link Format**. Disponível em: <<https://datatracker.ietf.org/doc/html/rfc6690>>. Acesso em: 28 de setembro de 2022.

-
- [10] Nordic APIs. **CRUD vs. REST: What's the Difference?**. Disponível em: <<https://nordicapis.com/crud-vs-rest-whats-the-difference/>>. Acesso em: 28 de setembro de 2022.
- [11] John Deacon. **Model-View-Controller Architecture**. Disponível em: <<http://www.johndeacon.net/john-deacon/articles/model-view-controller-architecture/>>. Acesso em: 29 de setembro de 2022.
- [12] Berkeley Extension. **11 Most In-Demand Programming Languages in 2022**. Disponível em: <<https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/>>. Acesso em: 29 de setembro de 2022.
- [13] Simplilearn. **Most Popular Programming Languages to Learn in 2022**. Disponível em: <<https://www.simplilearn.com/best-programming-languages-start-learning-today-article>>. Acesso em: 29 de setembro de 2022.
- [14] Sociedade Brasileira de Computação, Congresso. **Anais**, Volumes 2-3. Texas, Estados Unidos: A Sociedade, 1993. p.5.
- [15] Node.js. **About Node.js**. Disponível em: <<https://nodejs.org/en/about/>>. Acesso em: 29 de setembro de 2022.
- [16] DB-Engines. **DB-Engines Ranking**. Disponível em: <<https://db-engines.com/en/ranking>>. Acesso em: 30 de setembro de 2022.
- [17] Docker. **Use containers to Build, Share and Run your applications**. Disponível em: <<https://www.docker.com/resources/what-container/>>. Acesso em: 30 de setembro de 2022.

Anexo A

Anexo Código Fonte

Link para o código fonte:

<https://github.com/Davi-Mota-Nogueira/Trabalho-Consulao-Curso>