UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE GRADUAÇÃO EM BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

MATHEUS POSSIDÔNIO GONÇALVES VIEIRA COSTA

DESENVOLVIMENTO BACK-END DE SISTEMA WEB PARA BUSCA DE SERVIÇOS DE SAÚDE

CAMPINA GRANDE
2024

UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO GRADUAÇÃO EM BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

MATHEUS POSSIDÔNIO GONÇALVES VIEIRA COSTA

DESENVOLVIMENTO BACK-END DE SISTEMA WEB PARA BUSCA DE SERVIÇOS DE SAÚDE

Relatório de Conclusão de curso apresentado ao Curso de ciências da computação do Centro de ciências e tecnologias da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em ciências da computação.

Orientador: Prof. Dr. Daniel Scherer.

CAMPINA GRANDE
2024

C837d Costa, Matheus Possidônio Gonçalves Vieira.

Desenvolvimento back-end de sistema web para busca de serviços de saúde [manuscrito] / Matheus Possidonio Goncalves Vieira Costa. - 2024.

67 p. : il. colorido.

Digitado. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2024. "Orientação : Prof. Dr. Daniel Scherer, Coordenação do Curso de Computação - CCT. "

1. Sistema de saúde. 2. Spring Boot. 3. Micro serviços. I. Título

21. ed. CDD 005

Elaborada por Lêda C. D. Andrade - CRB - 15/1032 BC/UEPB

MATHEUS POSSIDÔNIO GONÇALVES VIEIRA COSTA

DESENVOLVIMENTO BACK-END DE SISTEMA WEB PARA BUSCA DE SERVIÇOS DE SAÚDE

Relatório de Conclusão de curso apresentado ao Curso de ciências da computação do Centro de ciências e tecnologias da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em ciências da computação.

Área de concentração: Engenharia de Software

Aprovada em: 05 / Setembro / 2024.

BANCA EXAMINADORA

Prof. Dr. Daniel Scherer
(CCT/UEPB) Orientador(a)

Prof. Dra. Ana Isabella Muniz Leite
(CCT/UEPB) Examinador(a)

Profa. Dra. Katia Elizabete Galdino (CCT/UEPB)
Profa. (CCT/UEPB)(a)

AGRADECIMENTOS

Agradeço, inicialmente, a Deus pelas oportunidades de conquista e por cada dia ter a capacidade de viver neste mundo repleto de amor.

Agradeço à minha mãe, Elyzandra Gonçalves Costa, que, desde a minha infância, me forneceu a melhor educação e os melhores valores, fazendo-me sempre buscar o conhecimento da vida.

Agradeço a toda minha família e amigos, por todo o amor, conforto e fraternidade que me foi oferecido.

Agradeço a todos os meus amigos que sempre estiveram ao meu lado e me acompanharam nessa jornada acadêmica, incentivando uns aos outros constantemente.

Agradeço ao meu professor e orientador Daniel Scherer, pela paciência, dedicação e por ser um exemplo de cientista acadêmico.

Agradeço, de forma geral, à UEPB, por me proporcionar oportunidades de uma vivência acadêmica incrível e de grande aprendizado.

Agradeço a todas as amizades desenvolvidas nos diversos cursos da UEPB, sabendo que cada diálogo, aprendizado e experiência serviu para acrescentar o melhor em mim como ser humano.

“O verdadeiro encanto é a gentileza! Se torne uma boa pessoa. Só isso.”

(Arataka Reigen - ONE)

RESUMO

Este trabalho teve como objetivo o desenvolvimento de um sistema back-end para integração com um front-end já existente, voltado para o direcionamento de pacientes a unidades de saúde próximas, de acordo com suas necessidades específicas de atendimento. A problemática da superlotação dos hospitais centrais, muitas vezes resultante do atendimento de casos que poderiam ser resolvidos em unidades de saúde menos congestionadas, foi o ponto de partida para a elaboração desta solução tecnológica. Utilizando o framework Spring Boot, foi desenvolvida uma API robusta e escalável, capaz de gerenciar grandes volumes de dados e fornecer informações precisas e atualizadas sobre as unidades de saúde disponíveis. A metodologia aplicada envolveu uma análise detalhada das bases de dados de saúde, com o uso da linguagem Python para o tratamento, limpeza e organização dos dados. Esse processo foi essencial para garantir que o sistema pudesse operar com informações confiáveis, facilitando o acesso dos usuários a dados críticos para a tomada de decisão sobre onde buscar atendimento. Além disso, a arquitetura do sistema foi baseada em micro serviços, permitindo que cada componente funcione de forma independente, o que aumenta a flexibilidade, a manutenção e a escalabilidade do sistema. Os resultados obtidos indicam que o sistema desenvolvido tem o potencial de contribuir significativamente para a redistribuição mais eficiente dos pacientes, mitigando a superlotação dos grandes centros médicos e promovendo um uso mais racional dos recursos de saúde disponíveis. A integração bem-sucedida com o sistema front-end, previamente desenvolvido, demonstra a viabilidade técnica do projeto e sua aplicabilidade prática, oferecendo uma solução inovadora e eficaz para os desafios enfrentados pelo sistema de saúde pública no Brasil. Conclui-se que a aplicação de tecnologias modernas, como o Spring Boot, aliada a boas práticas de desenvolvimento de software e à análise de dados com Python, pode fornecer uma base sólida para o desenvolvimento de soluções que abordem problemas complexos, como a superlotação hospitalar e a acessibilidade aos serviços de saúde.

Palavras-Chave: sistema de saúde; Spring Boot; micro serviços.

ABSTRACT

This work aimed to develop a backend system for integration with an existing frontend, focused on directing patients to nearby healthcare facilities according to their specific medical needs. The issue of overcrowding in central hospitals, often caused by the treatment of cases that could be resolved in less congested healthcare units, served as the starting point for developing this technological solution. Using the Spring Boot framework, a robust and scalable API was developed, capable of managing large volumes of data and providing accurate and up-to-date information about available healthcare facilities. The methodology applied involved a detailed analysis of healthcare databases, utilizing the Python language for data processing, cleaning, and organization. This process was essential to ensure that the system operates with reliable information, facilitating user access to critical data for decision-making on where to seek care. Additionally, the system's architecture was based on microservices, allowing each component to function independently, which increases the system's flexibility, maintainability, and scalability. The results indicate that the developed system has the potential to significantly contribute to the more efficient redistribution of patients, mitigating the overcrowding of major medical centers and promoting a more rational use of available healthcare resources. The successful integration with the previously developed frontend system demonstrates the technical feasibility of the project and its practical applicability, offering an innovative and effective solution to the challenges faced by the public healthcare system in Brazil. It is concluded that the application of modern technologies, such as Spring Boot, combined with best software development practices and data analysis using Python, can provide a solid foundation for developing solutions that address complex problems, such as hospital overcrowding and access to healthcare services.

Keywords: healthcare system; Spring Boot; microservices.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE LISTAGENS

SUMÁRIO

1 INTRODUÇÃO

Observando o cenário de desenvolvimento de sistemas de softwares, é observado o foco de produção baseado em especificações e demandas, sendo esta, uma prática fundamental na engenharia de software, principalmente em ambientes mais complexos onde a precisão e eficiência são essenciais. A partir de especificações bem definidas, desenvolvedores são capazes de converter requisitos em sistemas executáveis, assim garantido que o produto final atenda o mais próximo possível de completude, se tratando das necessidades dos usuários e do mercado. Em sistemas web, essa abordagem se faz ainda mais relevante, pois a interação entre o front-end e back-end precisa ser orquestrada de forma mais cuidadosa possível, visando oferecer uma experiência fluida e eficaz.

O desenvolvimento de sistemas modernos web trata de um processo onde muitas vezes integra múltiplas camadas, tecnologias e práticas especializadas. Conforme destacado por Sommerville (2011), "o estágio de implementação do desenvolvimento de software é o processo de conversão de uma especificação do sistema em um sistema executável" (p. 25). Partindo dessa visão, temos o desenvolvimento de sistemas web nesse mesmo contexto, onde envolve a tradução cuidadosa das especificações, muitas vezes definidas a partir de análise de requisitos rigorosa ou de documentação detalhada, em soluções funcionais e práticas. Percorrendo esse processo não apenas é assegurado um sistema final que corresponda as expectativa dos clientes, mas também uma comunicação fluida e eficaz entre o back-end, que lida com o processamento e a lógica de dados, e o front-end , que serve como interface direta para o usuários assim proporcionando uma experiência intuitiva e interativa.

Neste cenário, frameworks como o Spring Framework e seu derivado, o Spring Boot, emergiram como soluções populares no desenvolvimento back-end devido à sua flexibilidade, escalabilidade e facilidade de configuração. O Spring Framework oferece uma ampla gama de funcionalidades que auxiliam no desenvolvimento de sistemas empresariais complexos, enquanto o Spring Boot automatiza grande parte da configuração inicial, permitindo que os desenvolvedores foquem diretamente na lógica de negócio. Segundo Weissman (2015), o Spring

reaproveita as tecnologias a fim de aumentar a produtividade do desenvolvedor, tratando-se de uma excelente ferramenta quando se fala de criação de sistemas que utilizam a arquitetura de microservices. Tais ferramentas têm sido amplamente adotadas por desenvolvedores devido à sua capacidade de lidar com grandes volumes de dados e facilitar a integração com outros sistemas.

Com o decorrer dos últimos anos, a arquitetura de microservices, definida principalmente por se basear em uma comunicação, quase sempre web, entre componentes altamente independentes, com isso ganhando destaque no desenvolvimento de sistemas distribuídos. A utilização dessa arquitetura permite em um sistema desenvolvido que diferentes partes, implementados em diferentes escalas, atuem de forma independente, e comunicando entre si utilizando protocolos de comunicação web, como por exemplo o protocolo REST e o JSON. Dessa forma é aumentada não só a flexibilidade e a resiliência do sistema. como também facilita a integração de novos serviços e tecnologias

O uso de sistemas web para melhorar a qualidade de vida da população é uma das aplicações mais impactantes da tecnologia moderna. Serviços web focados em saúde, por exemplo, têm o potencial de revolucionar o acesso a informações e serviços médicos, tornando-os mais acessíveis e eficientes para a população. No Brasil, um dos grandes desafios enfrentados é a superlotação dos hospitais, que muitas vezes se deve a atendimentos superficiais em centros médicos inadequados. Como mostrado por Uzelli(2013), temos a apresentação da problemática da necessidade de grandes deslocamentos até uma unidade de atendimento de saúde, o que muitas vezes pode dificultar que, principalmente a população de baixa renda, busque por atendimento. Essa situação é agravada por idas desnecessárias a hospitais, onde pacientes buscam atendimento em grandes centros médicos sem considerar unidades de saúde mais próximas e adequadas às suas necessidades.

Durante a busca pelos grandes centros médicos e sem um direcionamento adequado, muitas vezes os pacientes realizam um deslocamento de longas distâncias para ser atendido, contribuindo assim para uma ineficiência do sistema de saúde como um todo. A análise da problemática da superlotação hospitalar resultou então no desenvolvimento de um projeto de solução tecnológica visando mitigar esse desafio. O projeto “DESENVOLVIMENTO FRONT-END DE APLICATIVO PARA BUSCA DE SERVIÇOS DE SAÚDE”, por VENTURA(2022). O estudo apresenta o desenvolvimento de um sistema front-end, que visa facilitar o acesso da população

as unidades de saúde mais próximas, fornecendo informações claras e precisas sobre onde buscar atendimento médico de forma adequada e eficiente.

Partindo dessa contextualização, o presente trabalho visa dar continuidade ao projeto de VENTURA(2022). Com o sistema Front-end, previamente desenvolvido, é dado é proposto o desenvolvimento do sistema back-end, o mesmo será responsável por processar e gerenciar todos os dados necessários para o funcionamento pleno do sistema, enquanto o projeto de VENTURA(2022)já desenvolvido, apresenta essas informações de forma acessível e agradável ao usuário. Com esta integração é esperado não apenas a melhora na eficiência de acesso a serviços de saúde por parte da população em geral, como também oferecer uma solução escalável robusta que atenda às necessidades crescentes da população brasileira.

1.1 Objetivos

O principal objetivo deste trabalho é desenvolver um sistema back-end robusto e escalável que se integre de maneira eficiente ao front-end previamente desenvolvido, visando melhorar o acesso da população às unidades de saúde. Este back-end será responsável por processar, gerenciar e fornecer os dados necessários ao front-end, permitindo que os usuários encontrem de forma rápida e precisa as unidades de saúde mais adequadas às suas necessidades.

1.1.1 Objetivos Específicos

- Praticar conhecimentos adquiridos no curso acadêmico.
- Realizar a análise e o estudo das bases de dados de unidades de saúde.
- Analisar e implementar os requisitos funcionais e não funcionais necessários para a criação de um back-end.
- Desenvolver uma aplicação back-end capaz de atender com completude os requisitos fornecidos pelo sistema já implementado.
- Realizar a conexão entre os dois sistemas, back-end e front-end.

2 REVISÃO DE LITERATURA

Para o adequado desenvolvimento deste estudo, é necessário compreender os conceitos fundamentais que sustentam a análise. Neste capítulo, serão abordados os conceitos centrais que serviram de alicerce para todo o projeto. Tendo em vista que este trabalho visa dar continuidade ao estudo “DESENVOLVIMENTO FRONT-END DE APLICATIVO PARA BUSCA DE SERVIÇOS DE SAÚDE”, da autora VENTURA(2022). Foi observado a necessidade de dividir o foco da revisão em dois âmbitos distintos, onde o primeiro se deu pela análise do trabalho de VENTURA(2022) para assim adquirir argumentação e fundamentação acerca do assunto discutido, além de assim pontuar as necessidades fornecidas pelo sistema já existente. Em simultâneo a isso, foi realizada uma pesquisa em torno dos dados que serviriam para alimentar o sistema como um todo, tanto os tipos de dados necessários como suas fontes e validações.

A seguir, serão explorados de forma mais detalhada essas duas frentes de análise, a fundamentação teórica com base no estudo de VENTURA(2022), seguida pela investigação das fontes de dados necessárias para o funcionamento do sistema que atenda o projeto como um todo.

2.1 Análise de trabalhos anteriores

O trabalho desenvolvido por Daniele Ventura Batista, intitulado Desenvolvimento Front-End de Aplicativo para Busca de Serviços de Saúde (2022), proporciona um arcabouço fundamental para o desenvolvimento do back-end que dará continuidade ao sistema. VENTURA(2022) concentrou-se em criar uma interface amigável e acessível que permitisse aos usuários localizar unidades de saúde próximas, atendendo às suas necessidades específicas, com o objetivo de reduzir a superlotação em prontos-socorros e melhorar o direcionamento de pacientes a serviços de saúde adequados.

Como descrito por VENTURA(2022), no Brasil comumente há a cultura de recorrer aos serviços de pronto socorro, assim gerando idas desnecessárias à

emergência, em busca de orientações ou atendimento, comportamentos como esses terminam por gerar aumento considerável nas filas de atendimento. Além disso outro problema importante está na demora de acesso ao serviço de saúde, como demonstra no estudo “Avaliação das dificuldades enfrentadas pelo paciente para realização de uma consulta médica de nível terciário”, Uzeli(2013) após uma análise com mais de 205 pacientes de Base do distrito federal, foi constatado ao final da pesquisa que no Distrito federal, a precarização no acompanhamento regular de atenção primária nao chega nem a metade dos pacientes atendidos. Por fim, como demonstra em seu texto VENTURA(2022), evidencia tanto a superlotação das unidades de saúde quanto a possibilidade de diminuição das idas desnecessárias a serviços de emergência. Essa solução se daria pelo uso de aplicativos e dispositivos móveis para o direcionamento do paciente a uma assistência que fosse necessária, assim deixando os locais para atendimentos cruciais mais focado ao seu tipo de atendimento.

Prosseguindo em sua pesquisa, VENTURA(2022) fez a análise de aplicativos já existentes no âmbito de saúde. Foram identificados e analisados sete aplicativos, e, após o seu comparativo, foi gerado o Quadro 1.

Quadro 1 - Comparativo dos trabalhos relacionados

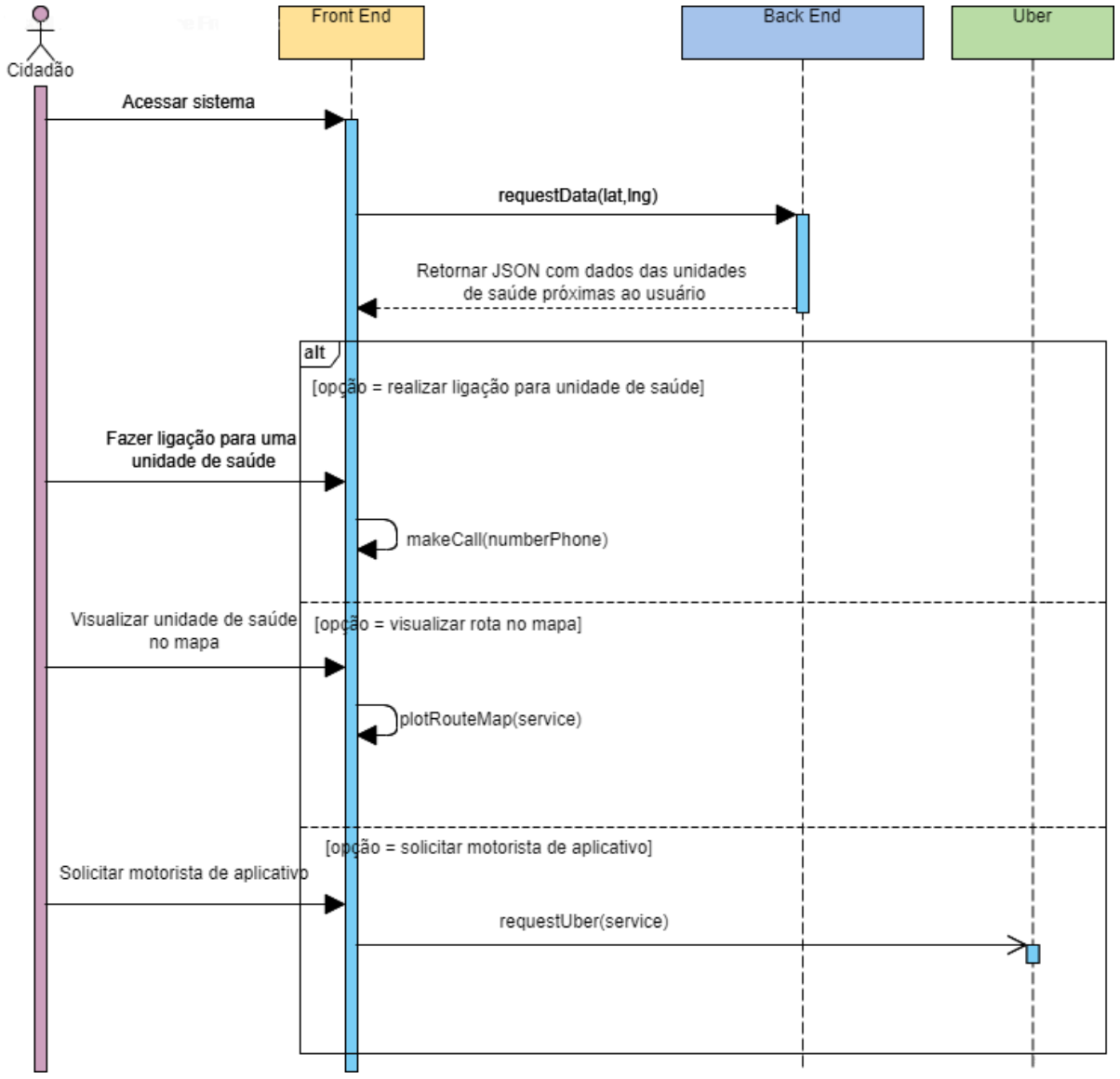
Funcionalidades/ Aplicativo	Conecte SUS	E-saúde SP	Dr.consulta	Sabin	Conecta Recife	App SAU	Unimed Cliente
Cartão de Vacinas	Sim	Sim	Não	Não	Sim	Não	Não
Localização de Serviços de Saúde	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Agendamento de Exames	Não	Não	Sim	Sim	Não	Não	Sim
Resultados de Exames	Não	Sim	Sim	Sim	Não	Não	Sim
Agendar Vacinas	Não	Sim	Sim	Não	Sim	Não	Não
Teleconsultas	Não	Sim	Sim	Não	Não	Não	Não
Agendamento de	Não	Não	Sim	Não	Não	Não	Sim

Consulta Presencial							

Fonte: VENTURA(2022)

Dando continuidade à análise de material produzido, alguns diagramas e demonstrações de demandas fornecidos por VENTURA(2022), deram um norte inicial, tanto a um funcionamento mais geral da aplicação, quanto a requisitos de dados e formatações. A seguir é demonstrado um diagrama de atividade (Figura 1) mais geral do funcionamento da aplicação, em seguida um exemplo de arquivo Json esperado pelo front-end (Listagem 1).

Figura 1 - Diagrama de sequência - Buscar unidades de saúde.



Fonte: VENTURA(2022).

Listagem 1 - JSON

```
[{
        "id": 1,
        "name":"HOSPITAL MUNICIPAL DR SEVERINO BEZERRA DE
    CARVALHO", "address": "FLORIANO PEIXOTO","district":
    "CENTENARIO", "phone": "(83) 3341 2097",
        "time": ["Segunda-Feira: 07:00 às 17:00",
                        "Terça-Feira: 07:00 às 17:00",
                        "Quarta-Feira: 07:00 às 17:00",
                        "Quinta-Feira: 07:00 às 17:00",
                        "Sexta-Feira: 07:00 às 17:00"],
        "lat": -7.221412,
        "lng": -35.888401,
        "city": "CAMPINA
    GRANDE-IBGE-250400", "type":
    "HOSPITAL ESPECIALIZADO",
        "management":
    "MUNICIPAL",
    "dependency": "MANTIDA",
    "CNES": 7113692,
        "speciality":["PEDIATRIA", "CLÍNICO
    GERAL"], "CEP": 58428-130}]
```

Fonte: VENTURA(2022).

2.2 Pesquisa das Fontes de Dados Suas Formas de Acesso

Nesta seção, será abordado e discutido as estratégias para acesso e a integração dos dados para o *back-end* do sistema, com o foco nas plataformas que disponibilizam esses dados, como o DATASUS. A análise abordará as evoluções ao longo dos anos e das plataformas e discorre sobre as formas de extração, as especificidades de acesso às bases de dados, aplicação de tratamentos, a integridade e confiabilidade dos dados que alimentarão o sistema.

Ademais, também é posto em vista no que se embasa essas bases de dados, onde o sistema necessita de algumas informações específicas retiradas dessas bases de dados, essas informações devem ser retiradas da base de dados, como endereço, nome das unidades de saúde, e seus códigos únicos de identificação, CNES. Através de uma base de dados bem tratada será possível alimentar o

back-end, de forma que ele seja capaz de fornecer todas as informações necessárias para o funcionamento do *front-end.*

2.2.1 DATASUS

Com início das buscas por bases de dados, foi descoberto o DATASUS(O Departamento de Informática do Sistema Único de Saúde), um órgão que surgiu em 1991, com a responsabilidade de fornecer dados do SUS(Sistema Único de Saúde), para assim fornecer suporte de informática necessários para o planejamento operação e controle de informações. Com essas bases de dados é possível ter acesso a uma ampla gama de atributos e informações sobre cada unidade de saúde cadastrada. As plataformas demonstradas nos próximos tópicos irão descrever de uma forma geral seu funcionamento, seus empecilhos de usos e as conclusões chegadas ao fim de cada análise

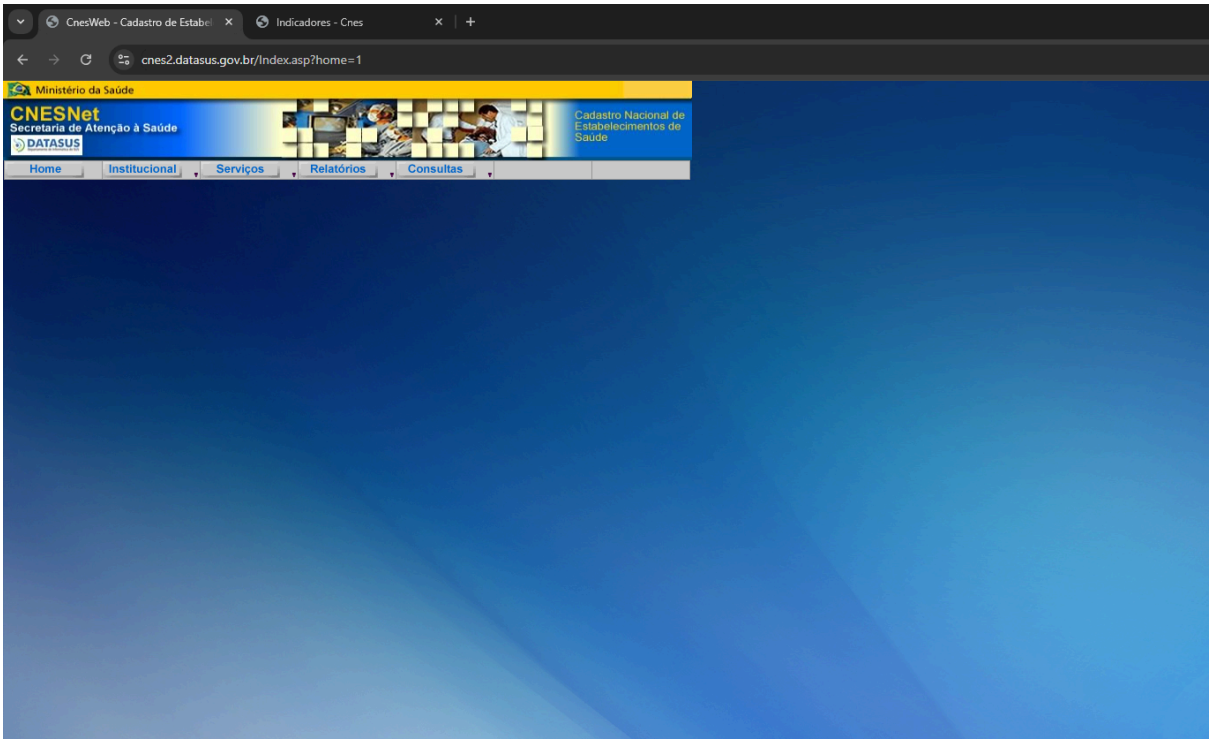
2.2.2 CNES

Segundo o Ministério da Saúde(Brasil,2024) o CNES(Cadastro Nacional de Estabelecimentos de Saúde), tem como objetivo ser a base para operacionar os sistemas de informação em saúde, sendo estes imprescindíveis para um gerenciamento eficaz e eficiente do SUS. Além disso o CNES, tratasse de um código vinculado a um estabelecimento específico de saúde, assim fornecendo diversas informações da unidade de saúde como endereço, gestores, serviços prestados e horários, dentre outros. Esses dados são informações essenciais esperado pelo sistema do front-end, o foco da busca passou a ser sistemas do DATASUS, que fornecessem bases de dados dos CNES, para assim fornecer as informações necessárias para o back-end e o sistema como um todo.

2.2.3 CNESNET DATASUS

Na pesquisa por plataformas que fornecessem dados esperado para a aplicação o primeiro encontrado foi o CNESNet, mantido pelo DATASUS, tratasse de uma plataforma mais antiga, e recentemente descontinuada. Nas imagens 2 e 3 temos algumas demonstrações dessa plataforma.

Figura 2 - Tela inicial do sistema CNESNet.



Fonte: CNESNET(2024).

Figura 3 - Tela CNESNet, amostragem estabelecimentos de um estado.

Cnes	Estabelecimento	Data Comp. Inicial	Data Comp. Final
3006204	CLINICA SANTA CLARA		
2613743	SAS		
2707519	HOSPITAL PADRE ZE		
2755483	CLINICA DOM RODRIGO LTDA		
2399776	HOSPITAL SAO VICENTE DE PAULO		
3056724	HOSPITAL UNIMED JOAO PESSOA		
2362848	HOSPITAL ANTONIO TARGINO		
2399318	HOSPITAL INFANTIL ARLINDA MARQUES		
2362880	HOSPITAL MUNICIPAL DR EDGLEY		
6940315	UNIDADE DE PRONTO ATENDIMENTO OCEANIA		
2504537	HOSPITAL DISTRITAL DEP MANOEL GONCALVES DE ABRANTES		
5654319	HOSPITAL UNIVERSITARIO NOVA ESPERANCA HUNE		
2592746	HOSPITAL E MATERNIDADE FLAVIO RIBEIRO COUTINHO		
3398315	PROCARDIO HOSPITAL MEMORIAL SAO FRANCISCO		
2400324	HOSPITAL EDSON RAMALHO		
2400065	HOSPITAL SAMARITANO		
2363070	HOSPITAL MUNICIPAL PEDRO I		
2605473	COMPLEXO HOSPITALAR DEP JANDUHY CARNEIRO		
2593262	HOSPITAL DE EMERGENCIA E TRAUMA SENADOR HUMBERTO LUCENA		
6754325	UPA UNIDADE DE PRONTO ATENDIMENTO		
2399814	HOSPITAL DE GUARNICAO DE JOAO PESSOA		
2362821	CLIPSI		
7870930	HOSPITAL DAS NEVES		
2399555	HOSPITAL MUNICIPAL SANTA ISABEL		
2399628	COMPLEXO HOSPITALAR DE MANGABEIRA GOV TARCISIO BURITY		
9132686	UPA CRUZ DAS ARMAS		
2603802	COMPLEXO DE SAUDE DO MUNICIPIO DE GUARABIRA		
3043142	AMIP		
2400243	HOSPITAL UNIVERSITARIO LAURO WANDERLEY		
2315793	HOSPITAL ESCOLA DA FAP		
2676060	HOSPITAL UNIVERSITARIO ALCIDES CARNEIROUFCG		
2362856	HOSPITAL EMERGENCIA E TRAUMA DOM LUIZ GONZAGA FERNANDES		
Total			32

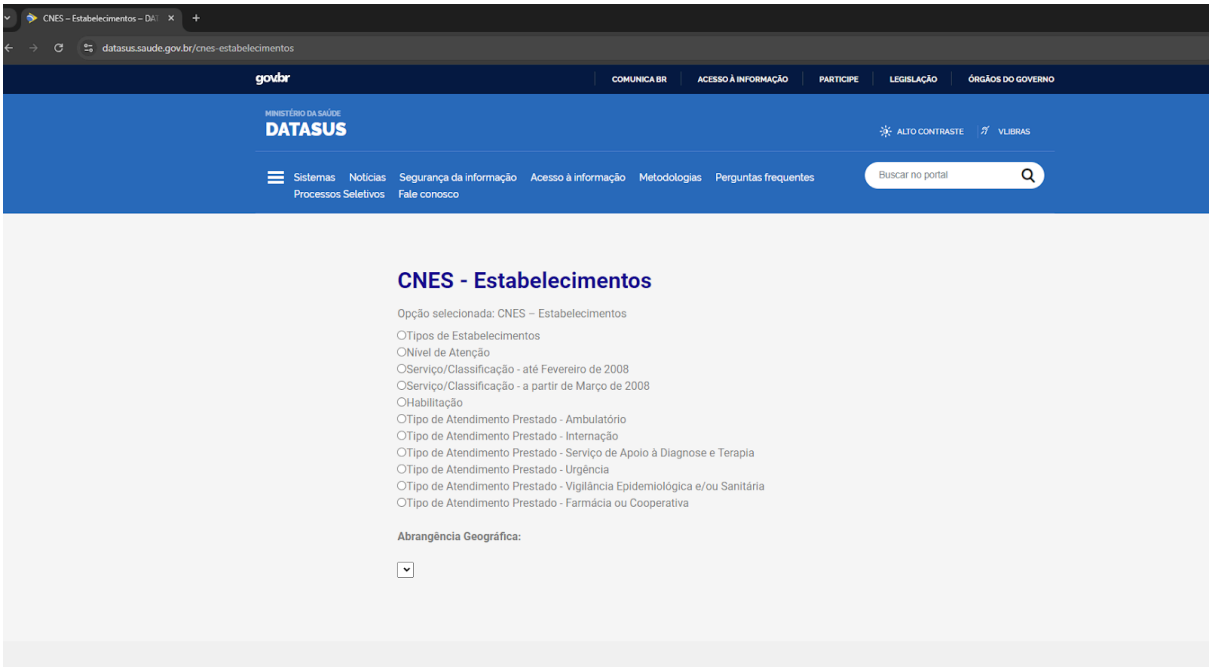
Fonte: CNESNET(2024).

Como mostrado nas figuras 2 e 3, podemos observar uma interface mais antiga, e com uma forma de acesso aos dados mais fragmentada. Para cada parâmetro necessário era gerado um arquivo diferente. Outro ponto negativo se deu pela partição de dados em estados, sendo necessário manualmente fazer a coleta de cada estado brasileiro diferente e individualmente. Quando solicitado os estabelecimentos, era apenas retornado o seu nome e seu código CNES, assim não sendo interessante, pois o sistema precisa de muitos dados auxiliares. Portanto foi posto como inviável a utilização dessa plataforma.

2.2.4 TabNet DATASUS

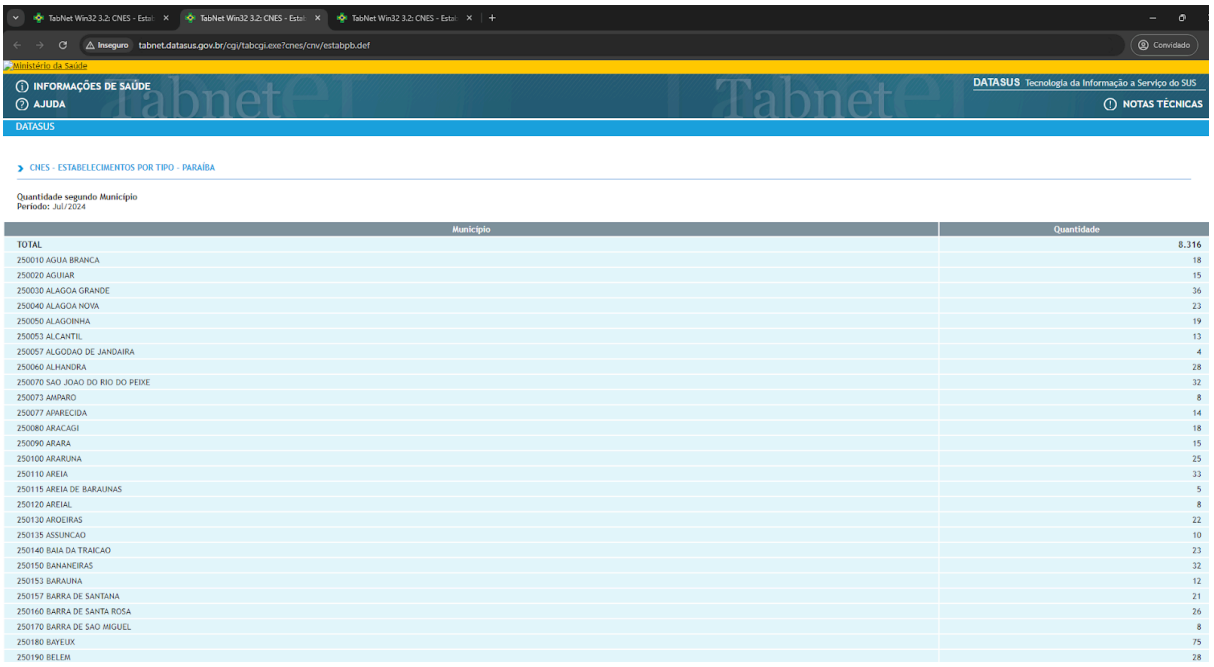
Passando para a próxima plataforma analisada chegamos na plataforma do datasus.saude.gov.br, Nessa plataforma temos uma interface mais amigável e atual. Demonstrado na figura 4 e 5 é visto a plataforma.

Figura 4 - Tela DataSus na área de estabelecimentos



Fonte: CNESNET(2024).

Figura 5 - Visualização dos dados gerados.



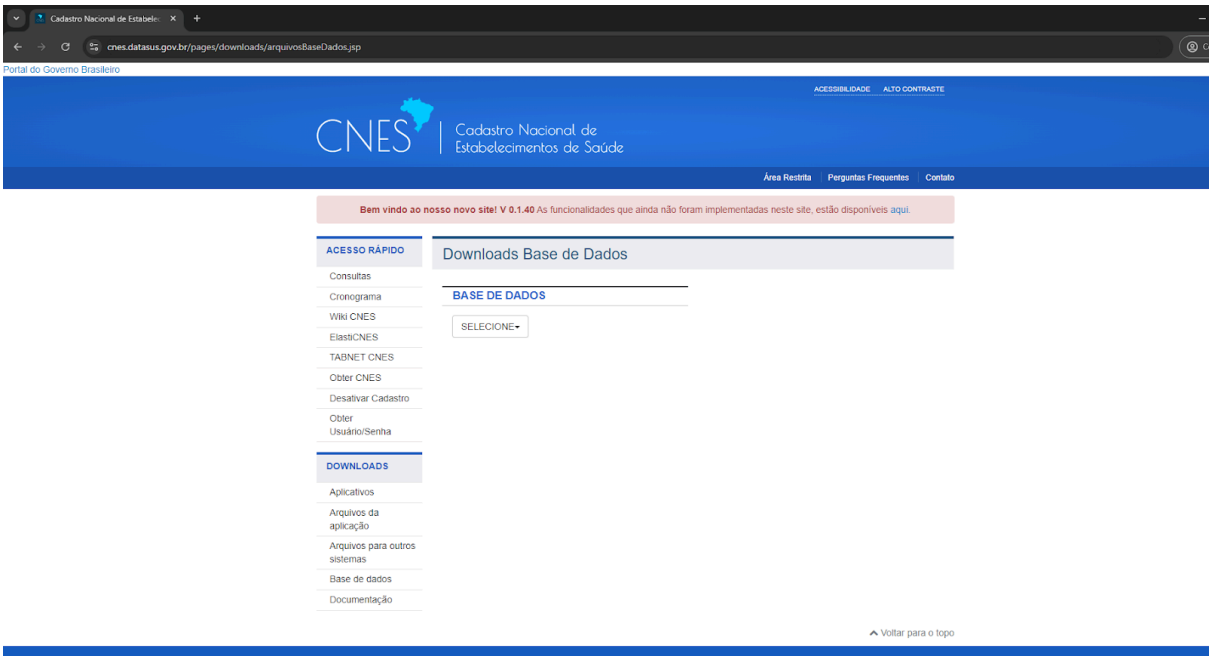
Fonte: CNESNET(2024).

Apesar da sua interface mais amigável, a plataforma TABNET apresenta o mesmo problema da CNESNet, possui dados muitos fragmentados, sendo assim necessário o acesso a vários arquivos diferentes e a necessidade de um trabalho manual muito grande. Contudo o pior problema enfrentado nesta plataforma se deu por ela utilizar uma tabulação de dados própria, o TABWIN. Essa ferramenta trata-se de um programa desenvolvido justamente para manejo de dados do DATASUS, entretanto como o objetivo era acessar esses dados para ter o controle deles em um sistema próprio, não parecia interessante insistir no uso dessa ferramenta externa, com essa conclusão foi dado continuidade a procura de novas plataformas.

2.2.5 CNES DATASUS

A próxima plataforma analisada parecia bastante promissora, a CNES DATASUS, não apresentava a utilização do TABWIN, possui uma interface atual e direta. Na figura 6 é possível analisar a interface do sistema.

Figura 6- Tela cnes datasus na área de downloads de bases de dados.



Fonte: CNES DATASUS(2024).

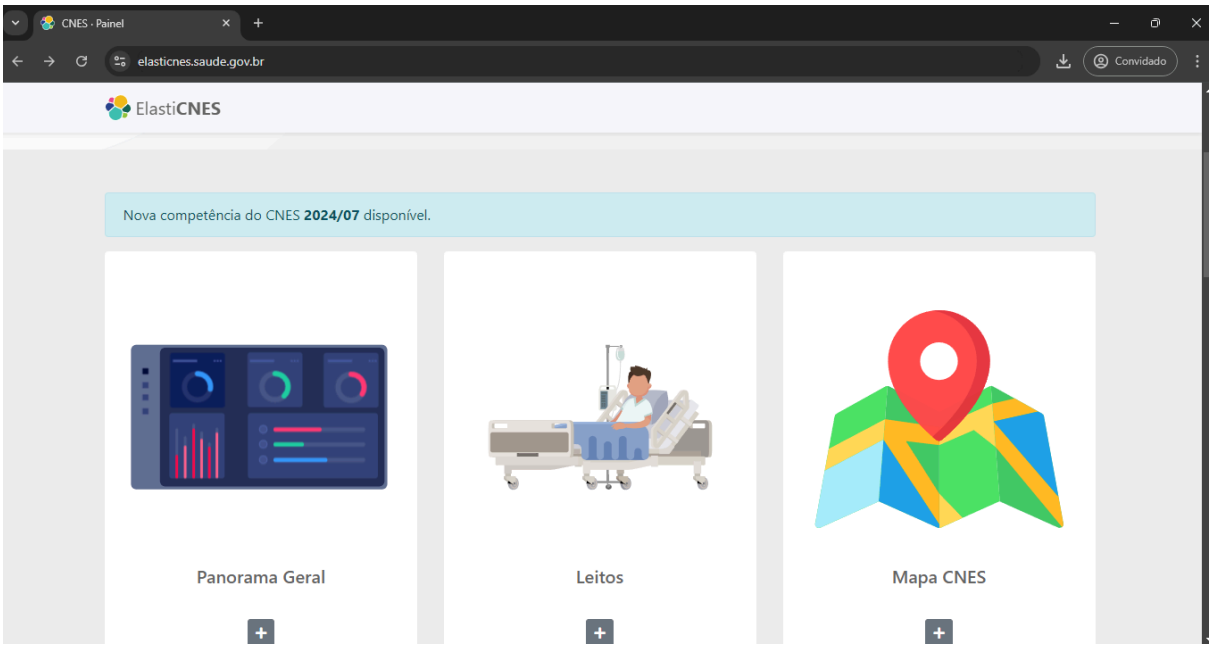
Ao analisar de forma detalhada, podemos ver na aba lateral da figura 6, a aba de acesso rápido, onde é possível fazer consultas precisas e obter informações precisas de unidades de saúde, além da já apresentada posteriormente TABNET. E finalmente mais embaixo o download de bases de dados. Contudo as tentativas de

download pelo navegador foram dificultadas, e ao analisar melhor, observamos que essa plataforma seria uma refatoração do CNESNet, onde na página inicial é apresentado o direcionamento para ela em caso de alguma funcionalidade faltante. Isso causou uma incerteza quanto a essa plataforma, e ao continuar a análise, uma plataforma que ainda não tinha sido estudada é apresentada e nos apresenta um grande potencial.

2.2.6 Elastic CNES

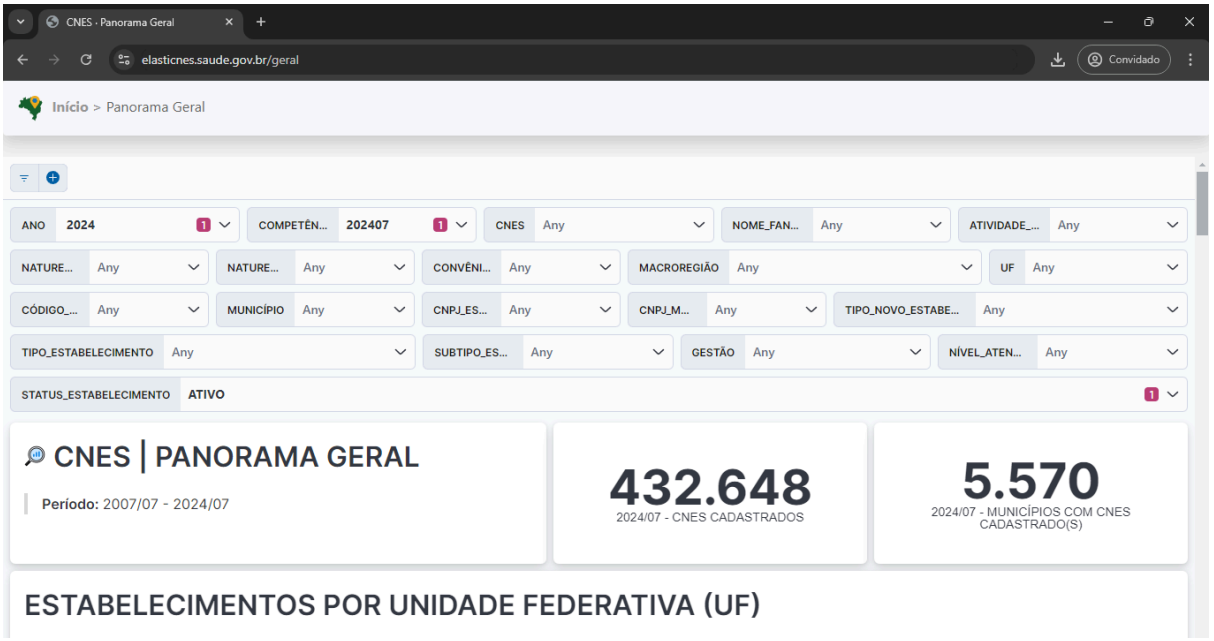
Em prosseguimento à procura de plataformas de fornecimentos de dados chegamos na mais recente. A elastic cnes, tratasse de uma plataforma de uso facilitado e demonstração de dados em formas de paineis, com capacidade de aplicar filtros e escolher atributos dos dados, Através desses filtros podem ser escolhidos os atributos baixados no arquivo final para cada unidade de saúde. Além de uma pré-visualização de dados bem interessante, nessa plataforma é possível baixar de forma facilitada bases de dados em arquivos csv, capazes de serem tratados e avaliados, e com alguns scripts de código serem convertidos para dados que condizem com o esperado do sistema. Na figura 7, 8 e 9 temos a apresentação da plataforma.

Figura 7 - Tela inicial Elastic CNES.



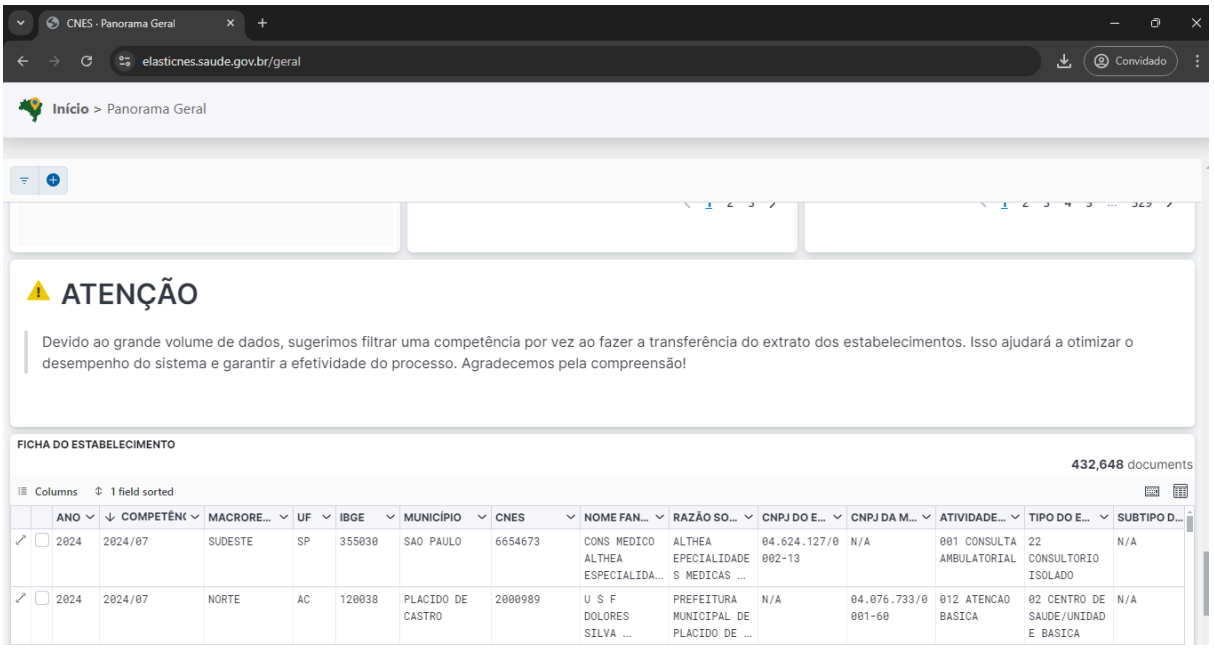
Fonte: Elastic Cnes(2024).

Figura 8- Tela Elastic cnes panorama geral.



Fonte: Elastic Cnes(2024).

Figura 9- Tela Elastic Cnes demonstração de dados.



Fonte: Elastic Cnes(2024).

Tendo em vista os requisitos da nossa aplicação, a plataforma Elastic cnes fornece dados em formato aceitável, uma pré-visualização dos dados interessante, e

dados complementares interessantes, informações como tipos de estabelecimento, códigos cnes, informações de localização, dentre outras. A única dificuldade foi o download dessas bases, como o próprio sistema notifica, dependendo da demanda de volume de dados pode haver problemas de desempenho, assim sendo necessário fazer downloads de bases particionadas.

Analisando as plataformas encontradas, foi observado a variedade de formas de acesso aos dados do DATASUS, em específico os dados referentes a unidades de saúde, que são as informações necessárias para o funcionamento dos *back-end*. Cada plataforma possui suas especificidades, e detalhes que quando levado em consideração, fizeram com que uma plataforma se destacasse perante as outras. A elastic Cnes, foi a forma de acessar os dados de unidades de saúde que mais se encaixou com as necessidades do projeto, as bases de dados se apresentam mais concisas, amplas quanto aos atributos de informações, e os filtros e pré visualização dos dados facilitou o manejo das bases de dados. Portanto ela foi escolhida como plataforma para acesso aos dados que alimentarão o projeto.

3 METODOLOGIA

Considerando os resultados apresentados por VENTURA(2022) e tendo como objetivo o desenvolvimento do back-end, iniciou-se uma busca por embasamento científico atualizado através de pesquisas sobre as tecnologias de desenvolvimento que serão utilizadas no projeto em questão, bem como as mudanças no acesso aos dados que são o cerne da nossa aplicação, foi iniciado o processo de análise e planejamento do sistema. Com esse prévio conhecimento do território a qual seria estudado, foi iniciado o processo de desenvolvimento do projeto, visando proporcionar não apenas um produto final, mas também uma experiência de aprendizado mais completa e condizente com futuros ambientes, tanto de mercado de trabalho quanto de pesquisas acadêmicas. Este trabalho visa além de descrever e demonstrar o desenvolvimento de um projeto, todas as etapas percorridas evidenciando a utilização de diversas especialidades adquiridas através de experiências vividas na formação acadêmica, visando assim fornecer um produto que atenda completamente as expectativas, tanto no processo de construção até desfrutar do resultado final.

Os primeiros passos do projeto se deram pela análise do texto de VENTURA(2022),buscando principalmente a identificação de requisitos, tanto os requisitos funcionais como os não funcionais. Essa escolha foi tomada pela existência de um sistema prévio implementado, com isso, eram visto demandas claras e fundamentadas pelo front-end(o principal consumidor dos dados do sistema), além disso, era visto as necessidades de requisitos específicas do tipo de aplicação que deveria ser construído e levar assim em consideração tanto o seu ambiente de uso quanto sua usabilidade. Após a coleta desses requisitos foi possível aprofundar a análise e avançar para a etapa de diagramação do projeto. Visando descrever de forma eficiente e sistemática o projeto foram elaborados diversos diagramas para descrever tanto a estrutura quanto os comportamentos esperados pelo sistema, com isso poderia ser feita a validação e documentação do sistema como um todo. Apesar de diversas vezes ser uma etapa negligenciada durante o projeto, vale a pena reforçar sua importância, e não apenas para facilitar o processo de desenvolvimento de código mas para geração de materiais de consulta

que posteriormente possuem grande serventia para casos de consulta para reparos de códigos ou até mesmo melhorias e evoluções no projeto.

Dando seguimento, na próxima etapa, foi analisado e definido o conjunto com as tecnologias e ferramentas a serem utilizadas no projeto. As decisões foram tomadas levando em consideração as necessidades que deviam ser atendidas, além de também possuir relevância no mercado de trabalho e de pesquisa, pois assim, teríamos o uso de tecnologias validadas tanto no meio empresarial quanto científico. Outro valor levado em consideração foram as experiências acumuladas ao longo da carreira academia e afinidades com tecnologias, assim sendo capaz de colocar em prática as habilidades que foram adquiridas ao longo do curso de graduação. Finalmente foi iniciado o processo de desenvolvimento do sistema propriamente dito, seguindo os requisitos e diagramas que foram criados nas etapas anteriores e utilizando as ferramentas e tecnologias selecionadas, foi determinado o fluxo de trabalho cíclico de análise, desenvolvimento e validação. Por fim, todo o processo de desenvolvimento e documentação do projeto foi concluído e será detalhado ao longo deste trabalho.

4 TECNOLOGIAS ENVOLVIDAS

Nesta seção será feita uma breve apresentação das tecnologias,conceitos e ferramentas envolvidas no projeto, além de breves explicações e justificativas de seus usos no contexto do projeto.

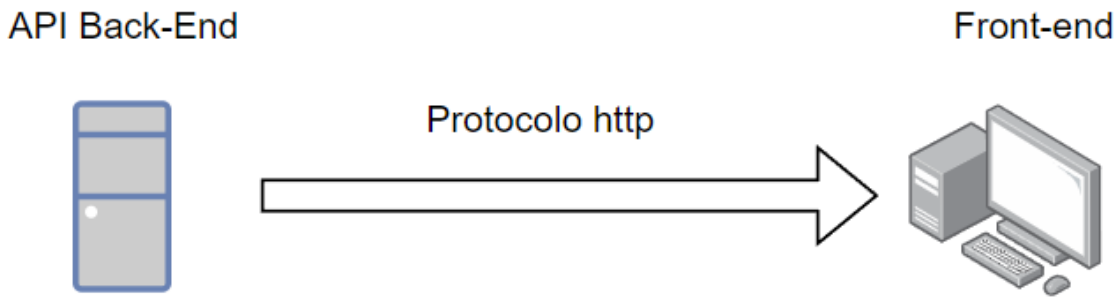
4.1 Http

Segundo a documentação da MDN(2024) temos *http(Hypertext Transfer Protocol)*, como um protocolo de transferência de dados, de forma simplificada temos uma comunicação entre um cliente e um provedor, através de requisições iniciadas por um cliente até um destinatário, geralmente documentos como *HTML(HyperText Markup Language)*, e utilizadas comumente em navegadores, pode ser feito o acesso a diversas fontes de conteúdos distintos e através dessas requisições adquirir ou enviar as informações. Como o projeto estudado se trata de um front-end e um *back-end*, é necessário que haja uma linha de comunicação entre eles, é justamente nesse espaço que entra o protocolo *http*, unindo os dois programas.

4.2 Api

Tendo em mente inicialmente a explicação do nome *API*,como traduzido para o portugues, interface de programação de aplicativos, uma breve e sucinta explicação é de Medeiros(2014), onde ele explica que *API* tratasse de um conjunto de padrões de programação que tornam possíveis construir e utilizar aplicações, apesar de ser um conceito amplo, podemos entender uma *API* como um código capaz de fornecer um serviço esperado ou sistemático, seja entrega de informação, tratamento de alguma informação ou até mesmo chamada de uma outra *API*. A construção da demanda do *front-end*, o *back-end* que forneça os dados de unidades de saúde, será uma *API* que detém uma regra de negócio e fornece esses dados de acordo com ela. Na figura 10 é demonstrada uma comunicação entre sistemas, utilizando o conceito de *API.*

Figura 10 - Representação de comunicação dos sistemas.

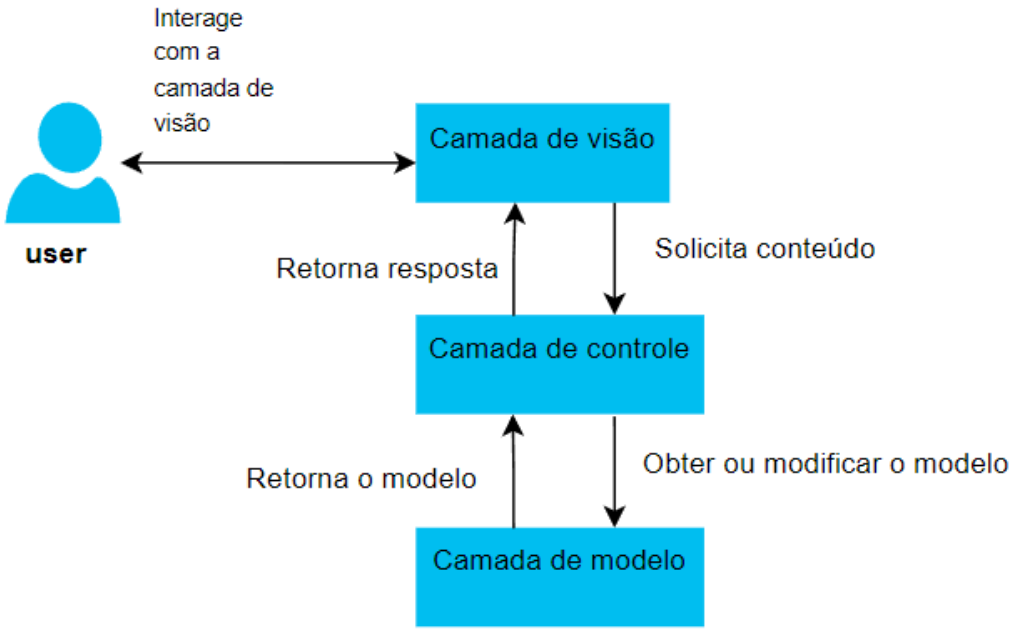


Fonte: Elaborado pelo autor, (2024).

4.3 Mvc

O significado das siglas MVC(*Model View Controller*), traduzindo para modelo, visão e controlador, tratasse de um padrão de software, onde há uma separação das estruturas nas três camadas que dão o seu nome, controlador, visão, modelo. Cada camada possui sua responsabilidade, onde a camada de modelo dita o tratamento dos dados e como eles são modelados, o controlador controla o fluxo dos modelos de dados, aplicando a lógica de negócio desejada, e por fim a camada de visão fica responsável por transformar tudo isso em informação visível e interpretável para o usuário final. Como demonstrado na figura 11, é apresentada uma abstração do modelo *MVC*.

Figura 11 - Representação do modelo MVC.

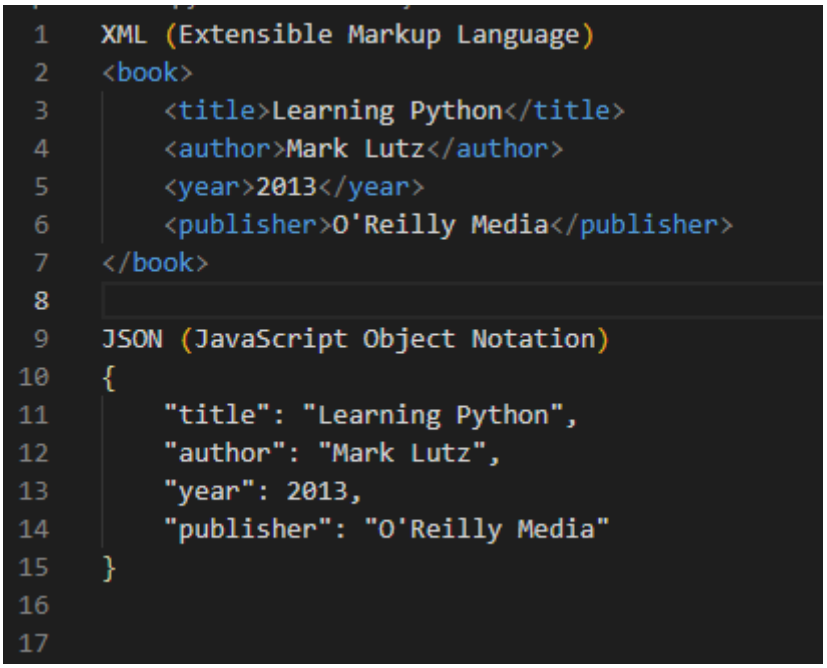


Fonte: Elaborado pelo autor, (2024).

4.4 Json

De acordo com Gama(2013) *JSON*, ou *JavaScript Object Notation*, nada mais é do que um formato de troca de informações entre sistemas, além de ser leve possui uma fácil leitura e interpretação dos dados. Em sistemas mais antigos era utilizado uma formatação de dados no modelo XML(*Extensible Markup Language*), contudo, foi substituído pelo *JSON*, por sua praticidade. Na Figura 12, onde é apresentado um modelo de dados utilizando o JSON e o XML, é possível observar a eficiência do protocolo JSON.

Figura 12 - Comparação do modelo JSON e XML.



Fonte: Elaborado pelo autor, (2024).

Utilizando o protocolo de comunicação *HTTP* em conjunto com o padrão de design MVC, é possível a comunicação entre o *front-end* e o *back-end*, e essa comunicação se dá utilizando a linguagem *JSON*.

4.5 Python

Como cita em sua documentação oficial PYTHON.org(2024)"*Python* é uma linguagem de programação que permite que você trabalhe rapidamente e integre sistemas de forma mais eficaz". *Python* trata de uma linguagem de programação que vem se destacando cada vez mais nas últimas décadas. Como mostra Stackoverflow(2023), *Python* chegou a ultrapassar várias linguagens e chegou a ser a terceira linguagem mais comumente usada, e quando se trata de desenvolvedores que estão aprendendo a codificar ficou em primeiro lugar. O *python* tem se destacado bastante principalmente na área de dados, e pela demanda do projeto de trabalhar com bases de dados extensas e com diversos dados, foi adotado o uso do *python* nessa área de tratamento de dados. Além disso, especificamente nesse projeto, o python foi essencial para o manejo de dados, pois com ele foi capaz de ser realizada junções de dados, validações, e principalmente correções de inserção de dados.

4.6 Java

Java, assim como *Python* tratasse de uma linguagem de programação, com um forte espaço tanto no mercado de trabalho, quanto acadêmico,o java demonstra seus pontos fortes principalmente pela sua documentação concisa e facilidade de trabalho principalmente na área de construção de *APIs* de *back-end*. De acordo com Stackoverflow(2023) java também se mostra uma forte linguagem permanecendo no top 10 linguagens mais utilizadas entre desenvolvedores profissionais, além de ser cerca de 37% das vezes escolhida como propensa a escolha de linguagem para iniciantes. Além disso, ao longo do período de graduação, por sua farta documentação e material literário, possui bastante espaço na carreira de graduação acadêmica superior.

4.7 Banco de Dados

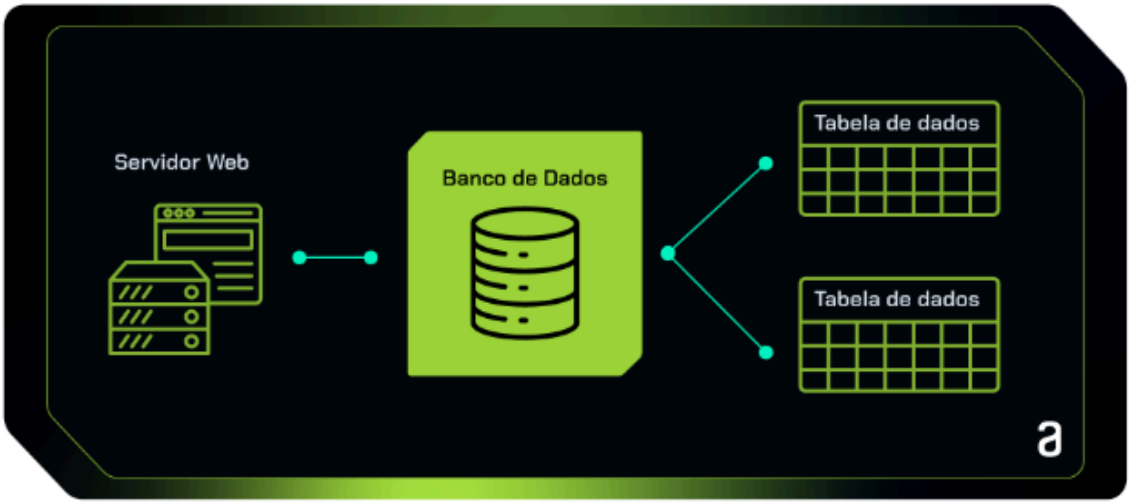
De acordo com Rezende(2006), temos que um banco de dados trata-se de um conjunto de informações parecidas e que se relacionam, sendo possíveis de agrupar. Com isso temos de forma simples que o banco de dados se trata de uma estrutura onde iremos agrupar tipos específicos de informações. Além disso, temos sistemas gerenciadores de banco de dados, e através deles é possível através de linhas de códigos realizar operações com essas informações do banco de dados. No caso do sistema que será implementado, será utilizado um banco de dados para armazenar os dados de uma forma ordenada, e que nos forneça a capacidade de realizar a persistência das informações, ou seja, a capacidade de manter essas informações mesmo se houver o encerramento da execução do sistema.

4.8 Postgresql

O postgresql, trata-se de um sistema que gerencia um banco de dados, que utiliza uma linguagem baseada no SQL(Linguagem de consulta estruturada). O PostgreSQL é conhecido por sua conformidade com ACID (Atomicidade, Consistência, Isolamento, Durabilidade), garantindo transações seguras e confiáveis. Além disso, ele suporta tipos de dados avançados, índices, e extensões, tornando-o adequado para uma ampla variedade de aplicações, desde pequenas

aplicações web até grandes sistemas empresariais. Na figura 13, é apresentada uma abstração do funcionamento de um banco de dados, de uma forma visual.

Figura 13 - Funcionamento de um banco de dados.



Fonte: Elaborado pelo autor, (2024).

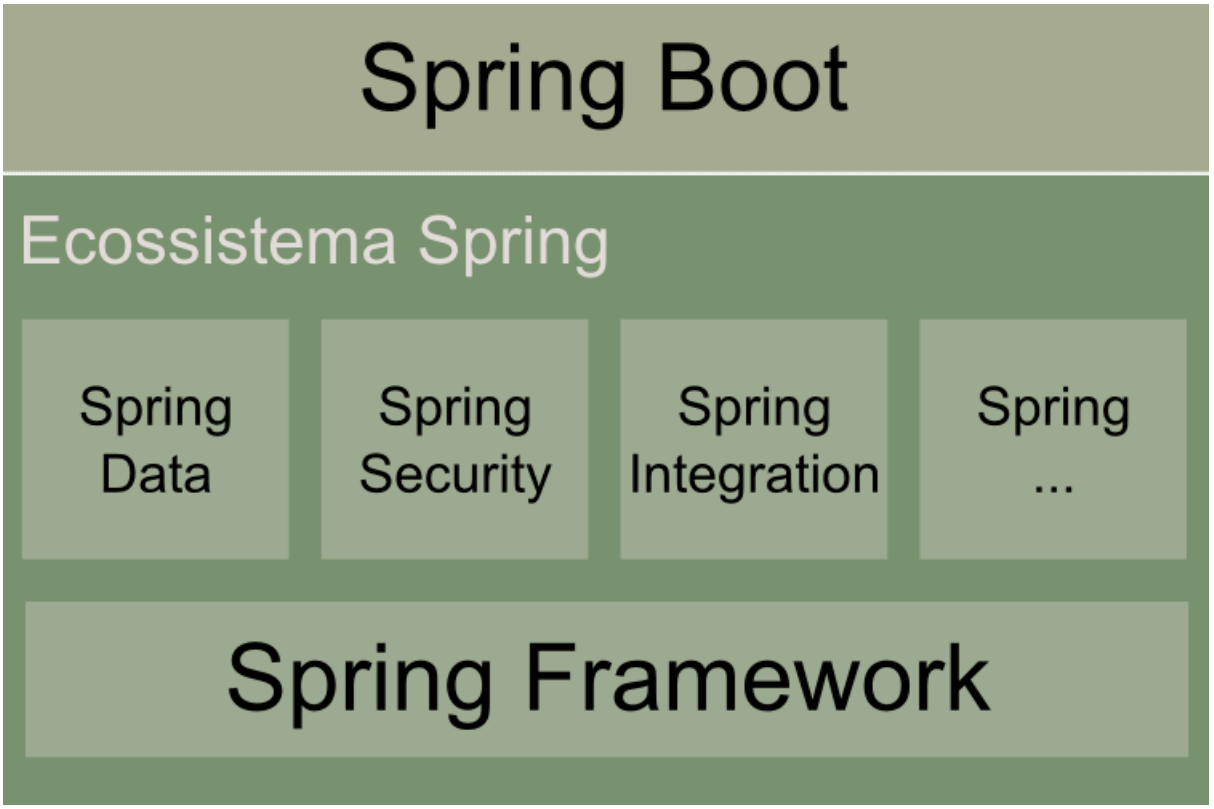
4.9 Projeto Spring Framework

O *Spring Framework* é uma plataforma amplamente utilizada e completa, utilizando a linguagem Java auxilia no desenvolvimento de aplicações mais robustas, escaláveis e flexíveis. Teve seu lançamento em 2002, desde então o *Spring* se destaca pela sua modularidade, que o torna mais simples principalmente para integração com outras tecnologias. Assim como todos os *Frameworks*, o *Spring* surge com o foco no meio empresarial facilitando o desenvolvimento de softwares fazendo abstrações para tarefas complexas como a gestão de dependências e transações e comunicação entre componentes.

Como cita WEISSMAN(2015), O *Spring* reaproveita as tecnologias a fim de aumentar a produtividade do desenvolvedor, tratasse de uma excelente ferramenta quando se fala de criação de sistemas que utilizam a arquitetura de microservices. Contudo o *Spring framework* não se limita a isso, muito pelo contrário, o ecossistema Spring apresenta soluções para diversas áreas, como por exemplo *Spring Data* para manejo de dados, *Spring security* para segurança, o *Spring boot* para criação de aplicações web que será visto posteriormente, dentre outras

soluções. Na figura 14 é possível observar algumas possibilidades do ecossistema Spring.

Figura 14 - Ecossistema Spring.



Fonte: Elaborado pelo autor, (2024).

Ainda citando WEISSMAN(2015) a facilidade que o desenvolvedor terá no ecossistema spring é muito interessante, uma vez que ao possuir conhecimento no ecossistema ele é altamente reaproveitado, tendo em vista que o foco em configurações e organização da aplicação é muito semelhante, o desenvolvedor é capaz de vastas soluções diferentes.

Uma das características mais marcantes do *Spring* se dá pelo grande suporte à injeção de dependência, que permite que objetos sejam criados e gerenciados pelo *framework*, assim reforçando a ideia apresentada por Weissman. Além dessa facilidade desenvolvimento também é adicionado muitas melhorias sistemáticas, sendo exemplos disso, a redução de acoplamento de código,a facilidade de suporte e manutenção do código.

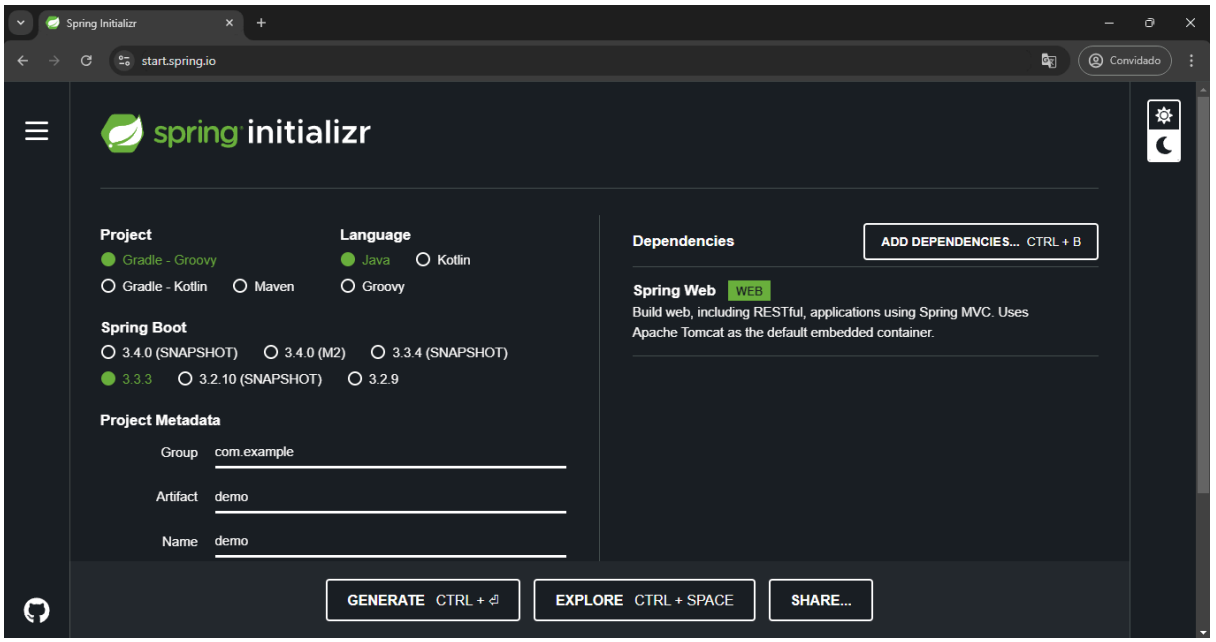
4.10 Spring Boot

O *Spring Boot* trata de um dos módulos do *Spring Framework*. Lançado em 2014, o foco era simplificar o processo de criação de aplicações *Spring*. Essa facilidade se deu principalmente pela facilidade de configurações automáticas do *Spring Boot.* Como explica Weissman(2015) o Spring boot resolve o problema de gerenciamento de dependências, facilitando a modularização das diversas tecnologias do projeto *Spring* em "*Starter POMs*", que são arquivos que já contém as configurações necessárias para o programador usar as funcionalidades contidas naquele módulo. O uso desses starter POMs facilitou o uso de diversos módulos, pois tanto facilita o processo de gerenciamento de dependências como permite o programador utilizar diversos recursos de forma simples e intuitiva.

Outra característica essencial no Spring Boot se dá pela Configuração automática adicionando vários suportes na criação do projeto, uma demonstração disso é o servidor embutido na aplicação que já vem, o Tomcat. que permite a aplicação ser executada de forma standalone, ou sejam, sem a necessidade de implantar um servidor externo.

O Spring Boot é ideal para a criação rápida de aplicações, especialmente micro serviços, onde a simplicidade e a velocidade de desenvolvimento são fatores cruciais. Com ele, é possível focar no código da aplicação sem se preocupar excessivamente com a infraestrutura subjacente. Na figura 15 é demonstrado um modelo de criação de um projeto Spring Boot e adição dos Starte POMs.

Figura 15 - Demonstração de criação de um projeto Spring com Startes POMs.



Fonte: Elaborado pelo autor, (2024).

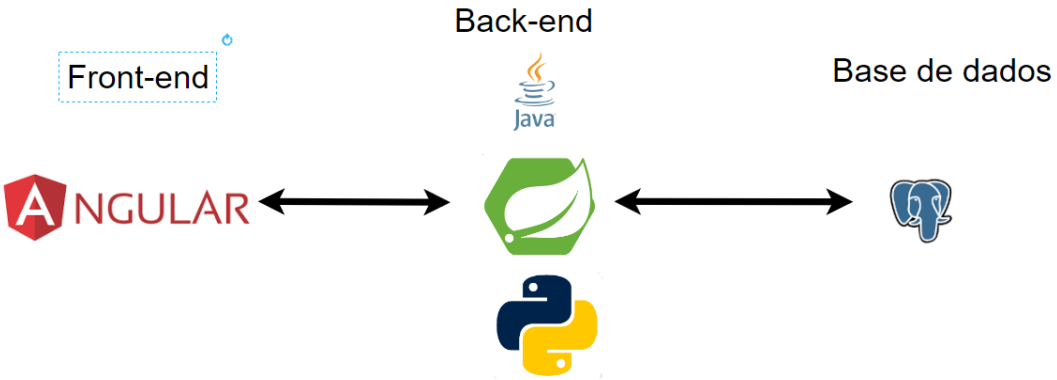
4.11 Ferramentas Utilizadas

Como mostrado anteriormente nessa seção, houve apresentação de diversas tecnologias e conceitos, o entendimento prévio dele é essencial para o entendimento das próximas etapas. Além das tecnologias foram necessárias ferramentas e plataformas para o uso das mesmas. Algumas delas foram:

- VISUAL STUDIO CODE
    - Um editor de código que possui diversas extensões, assim o programador pode torná-lo otimizado e que atenda as especificidades da sua área.
- DRAW.IO
    - Uma ferramenta online de diagramação, com ela é possível escrever diagramas de diversos tipos, desde UM, até mesmo diagramas mais simples para comunicação e demonstração de ideias de uma forma visual.

Por fim temos A união dos conceitos apresentados anteriormente, reunidos na figura 16, para definir o projeto de uma forma geral.

Figura 16 - Ferramentas utilizadas.



Fonte: Elaborado pelo autor, (2024).

5 DESENVOLVIMENTO

Inicialmente idealizado no projeto desenvolvido por Daniele Ventura Batista, intitulado *Desenvolvimento Front-End de Aplicativo para Busca de Serviços de Saúde* (2022), o seu projeto tratava do fornecimento de informações de unidades de saúde. Com toda a análise feita do projeto já existente, o *front-end* , nesta seção será apresentado o processo de desenvolvimento da próxima etapa deste projeto, inicialmente a coleta de requisitos funcionais e não funcionais, que foram convertidos em diagramas, para facilitar por fim a transformação em código. Além disso, serão apresentadas as dificuldades ao longo desse processo de desenvolvimento, bem como as soluções encontradas para tais.

5.1 Apresentação de Requisitos Funcionais e Não Funcionais

A descrição de requisitos gerou a construção de dois quadros(Quadros 1 e 2), que descrevem de forma sucinta os principais requisitos funcionais e não funcionais do sistema.

Quadro 2 - Requisitos funcionais do sistema.

ID	Requisitos Funcionais	Descrição
RF01	Manter unidades de saúde	realizar edição, exclusão, cadastro e visualização de informações das unidades de saúde
RF02	Importar bases de dados	Implementar requisição de armazenar dados a partir de um arquivo
RF03	Autenticação de requisição	A requisição deve ser validada para permitir ou não o acesso aos dados
RF04	Prover uma documentação de uso da api	Criar uma visualização de documentação e acesso a demonstração de uso da API
RF05	Fornecer dados versão mobile	Fornecer dados de uma versão específica condizente com as necessidades da plataforma.

Fonte: Elaborado pelo autor, (2024).

Quadro 3 - Requisitos Não Funcionais do Sistema.

ID	Requisitos Não Funcionais	Descrição
RNF 01	Desempenho	O sistema deve responder as requisições de forma eficiente e suportar altas cargas de requisição.
RNF 02	Segurança	O sistema deve garantir a segurança dos dados e o acesso a eles.
RNF 03	Usabilidade	O sistema deve fornecer uma experiência agradável e de uso intuitivo.
RNF 04	Escalabilidade	O sistema deve ser capaz de lidar com altas cargas de dados mantendo o desempenho de operabilidade.

Fonte: Elaborado pelo autor, (2024).

5.2 Modelagem

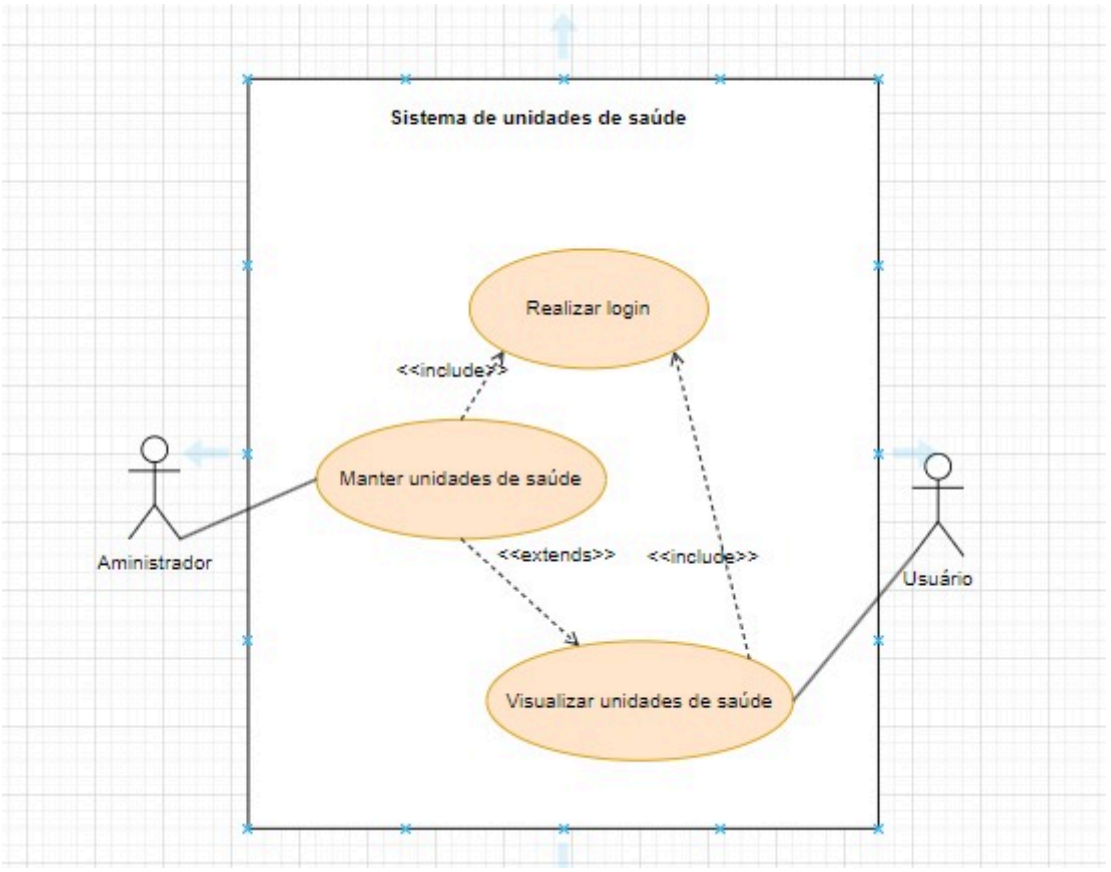
Considerando os requisitos que foram levantados, foi criado uma série de de diagramas, visando facilitar a criação de código, e uma melhor visualização gráfica do projeto. Foram construídos os mostrados nos próximos tópicos.

5.2.1 Diagrama de Caso de Uso

Apresentado como um dos primeiros diagramas gerados, o diagrama de caso de uso possui uma linguagem de fácil compreensão, proporcionando um esboço geral do funcionamento.

Nesse diagrama demonstrado na figura 17, é apresentado o administrador que ao realizar login, possui a habilidade de manter as unidades de saúde, isto é, criar, deletar, visualizar e editar as unidades de saúde. Outro agente apresentado é o usuário, que seria uma pessoa que utiliza o sistema apenas para visualizar as unidades de saúde próximas dele, mas também possui um login de acesso.

Figura 17 - Diagrama de caso de uso.



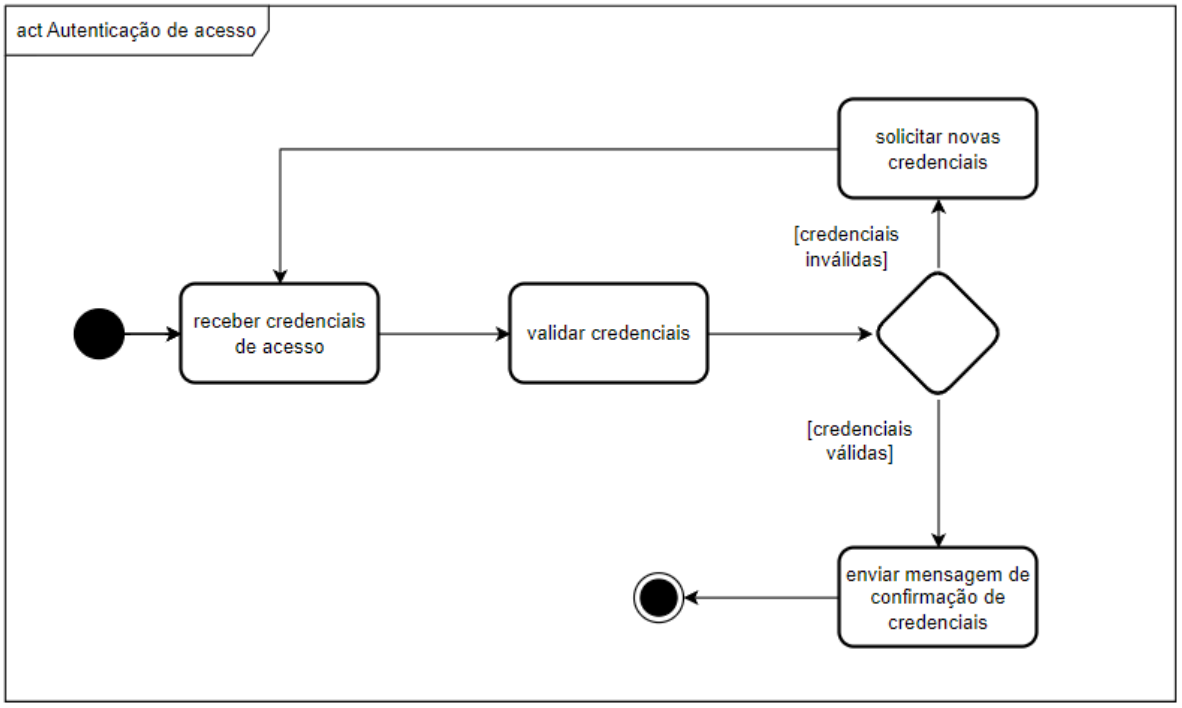
Fonte: Elaborado pelo autor, (2024).

5.2.2 Diagrama de Tarefa

Visando demonstrar os processos de passo a passo de uso do sistema, foram criados alguns diagramas de atividade, para assim poder visualizar o processo e o ciclo da atividade.

Na figura 18, temos o ciclo de atividade necessário para realizar autenticação no sistema.

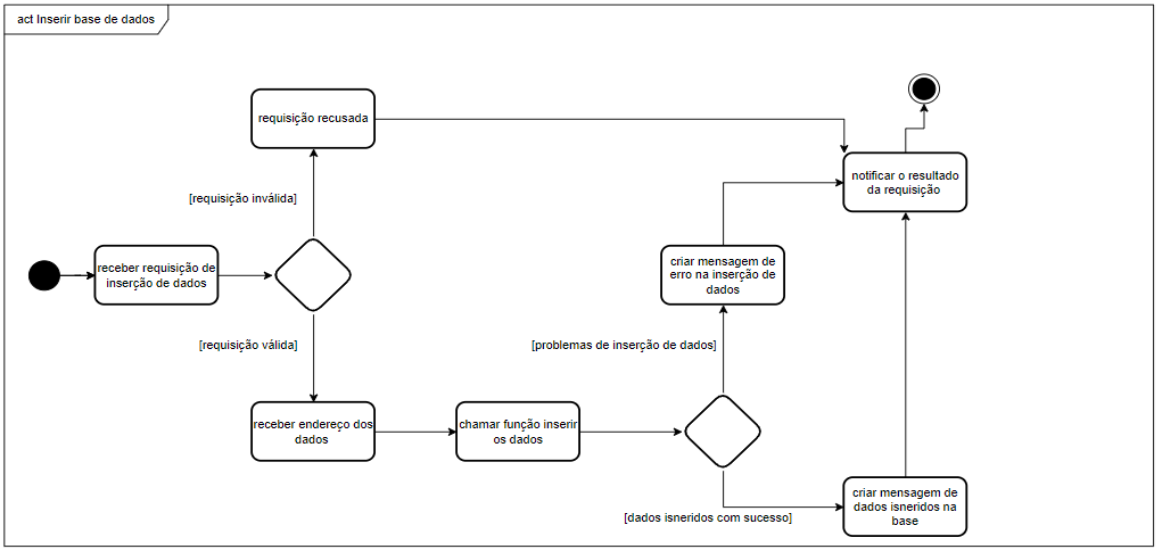
Figura 18 - Diagrama de tarefa para autenticar acesso.



Fonte: Elaborado pelo autor, (2024).

A figura 19, Demonstra como funcionará a inserção de dados fornecendo uma base de dados, invés do cadastro unitário de cada unidade de saúde.

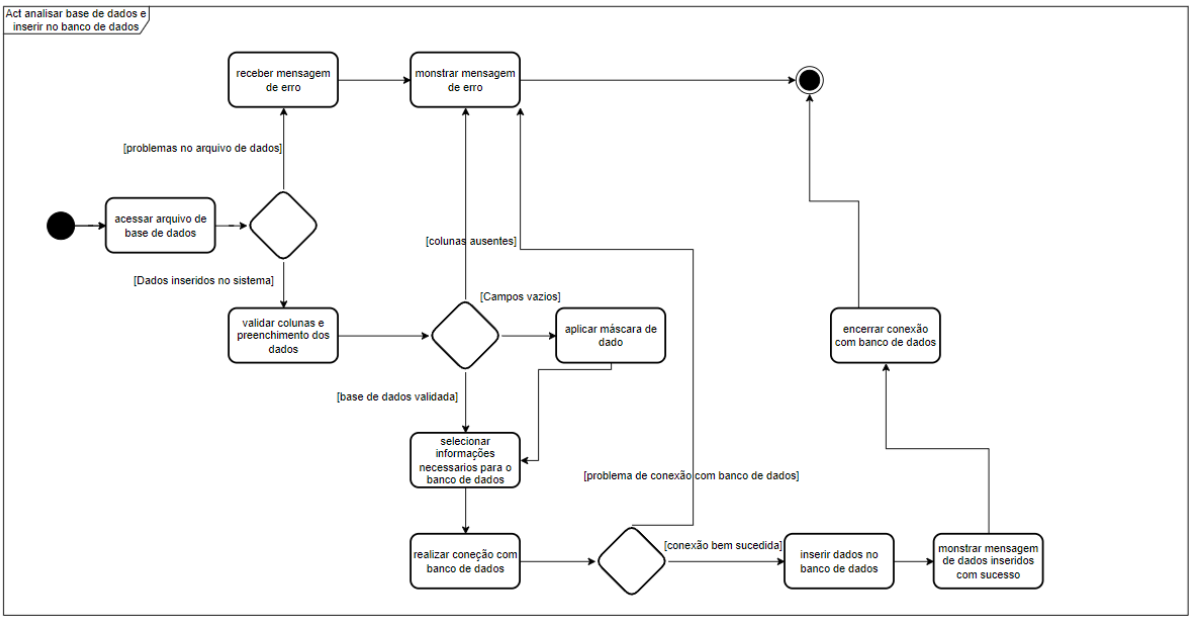
Figura 19 - Diagrama de tarefa para inserir uma base de dados.



Fonte: Elaborado pelo autor, (2024).

A figura 20 demonstra o acesso a unidades de saúde da versão de dados mobile.

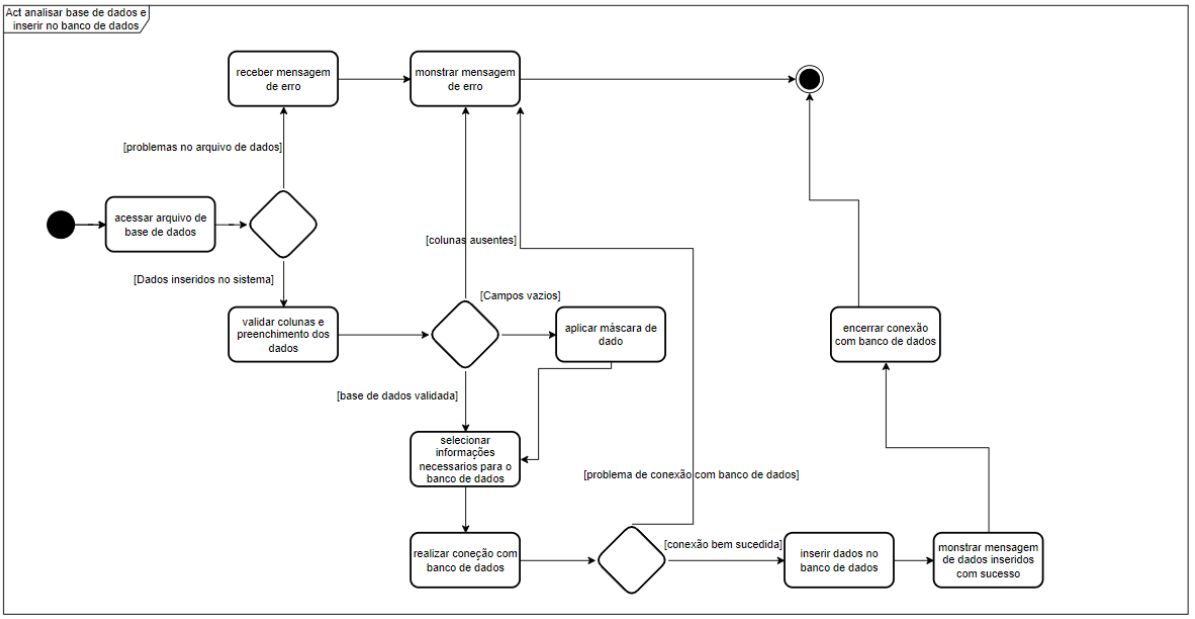
Figura 20 - Diagrama de tarefa para visualização de unidades de saúde.



Fonte: Elaborado pelo autor, (2024).

A figura 21 demonstra o processo de inserção de bases de dados no banco de dados.

Figura 21 - Diagrama de inserção de dados no banco de dados.



Fonte: Elaborado pelo autor, (2024).

5.3 Download das Bases de Dados.

Iniciando o processo de codificação do sistema, foi decidido focar os esforços na obtenção de uma base de dados de unidades de saúde, realizar o tratamento desses dados e validação deles, por fim inserir no banco de dados para o *back-end*.

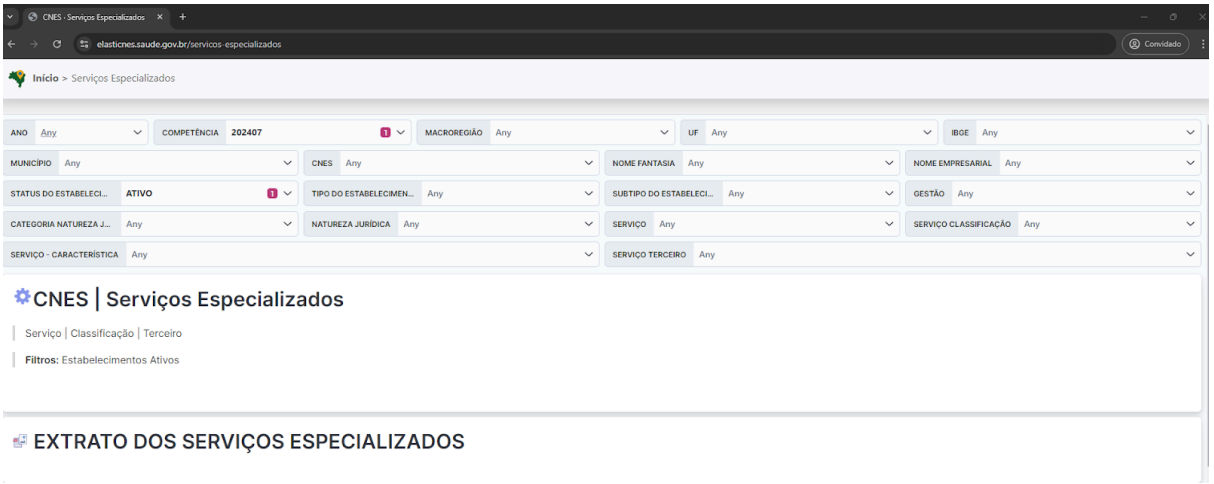
Seguindo essa sistemática, as diversas plataformas de fornecimento de dados do DATASUS citadas na seção de revisão da literatura, foi encontrada uma base de dados mais concisa no Elastic Cnes.

Na tentativa de baixar a base de dados com todos os estados brasileiros, o sistema apresentou inconsistências, assim sendo necessário a aplicação de filtros, aplicando o filtro de unidades da Paraíba foi gerado um arquivo com trinta e cinco colunas de atributos e pouco mais de quatro mil unidades distintas de saúde.

Entretanto, ao analisar as colunas da base de dados, foi notado a ausência de um parâmetro que definisse o tipo de atendimento prestado na unidade de saúde. Um esforço considerável foi aplicado para achar algum dos parâmetros existentes nessa base de dados que fornecesse alguma relação com o tipo de atendimento, porém não obteve êxito.

Continuando a pesquisa, acessando a documentação do CNES, foi encontrada uma outra base de dados, com menos parâmetros, entretanto com um código e descrição do serviço prestado na unidade de saúde.

Figura 22 - Tela Elastic Cnes para visualização de serviços especializados.



Fonte: Elaborado pelo autor, (2024).

5.4 Manipulação dos Dados Com Python

Com a realização do download do arquivo de dados com informações de unidades de saúde, foi utilizada a linguagem python para transformar esses dados brutos em uma base de dados aceita para o funcionamento do programa. Foram encontradas diversas inconsistências nas bases de dados. Para solucionar isso foi realizada a criação de scripts de códigos para diversos tratamentos e manipulações dos dados.

Além disso foi de diversa serventia a utilização de bibliotecas python, assim tendo acesso a diversas funções nas áreas de manejo de dados e inserção dos mesmos no banco de dados. Dando prosseguimento na descrição do processo, é demonstrado os passos adotados e os procedimentos realizados durante todo o processo.

Figura 23 - Primeiros passos de manejo de dados.

CriacaobaseDados.py

Arquivos finais python > CriacaobaseDados.py > ...

```python
import pandas as pd
#_____________________________  PASSO 1_____________________________________________________
#passo um é pegar os dois dataframes e unificar ele em um so e alinhar as colunas do csv como o esperado

# Carregar os arquivos CSV como DataFrames
df1 = pd.read_csv(r'Arquivos finais python\DadosBase\Dados.csv')
df2 = pd.read_csv(r'Arquivos finais python\DadosBase\EXTRATO DOS ESTABELECIMENTOS COM SERVIÇOS ESPECIALIZADOS.csv')

# Criando o terceiro DataFrame combinando os dados
df3 = pd.merge(df1[['CNES', 'NOME FANTASIA', 'LOGRADOURO' , 'BAIRRO', 'TELEFONE' , 'TURNO DE ATENDIMENTO',
                    'location', 'MUNICÍPIO', 'TIPO DO ESTABELECIMENTO', 'GESTÃO', 'STATUS DO ESTABELECIMENTO']],
               df2[['CNES', 'SERVIÇO']],
               on='CNES',
               how='inner')  # Use 'outer' se quiser incluir todos os CNES presentes em ambos os DataFrames

#convertendo a ordem das colunas para a ordem certa
df4 = df3[['NOME FANTASIA', 'LOGRADOURO' , 'BAIRRO', 'TELEFONE' , 'TURNO DE ATENDIMENTO',
           'location', 'MUNICÍPIO', 'TIPO DO ESTABELECIMENTO', 'GESTÃO', 'STATUS DO ESTABELECIMENTO','CNES','SERVIÇO']]

#criando csv com a base de dados unificada
df4.to_csv('Arquivos finais python\DadosGerados\dados_combinadosV1.csv', index=False)

# Visualizando o resultado
print(df4.head)
print('------------------------------------------------------------')
print(df4.isnull().sum())
for coluna in df4.columns:
    print(f'{coluna}')

```

Fonte: Elaborado pelo autor, (2024).

Como mostrado na figura 23, o primeiro passo, foi importado a biblioteca pandas, carregado na memória as duas bases de dados, como df1 e df2, então df3 é criada selecionado as colunas necessárias, da df1(base de dados geral) é retirado os atributos necessários para o sistema, da df2(base de dados de serviços), é selecionada a coluna de serviço e a coluna de códigos CNES. Contudo o grande diferencial se dá justamente pelo atributo código de cnes,pois com ele fazemos um relacionamento entre as duas bases de dados assim gerando uma base de dados, onde cada referência possui tanto a coluna de serviço prestado,quanto às informações gerais contidas na base mais extensa.

Figura 24 - Base de dados Resultantes.

```
Arquivos finais python > DadosGerados > dados_combinadosV1.csv > data
NOME FANTASIA,LOGRADOURO,BAIRRO,TELEFONE,TURNO DE ATENDIMENTO,location,MUNICÍPIO,TIPO DO ESTABELECIMENTO,GESTÃO,STATUS DO ESTABELECIMENTO,CNES,
EMULTI PILAR,RUA - S/N,CENTRO,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-7.26247,-35.2544",PILAR,71 CENTRO DE APOIO A SAUDE DA FAMILIA,MU
EMULTI,RUA ESTEVAN DINIZ - S/N,CENTRO,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-6.344,-37.747",CATOLE DO ROCHA,02 CENTRO DE SAUDE/UNIDAD
UBSF MATINADAS,RUA DO CRUZEIRO - S/N,ZONA RURAL,(83)33951319,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-7.6675085,-35.6238214",UMBUZEIRO,0
UBSF MATINADAS,RUA DO CRUZEIRO - S/N,ZONA RURAL,(83)33951319,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-7.6675085,-35.6238214",UMBUZEIRO,0
UBSF MATINADAS,RUA DO CRUZEIRO - S/N,ZONA RURAL,(83)33951319,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-7.6675085,-35.6238214",UMBUZEIRO,0
UBSF MATINADAS,RUA DO CRUZEIRO - S/N,ZONA RURAL,(83)33951319,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-7.6675085,-35.6238214",UMBUZEIRO,0
UBS CAIC II,RUA PROJETADA - S/N,SAO FRANCISCO,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-6.344,-37.747",CATOLE DO ROCHA,02 CENTRO DE SAUD
UBS CAIC II,RUA PROJETADA - S/N,SAO FRANCISCO,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-6.344,-37.747",CATOLE DO ROCHA,02 CENTRO DE SAUD
UBS CAIC II,RUA PROJETADA - S/N,SAO FRANCISCO,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-6.344,-37.747",CATOLE DO ROCHA,02 CENTRO DE SAUD
UBS CAIC II,RUA PROJETADA - S/N,SAO FRANCISCO,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-6.344,-37.747",CATOLE DO ROCHA,02 CENTRO DE SAUD
USF 01 JULIO FRANCISCO DE SOUSA,SITIO PITOMBEIRA - S/N,ZONA RURAL,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-8.021155456563902,-36.916809
USF 01 JULIO FRANCISCO DE SOUSA,SITIO PITOMBEIRA - S/N,ZONA RURAL,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-8.021155456563902,-36.916809
USF 01 JULIO FRANCISCO DE SOUSA,SITIO PITOMBEIRA - S/N,ZONA RURAL,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-8.021155456563902,-36.916809
USF 01 JULIO FRANCISCO DE SOUSA,SITIO PITOMBEIRA - S/N,ZONA RURAL,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-8.021155456563902,-36.916809
USF 01 JULIO FRANCISCO DE SOUSA,SITIO PITOMBEIRA - S/N,ZONA RURAL,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-8.021155456563902,-36.916809
USF 01 JULIO FRANCISCO DE SOUSA,SITIO PITOMBEIRA - S/N,ZONA RURAL,,03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE,"-8.021155456563902,-36.916809
```

Fonte: Elaborado pelo autor, (2024).

Contudo como visto na figura 24, apesar do funcionamento da união das duas bases de dados, houve a repetição das mesmas unidades de saúde, mas cada uma com um serviço de saúde diferente. Isso ocorre pois alguns estabelecimentos oferecem mais de um serviço, e durante a união dos dados o resultado é esse. Além disso, temos a coluna “location”, com as coordenadas em uma só coluna, contudo o frontend necessita delas separadas.

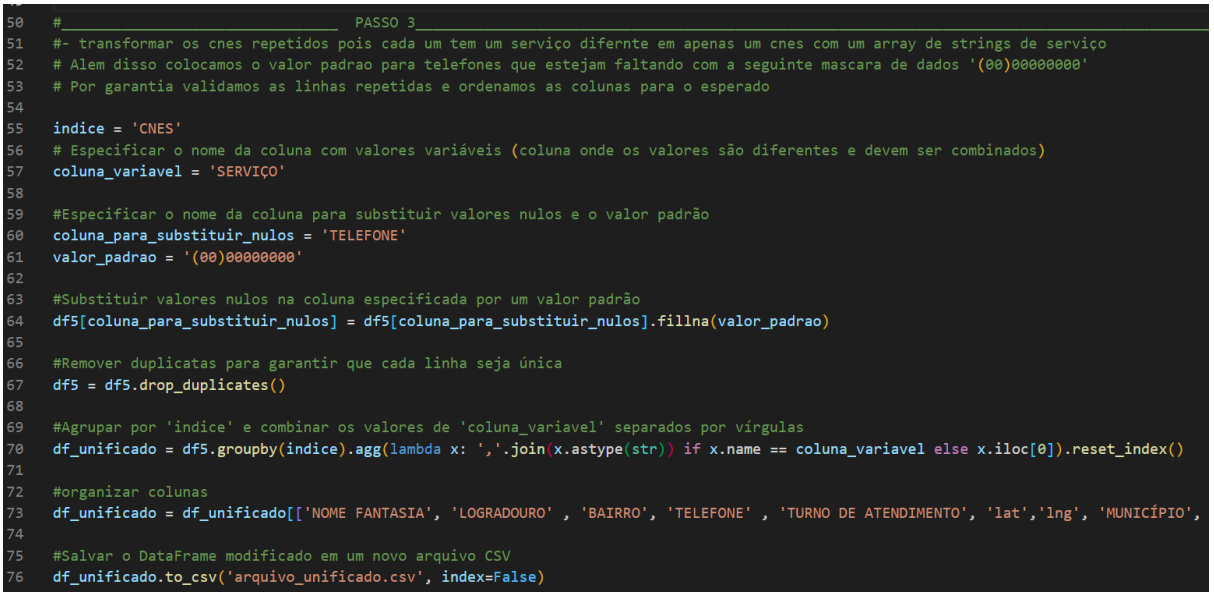
Figura 25 - Segundo passo de manejo de dados.

```python
#_______________________________ PASSO 2_______________________________________
#separar a coluna location que póssui o ponto e virgula em long lat

#separando a coluna location em colunas lat e lgn atraves do delimitador ,
df4[['lat','lng']] = df4['location'].str.split(',', expand=True)

#ordenando as colunas da forma necessaria
df5 = df4[['NOME FANTASIA', 'LOGRADOURO' , 'BAIRRO', 'TELEFONE' , 'TURNO DE ATENDIMENTO', 'lat','lng', 'MUNICÍPIO',
           'TIPO DO ESTABELECIMENTO', 'GESTÃO', 'STATUS DO ESTABELECIMENTO','CNES','SERVIÇO']]

#verificandoa ordem das colunas e se esta tudo correto
for coluna in df5.columns:
    print(f'{coluna}')

#verificando se existe algum dado faltando nas novas colunas
print(df5.isnull().sum())

#criação do csv apos o segundo passo
df5.to_csv('Arquivos finais python\DadosGerados\dados_combinadosV2.csv', index=False)

```

Fonte: Elaborado pelo autor, (2024).

No segundo passo demonstrado na figura 25, é realizada a partição da coluna "location“, em “lat” e “lng”, como é esperado no sistema, além disso é realizada a reordenação das colunas na base de dados.

Figura 26 - Terceiro passo de manejo dos dados.



```
#_____ PASSO 3_____
#- transformar os cnes repetidos pois cada um tem um serviço difernte em apenas um cnes com um array de strings de serviço
# Alem disso colocamos o valor padrao para telefones que estejam faltando com a seguinte mascara de dados '(00)00000000'
# Por garantia validamos as linhas repetidas e ordenamos as colunas para o esperado

indice = 'CNES'
# Especificar o nome da coluna com valores variáveis (coluna onde os valores são diferentes e devem ser combinados)
coluna_variavel = 'SERVIÇO'

#Especificar o nome da coluna para substituir valores nulos e o valor padrão
coluna_para_substituir_nulos = 'TELEFONE'
valor_padrao = '(00)00000000'

#Substituir valores nulos na coluna especificada por um valor padrão
df5[coluna_para_substituir_nulos] = df5[coluna_para_substituir_nulos].fillna(valor_padrao)

#Remover duplicatas para garantir que cada linha seja única
df5 = df5.drop_duplicates()

#Agrupar por 'indice' e combinar os valores de 'coluna_variavel' separados por vírgulas
df_unificado = df5.groupby(indice).agg(lambda x: ','.join(x.astype(str)) if x.name == coluna_variavel else x.iloc[0]).reset_index()

#organizar colunas
df_unificado = df_unificado[['NOME FANTASIA', 'LOGRADOURO' , 'BAIRRO', 'TELEFONE' , 'TURNO DE ATENDIMENTO', 'lat','lng', 'MUNICÍPIO',

#Salvar o DataFrame modificado em um novo arquivo CSV
df_unificado.to_csv('arquivo_unificado.csv', index=False)
```

Fonte: Elaborado pelo autor, (2024).

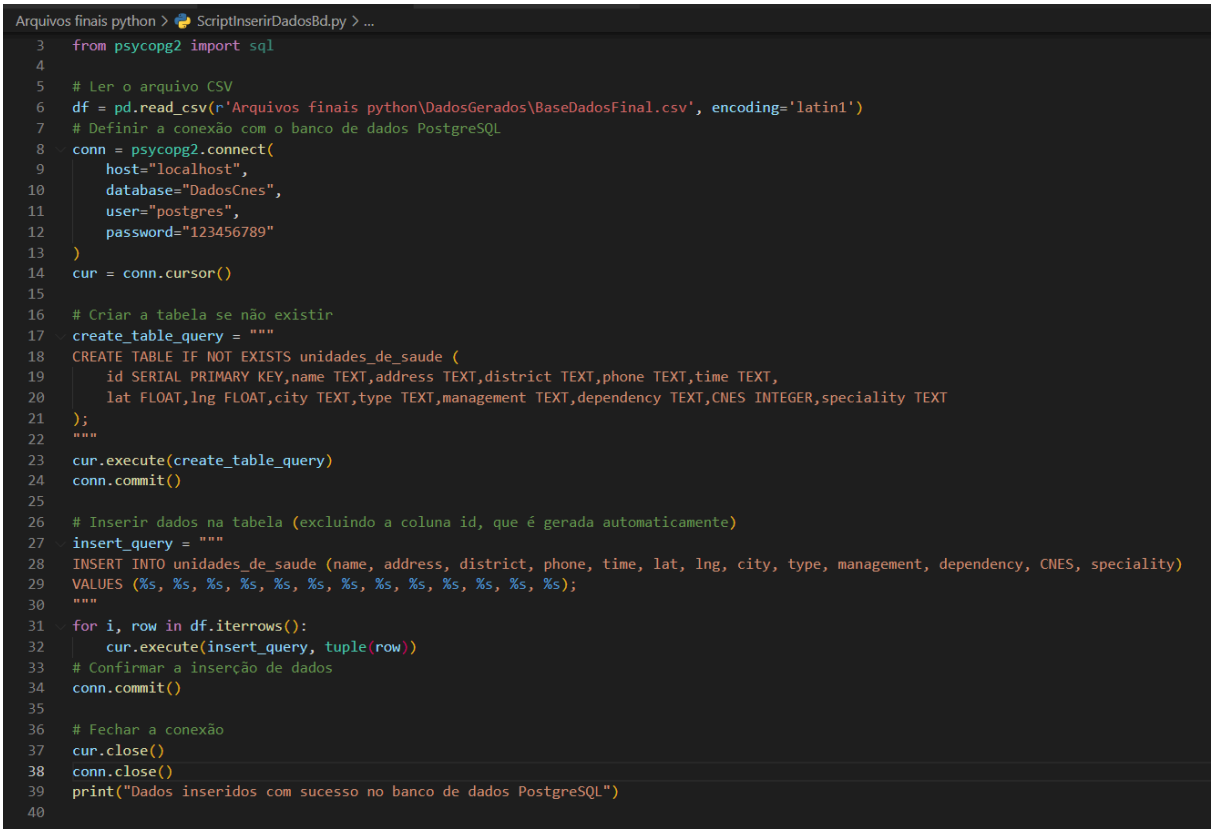
Por fim, chegando no terceiro e último passo de manejo dos dados, como mostrado na figura 26, é aplicada uma máscara para os campos vazios da coluna telefone, pois foi notado a ausência em diversas unidade de saúde a ausência desse atributo, e visando sanar futuras instabilidades causadas por campos vazios, foi aplicada essa máscara. Ademais, como último empecilho da base de dados temos unidades de saúdes repetidas com serviços diferentes, utilizando uma função específica para unificar essas linhas em um só atributo, é solucionado esse problema. Assim finalizando a etapa de criação do arquivo de dados, uma base de dados que cumpre os requisitos do sistema.

5.5 Inserção da Base Final de Dados no Banco Utilizando o Python

Após a finalização do processo de construção de um arquivo de dados referentes à unidade de saúde, chegou a hora de realizar a inserção desses dados no banco de dados. Utilizando la biblioteca “psycopg2”, criamos uma conexão com o banco de dados Postgresql, o banco de dados escolhido para o ecossistema do programa, faz a verificação da existência de uma tabela já existente no banco de dados, é realizada a inserção do arquivo de dados no banco de dados, a apresentação desse script de pode ser observado em código na figura 27.

Concluindo finalmente passamos a utilizar o *Python*, e o manejo das bases de dados.

Figura 27 - Script para Inserção de dados no PostgreSQL.



```
Arquivos finais python > ScriptInserirDadosBd.py > ...
3   from psycopg2 import sql
4
5   # Ler o arquivo CSV
6   df = pd.read_csv(r'Arquivos finais python\DadosGerados\BaseDadosFinal.csv', encoding='latin1')
7   # Definir a conexão com o banco de dados PostgreSQL
8   conn = psycopg2.connect(
9       host="localhost",
10      database="DadosCnes",
11      user="postgres",
12      password="123456789"
13  )
14  cur = conn.cursor()
15
16  # Criar a tabela se não existir
17  create_table_query = """
18  CREATE TABLE IF NOT EXISTS unidades_de_saude (
19      id SERIAL PRIMARY KEY,name TEXT,address TEXT,district TEXT,phone TEXT,time TEXT,
20      lat FLOAT,lng FLOAT,city TEXT,type TEXT,management TEXT,dependency TEXT,CNES INTEGER,speciality TEXT
21  );
22  """
23  cur.execute(create_table_query)
24  conn.commit()
25
26  # Inserir dados na tabela (excluindo a coluna id, que é gerada automaticamente)
27  insert_query = """
28  INSERT INTO unidades_de_saude (name, address, district, phone, time, lat, lng, city, type, management, dependency, CNES, speciality)
29  VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);
30  """
31  for i, row in df.iterrows():
32      cur.execute(insert_query, tuple(row))
33  # Confirmar a inserção de dados
34  conn.commit()
35
36  # Fechar a conexão
37  cur.close()
38  conn.close()
39  print("Dados inseridos com sucesso no banco de dados PostgreSQL")
40
```

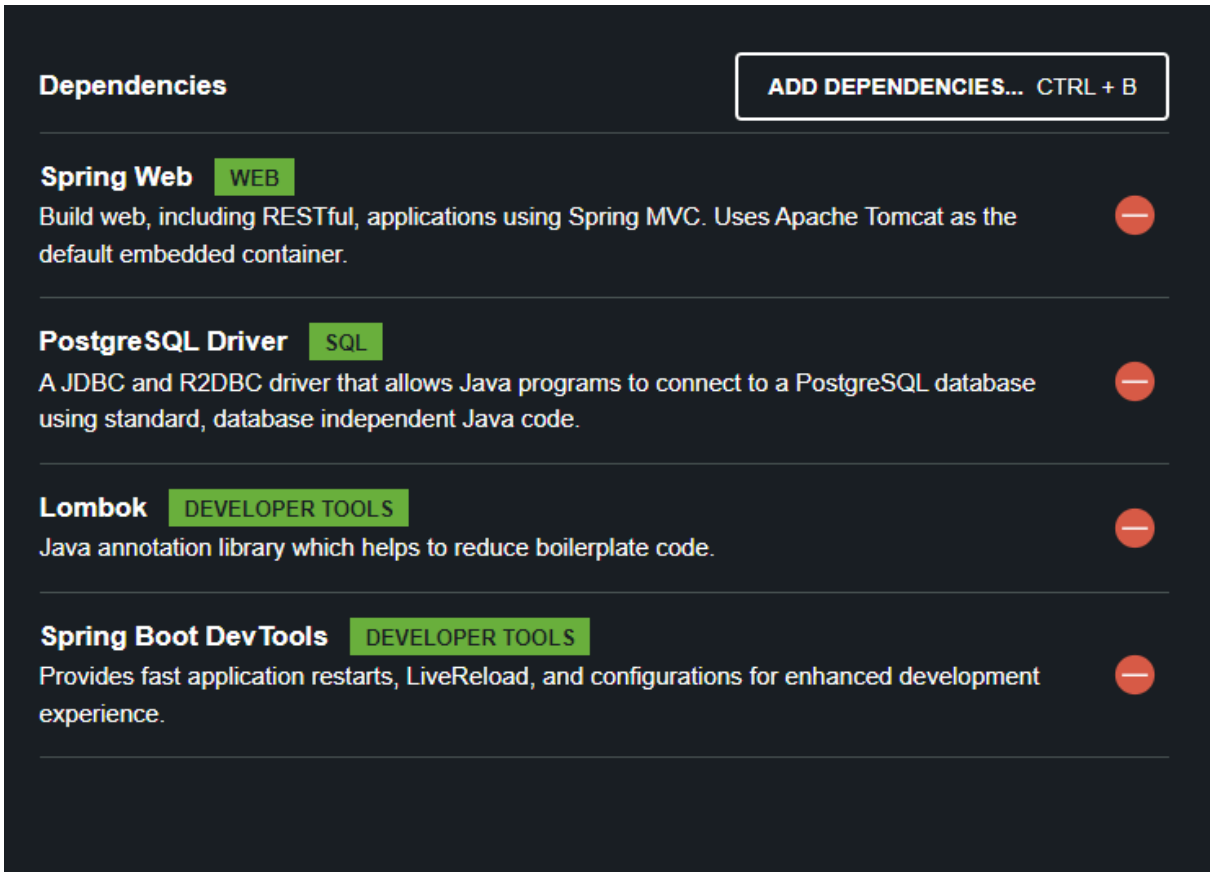
Fonte: Elaborado pelo autor, (2024).

5.6 Desenvolvimento da API Back-End de Fornecimento de Dados

Após a etapa de alimentação do banco de dados com uma base, chega a hora de focar os esforços para criação da API responsável por interagir com esses dados.

Partindo da criação do arquivo do projeto, foram adicionados algumas dependências, como mostrado na figura 28, inicialmente apenas as dependências essenciais foram inseridos, o PostgreSQL para realizar a conexão com o banco de dados, o Lombok, que tratasse de um biblioteca de anotações para reduzir código padrão, e o Spring Web e Dev tools para nos fornecer configurações iniciais do projeto.

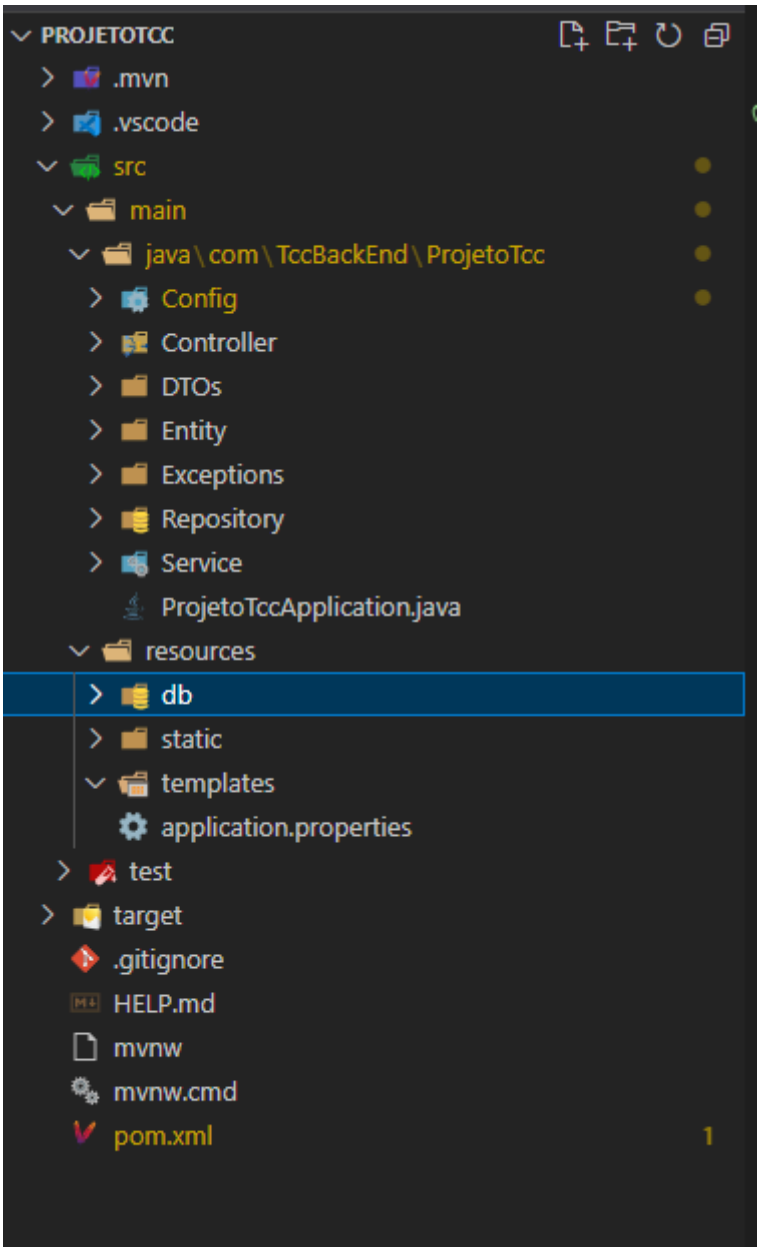
Figura 28 - Dependências iniciais do projeto.



Fonte: Elaborado pelo autor, (2024).

Após a criação do arquivo de projeto, foi importado para o editor de código e em seguida já foi feita a estruturação de pastas. Visando a organização do projeto desde o início e respeitando a arquitetura de projeto, o fluxo de trabalho foi facilitado.

Figura 29 - Estrutura de pastas do projeto.



Fonte: Elaborado pelo autor, (2024).

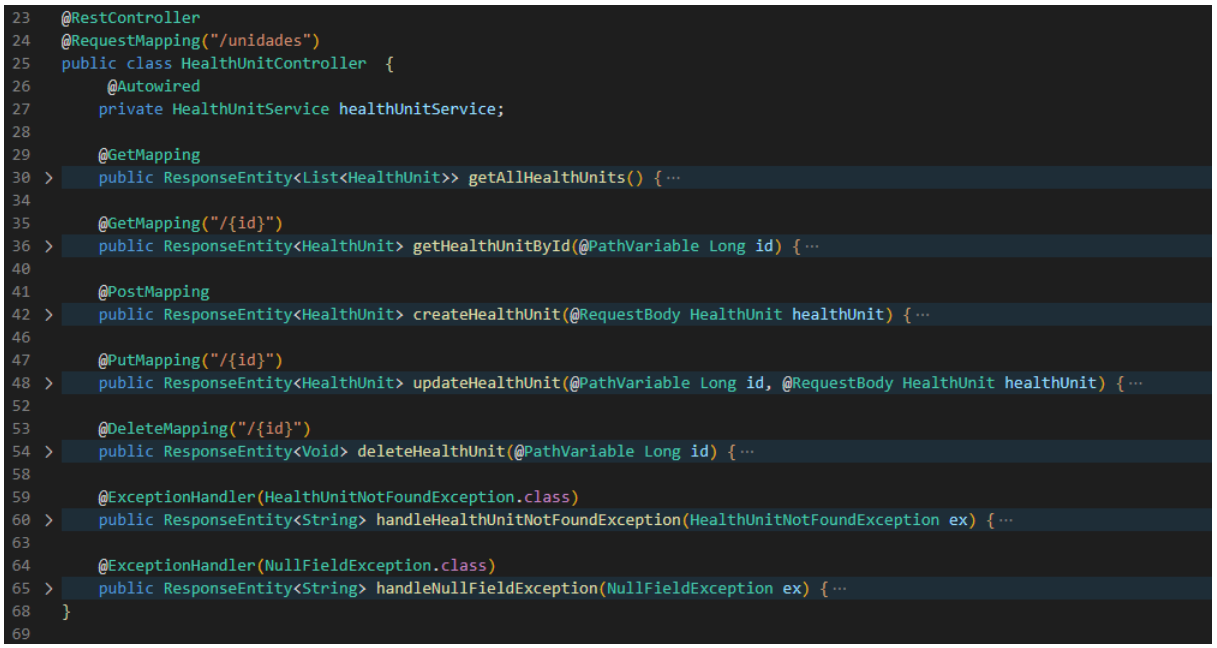
Como apresentado na figura 29,temos uma estrutura de pastas interessante, nela consiste:

- Pasta Config: Onde serão agrupados todos os arquivos de código referente a configurações, como arquivos de segurança, configurações web e configuração de documentação.
- Pasta Controller: Nesta pasta serão agrupados os controladores da aplicação, parte do código responsável por configurar as rotas de requisições e para onde direcionar essas solicitações.

- DTOs: Traduzido para Objeto de Transferência de Dado, possui a lógica de mapear um dado para uma formatação específica assim garantido a formatação do dado e a segurança.
- Entity: Pasta responsável por armazenar o modelo de dados que será movimentado pelo banco de dados, e até para modulação dos dados para as requisições.
- Exceptions: Pasta responsável por agrupar as exceções específicas da aplicação.
- Repository: Pasta responsável por armazenar os repositórios, principalmente as interfaces que fornecem as funções de manter os dados no banco de dados.
- Service: Criando uma camada extra para desacoplar a camada de serviço, temos a camada service, nela temos as partes de códigos com funções de regras de negócio e que serão utilizados na camada de controladores.
- Resources: Nela há a subpasta DB, que possui arquivos de manutenção e conexão referentes ao banco de dados.

Dando prosseguimento ao processo de geração de código, foi feita a conexão da API Back-end Spring com o banco de dados PostgreSQL. Foi Encapsulado os métodos de manter as unidades de saúde, isto é criar,excluir, visualizar e alterar, na camada de serviço e como mostrado na figura 30, demonstra um controlador da API capaz de realizar todas essas funções, através da comunicação no protocolo http, de acordo com a rota da requisição e do seu verbo. Com um código de fácil visualização e interpretação, ainda é apresentado o uso de métodos para captura de exceções geradas por requisições inconsistentes.

Figura 30 - Arquivo do controlador das rotas de manter unidades de saúde.

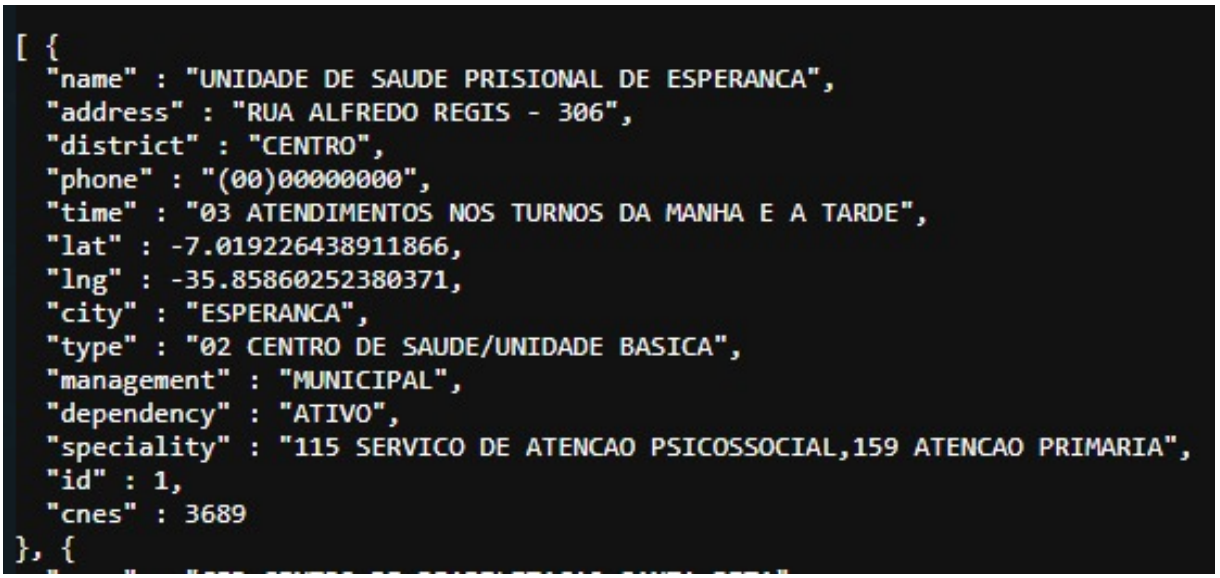


```
@RestController
@RequestMapping("/unidades")
public class HealthUnitController  {
     @Autowired
    private HealthUnitService healthUnitService;

    @GetMapping
    public ResponseEntity<List<HealthUnit>> getAllHealthUnits() {···

    @GetMapping("/{id}")
    public ResponseEntity<HealthUnit> getHealthUnitById(@PathVariable Long id) {···

    @PostMapping
    public ResponseEntity<HealthUnit> createHealthUnit(@RequestBody HealthUnit healthUnit) {···

    @PutMapping("/{id}")
    public ResponseEntity<HealthUnit> updateHealthUnit(@PathVariable Long id, @RequestBody HealthUnit healthUnit) {···

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteHealthUnit(@PathVariable Long id) {···

    @ExceptionHandler(HealthUnitNotFoundException.class)
    public ResponseEntity<String> handleHealthUnitNotFoundException(HealthUnitNotFoundException ex) {···

    @ExceptionHandler(NullFieldException.class)
    public ResponseEntity<String> handleNullFieldException(NullFieldException ex) {···
}
```

Fonte: Elaborado pelo autor, (2024).

Finalmente realizando as primeiras execuções do código foi iniciada a aplicação, sem problemas durante a compilação e execução do código. Entretanto uma situação peculiar se tornou o foco da análise. A seguir é mostrado o resultado da visualização dos dados no navegador.

Figura 31 - JSON retornado ao navegador pela aplicação.




```
[ {
  "name" : "UNIDADE DE SAUDE PRISIONAL DE ESPERANCA",
  "address" : "RUA ALFREDO REGIS - 306",
  "district" : "CENTRO",
  "phone" : "(00)00000000",
  "time" : "03 ATENDIMENTOS NOS TURNOS DA MANHA E A TARDE",
  "lat" : -7.019226438911866,
  "lng" : -35.85860252380371,
  "city" : "ESPERANCA",
  "type" : "02 CENTRO DE SAUDE/UNIDADE BASICA",
  "management" : "MUNICIPAL",
  "dependency" : "ATIVO",
  "speciality" : "115 SERVICO DE ATENCAO PSICOSSOCIAL,159 ATENCAO PRIMARIA",
  "id" : 1,
  "cnes" : 3689
}, {
```


Fonte: Elaborado pelo autor, (2024).

Como mostrado na figura 31, a ordem dos atributos no arquivos JSON, estavam dispostos de uma forma irregular,foi então iniciada a busca pelo causador dessa inconsistência.

Figura 32 - Classe de modelo de unidades de saúde utilizando a serialização JSON.

```java
@Table(name = "unidades_de_saude")
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonPropertyOrder({ "id", "name", "address", "district",
                     "phone", "time", "lat", "lng", "city", "type",
                     "management", "dependency", "cnes", "speciality" })

public class HealthUnit {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "name")
    private String name;

    @Column(name = "address")
    private String address;

    @Column(name = "district")
    private String district;

    @Column(name = "phone")
    private String phone;

    @Column(name = "time")
    private String time;

    @Column(name = "lat")
    private Double lat;

    @Column(name = "lng")
    private Double lng;
```

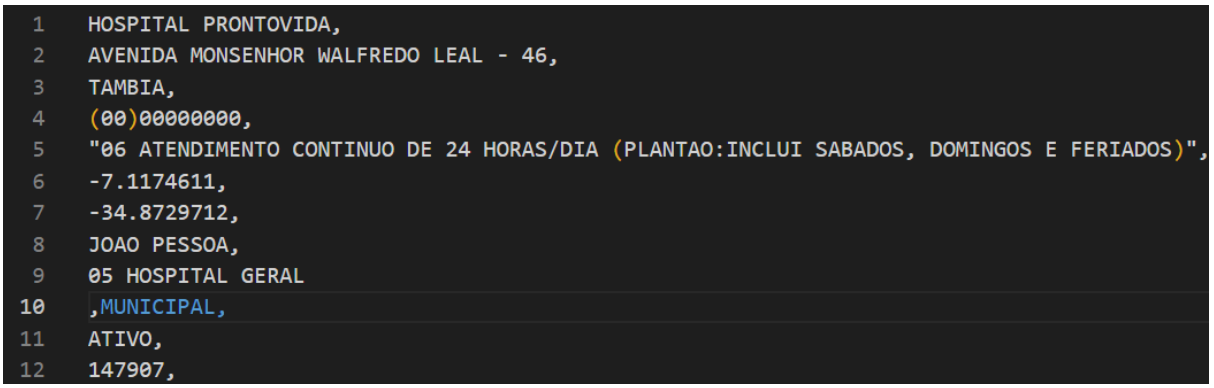
Fonte: Elaborado pelo autor, (2024).

Buscando garantir a serialização do JSON, ou seja, a ordem correta dos atributos, foi feita a análise na classe de modelo de entidade das unidades de saúde, nenhuma inconsistência aparente foi localizado, foi então adicionado a notação de um serializador do Spring, visando garantir a ordem correta dos atributos, como

mostrado na figura 32, Entretanto, não houve alteração na ordem da disposição dos dados no navegador. A análise e processo de depuração do código se intensificou.

Foi então focado a análise na base de dados, só então foi encontrado a seguinte inconsistência.

Figura 33 - Modelo de dados do arquivo de unidades de saúde.



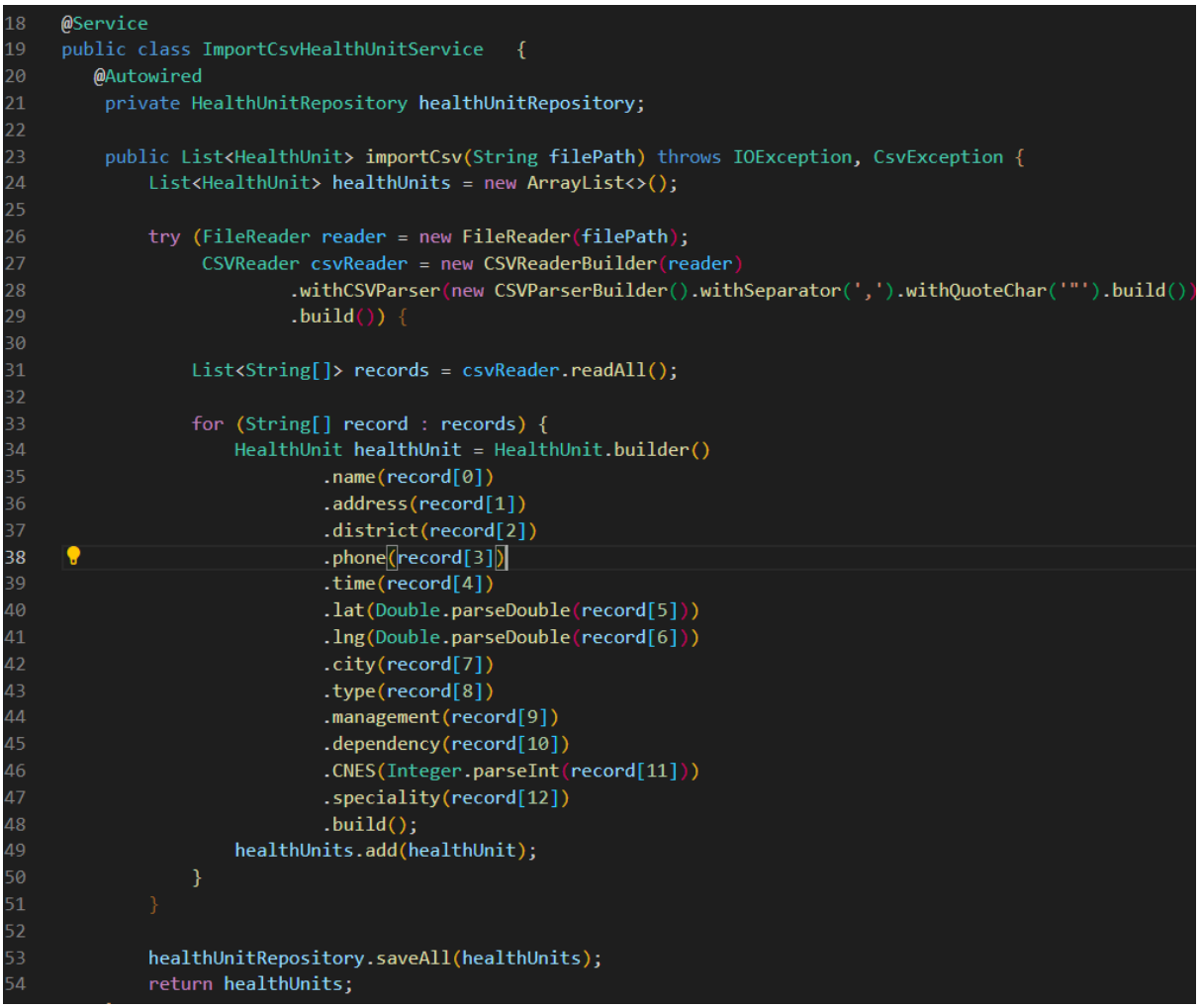
Fonte: Elaborado pelo autor, (2024).

Como apresentado na imagem 33, na linha 5, é observado que dentro desse atributo, há tanto o uso de aspas, como de parênteses. Isso pode ter ocorrido em decorrência do preenchimento incorreto dos dados. Apesar de parecer um erro pequeno, essa simples inconsistência gerou um efeito cascata em toda interpretação de código. Por ser um arquivo do tipo csv, ao ser interceptado o sistema analisa essas vírgulas como separador de colunas, gerando assim um descontrole na criação dos dados que eram utilizados na aplicação.

Para corrigir esse problema, foi inserido no script de inserção uma verificação para analisar o uso de aspas, assim o sistema será capaz de interpretar as vírgulas contidas entre aspas como conteúdo de texto e não como delimitadores de colunas.

Além disso, foi implementada a funcionalidade de inserção de base de dados através de um controlador, como mostrado na figura 34, recebendo o endereço dos dados o sistema fazia a interpretação do arquivo csv e inseri no banco de dados de forma automática. Corrigindo o código para que não ocorra o mesmo problema na interpretação dos dados, foi utilizado um método para fazer essa mesma interpretação dos aspas e outros caracteres especiais. Como resultado dessa implementação obtivemos o seguinte método na camada de serviço.

Figura 34 - Função de serviço de inserção de dados utilizando a API.



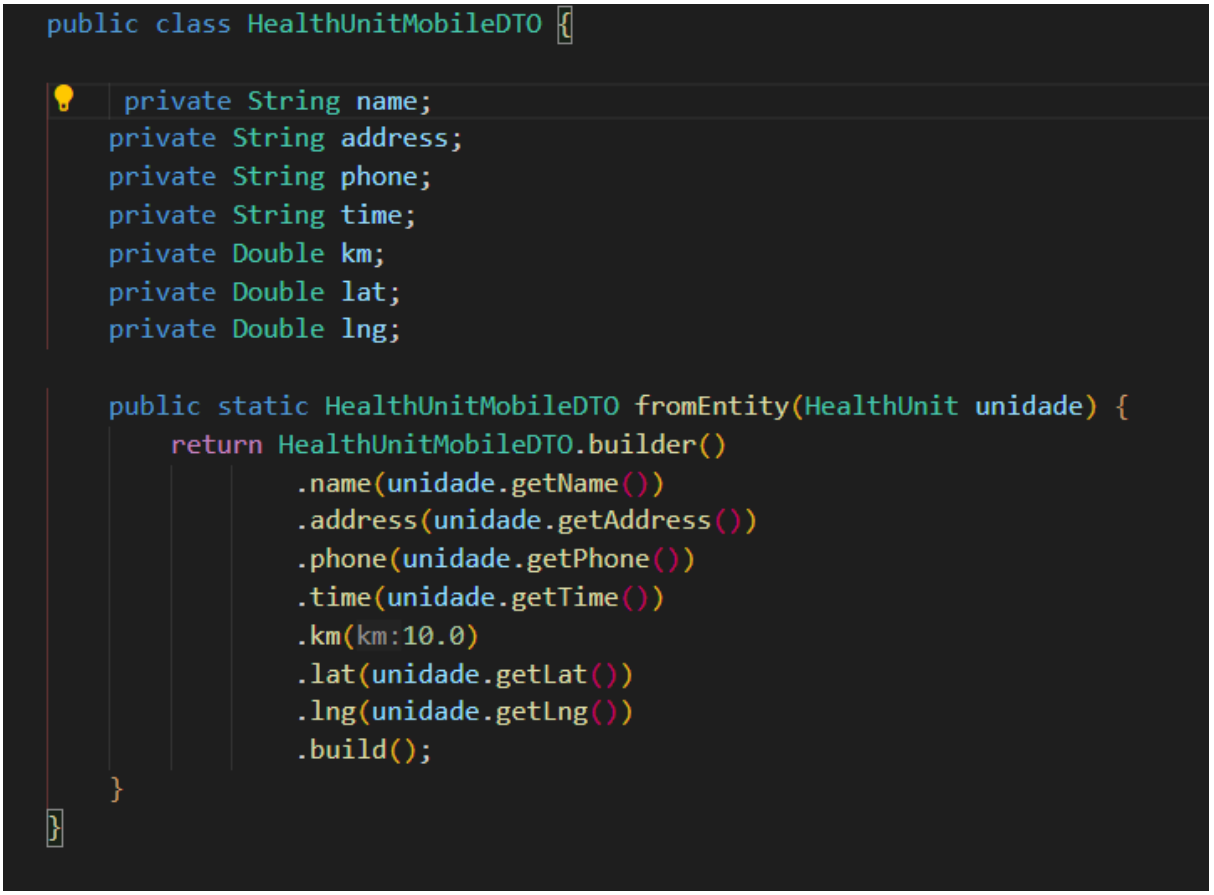
```java
@Service
public class ImportCsvHealthUnitService   {
   @Autowired
    private HealthUnitRepository healthUnitRepository;

    public List<HealthUnit> importCsv(String filePath) throws IOException, CsvException {
        List<HealthUnit> healthUnits = new ArrayList<>();

        try (FileReader reader = new FileReader(filePath);
             CSVReader csvReader = new CSVReaderBuilder(reader)
                     .withCSVParser(new CSVParserBuilder().withSeparator(',').withQuoteChar('"').build())
                     .build()) {

            List<String[]> records = csvReader.readAll();

            for (String[] record : records) {
                HealthUnit healthUnit = HealthUnit.builder()
                        .name(record[0])
                        .address(record[1])
                        .district(record[2])
                        .phone(record[3])
                        .time(record[4])
                        .lat(Double.parseDouble(record[5]))
                        .lng(Double.parseDouble(record[6]))
                        .city(record[7])
                        .type(record[8])
                        .management(record[9])
                        .dependency(record[10])
                        .CNES(Integer.parseInt(record[11]))
                        .speciality(record[12])
                        .build();
                healthUnits.add(healthUnit);
            }
        }

        healthUnitRepository.saveAll(healthUnits);
        return healthUnits;
```

Fonte: Elaborado pelo autor, (2024).

Em seguida foi adicionado a funcionalidade voltada ao fornecimento de dados de unidades de saúde, porém, com dados para um cliente mobile, onde possui um repertório de atributos reduzidos. Para isso foi criado uma classe de *DTO*, um objeto de transferência de dados, que mapeasse os atributos necessários e assim criasse um modelo de dado condizente com o esperado pela requisição.

Em seguida foi construído um controlador específico para lidar com esse manejo de dados específicos, contudo como esse tipo de cliente não possui permissões mais completas de manter os dados, foi necessário apenas a as rotas de visualização desses dados. Os resultados são demonstrados nas figuras 35 e 36.

Figura 35 - Arquivo de mapeamento de unidade de saúde no padrão cliente mobile.



```
public class HealthUnitMobileDTO {

    private String name;
    private String address;
    private String phone;
    private String time;
    private Double km;
    private Double lat;
    private Double lng;

    public static HealthUnitMobileDTO fromEntity(HealthUnit unidade) {
        return HealthUnitMobileDTO.builder()
                .name(unidade.getName())
                .address(unidade.getAddress())
                .phone(unidade.getPhone())
                .time(unidade.getTime())
                .km(km:10.0)
                .lat(unidade.getLat())
                .lng(unidade.getLng())
                .build();
    }
}
```

Fonte: Elaborado pelo autor, (2024).

Figura 36 - Controlador de acesso das unidades de saúde versão mobile.

```
@RestController
@RequestMapping("/mobile")
public class HealthUnitMobileController {

    @Autowired
    private HealthUnitService healthUnitService;

    @GetMapping("/")
    public List<HealthUnitMobileDTO> getAllHealthUnitMobile() {···

    @GetMapping("/{id}")
    public HealthUnitMobileDTO getHealthUnitMobile(@PathVariable Long id) {···
```

Fonte: Elaborado pelo autor, (2024).

Passando para a próxima etapa de codificação, a fim de cumprir os requisitos coletados, além de elevar o nível da aplicação. Foram adicionados mais alguns starter, no intuito de adicionar uma camada de segurança a aplicação, com o Spring Security. E a adição da documentação swagger, para facilitar a visualização da documentação e até mesmo executar comandos de testes e execuções da aplicação. A adição tanto do swagger como o spring security se deu através de adição de starters de dependências, e a criação de alguns arquivos de configurações necessários para seu funcionamento.

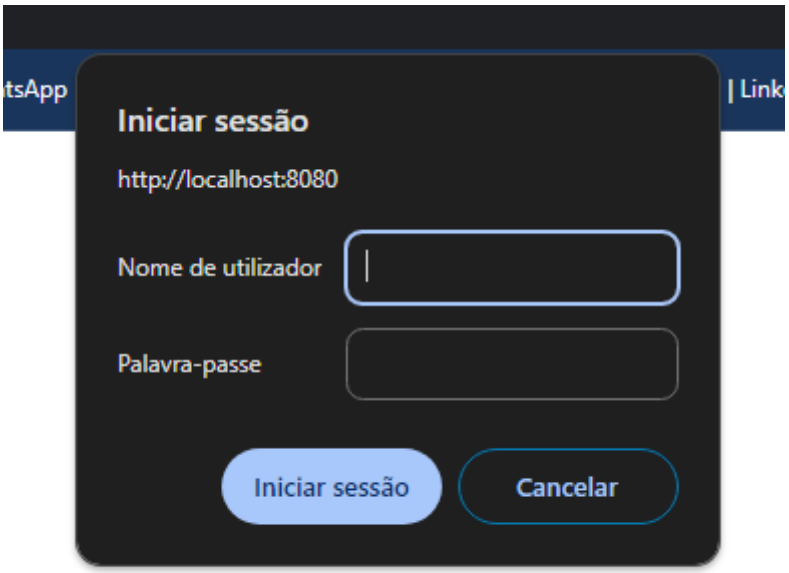
A seguir na figura 37 é demonstrado os arquivos de dependências. Após inseridas e devidamente configuradas obtemos os resultados apresentados nas figuras 38 e 39.

Figura 37 - Adição de dependências de segurança e documentação com swagger.

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.3.0</version>
</dependency>

<!-- Bibliotecas para segurança da aplicação -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
```

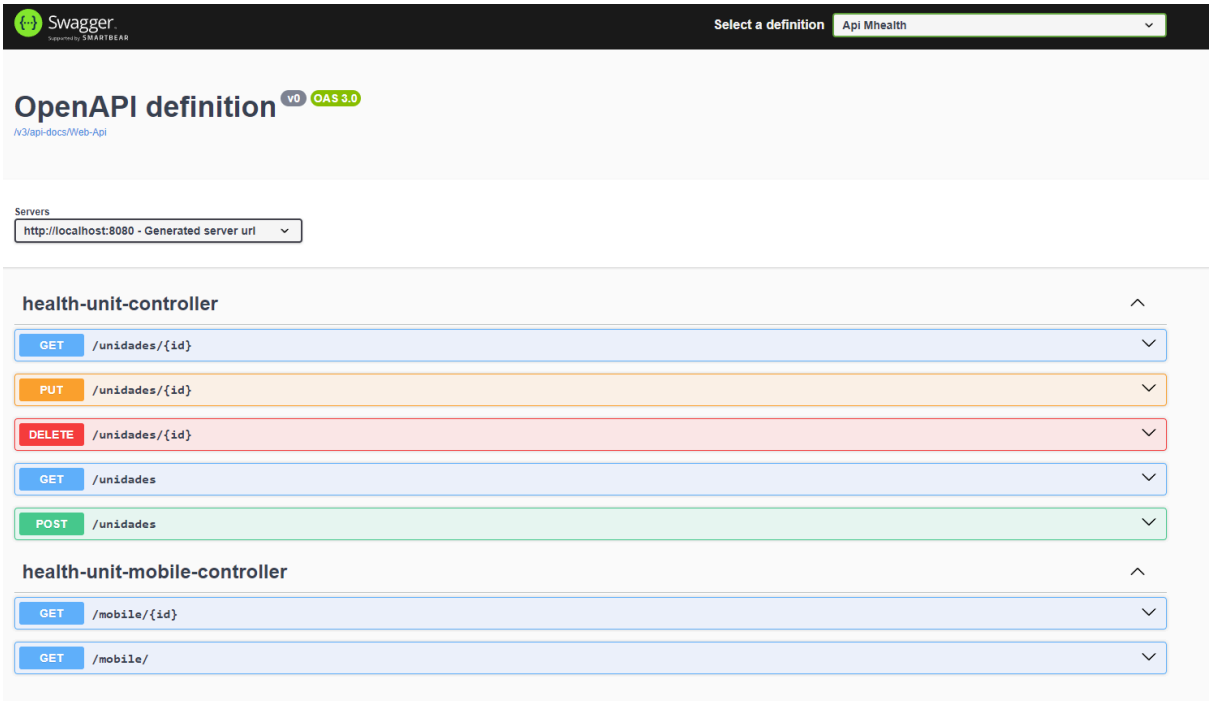
Fonte: Elaborado pelo autor, (2024).

Figura 38 - Adição de autenticação para acesso às unidades de saúde.



Fonte: Elaborado pelo autor, (2024).

Figura 39 - Apresentação do uso do Swagger.



Fonte: Elaborado pelo autor, (2024).

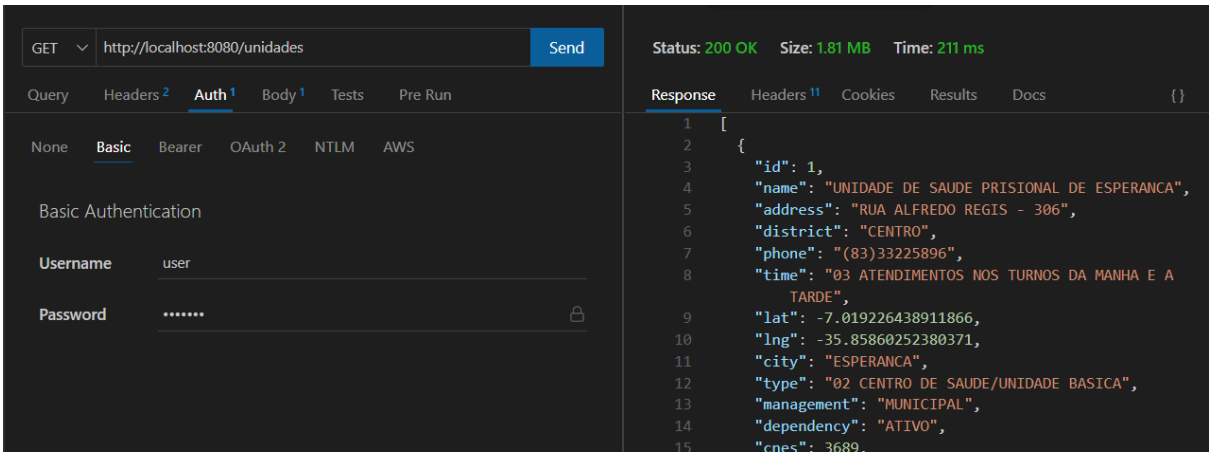
6 RESULTADOS

Com a finalização do processo de desenvolvimento, foi obtida uma *API Back-end* capaz de atender os requisitos fornecidos pela análise do trabalho de VENTURA(2022).O resultado foi uma *API* robusta e desenvolvida seguindo orientações de boas práticas de implementação. Além disso, houve a adição de funcionalidades visando facilitar o manejo dos dados que elevou o nível da aplicação para além de funções básicas de manter as unidades de saúde, isto é, criar, visualizar,editar e excluir. Além disso, um espaço de destaque neste trabalho se deu pela análise de dados, utilizando a linguagem *Python*.

Finalmente, foi concluída a integração com o sistema existente desenvolvido por VENTURA(2022).Nesta seção, serão apresentados esses resultados por meio de simulações de fluxos de requisições da *API back-end*, tanto no navegador utilizando a ferramenta *Swagger*, como em uma simulação de cliente utilizando a extensão do *visual studio code*, o *Thunder Client*, para disparar requisições na *API*. Por fim, será demonstrado o front-end consumindo esses dados e exibindo-os.

A figura 40, ilustra a realização de uma consulta de unidades de saúde, para tal é necessário a inserção das credenciais de autenticação. Como resultado obtido é retornado um *JSON*, com *status* ok, e uma listagem das unidades de saúde, com as informações corretas e serializadas.

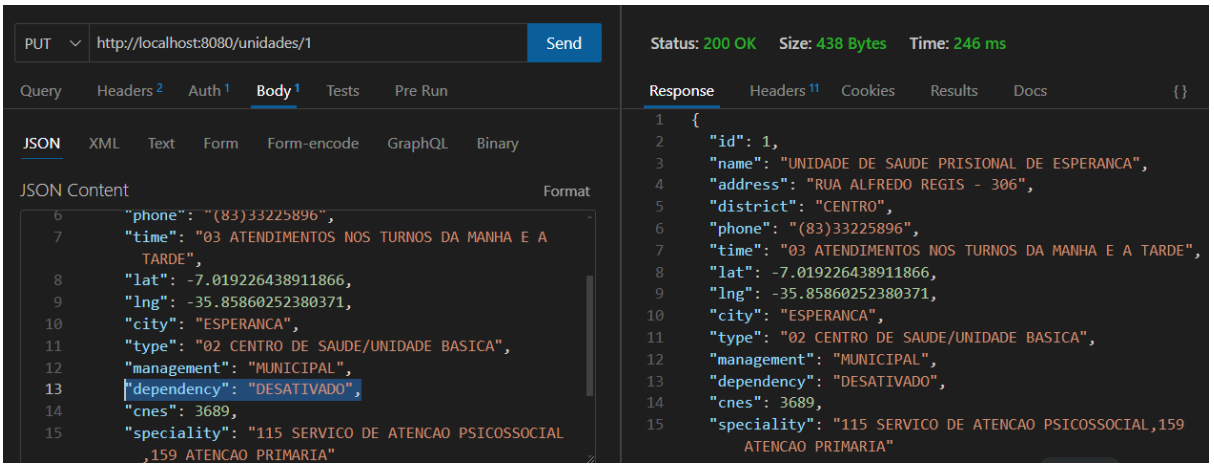
Figura 40 - Realização de requisição de visualização de dados.



Fonte: Elaborado pelo autor, (2024).

Também é realizada uma requisição onde podemos fazer uma alteração em uma informação específica, como mostrado na figura 41.

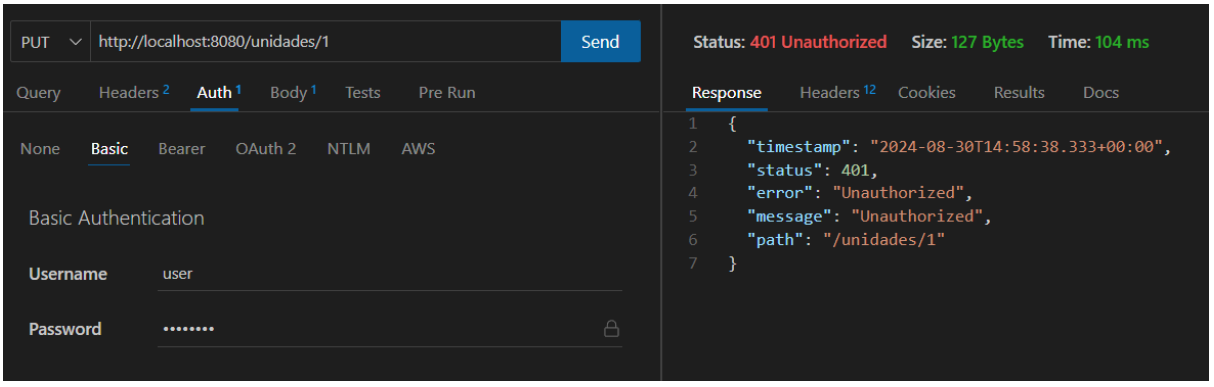
Figura 41 - Alteração de uma unidade de saúde específica.



Fonte: Elaborado pelo autor, (2024).

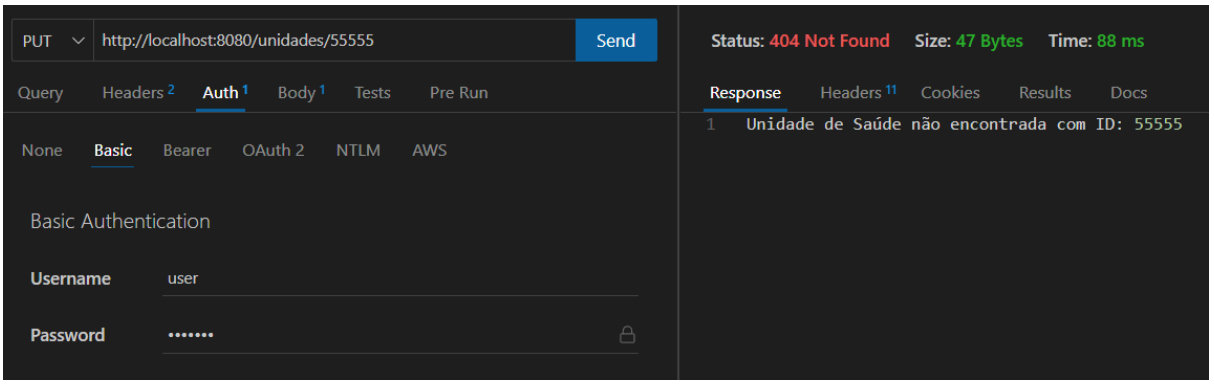
Contudo o sistema também consegue lidar com inconsistências nas requisições, sejam elas, problemas na autenticação como mostrado na figura 42, ou até mesmo a tentativa de acesso a informações não existentes nas bases de dados como mostrado na figura 43.

Figura 42 - Tentativa de acesso com credenciais inválidas.



Fonte: Elaborado pelo autor, (2024).

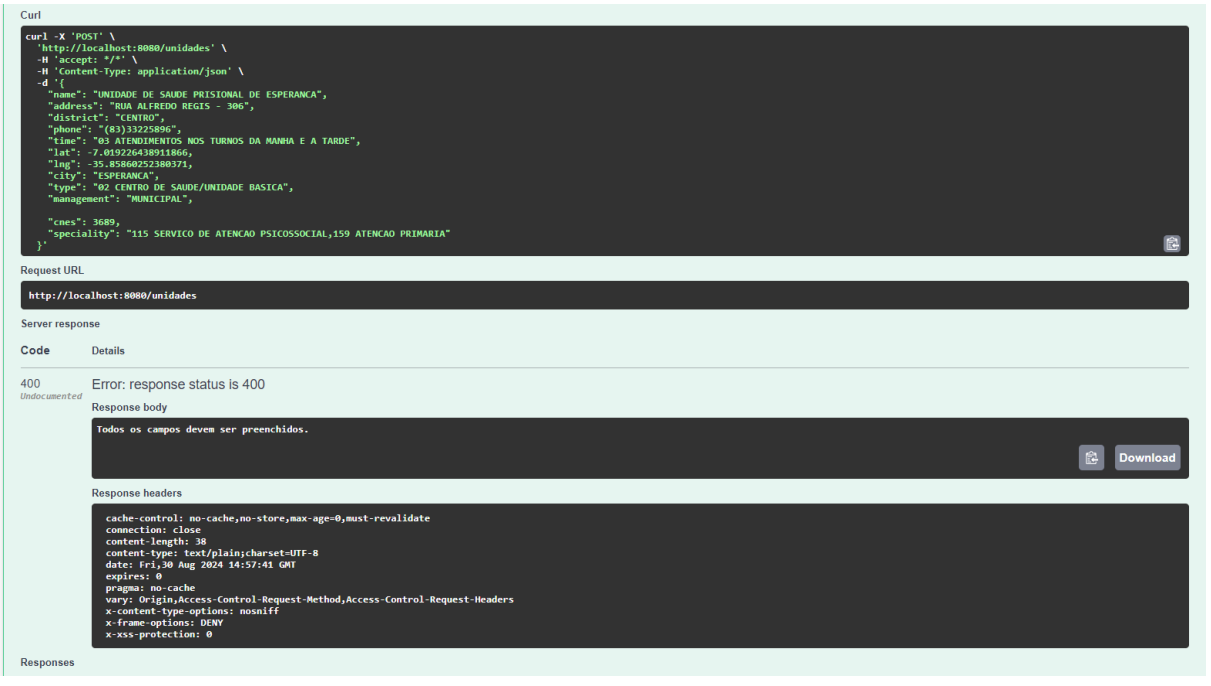
Figura 43 - Tentativa de acesso a uma unidade de saúde inexistente.



Fonte: Elaborado pelo autor, (2024).

Outra funcionalidade se dá pela verificação extra exercida na aplicação de campos vazios, para diminuir os problemas com controle de dados, como mostrado na figura 44, nessa figura também é apresentado o uso do recurso do *Swagger.* Mostrando a sua praticidade e facilidade de interpretação de uso da *API*. A ausência de um campo dispara uma mensagem específica, assim como em outros trechos as exceções são tratadas de acordo com o problema ocorrido, e fornece uma mensagem mais agradável e explicativa quanto ao erro ocorrido.

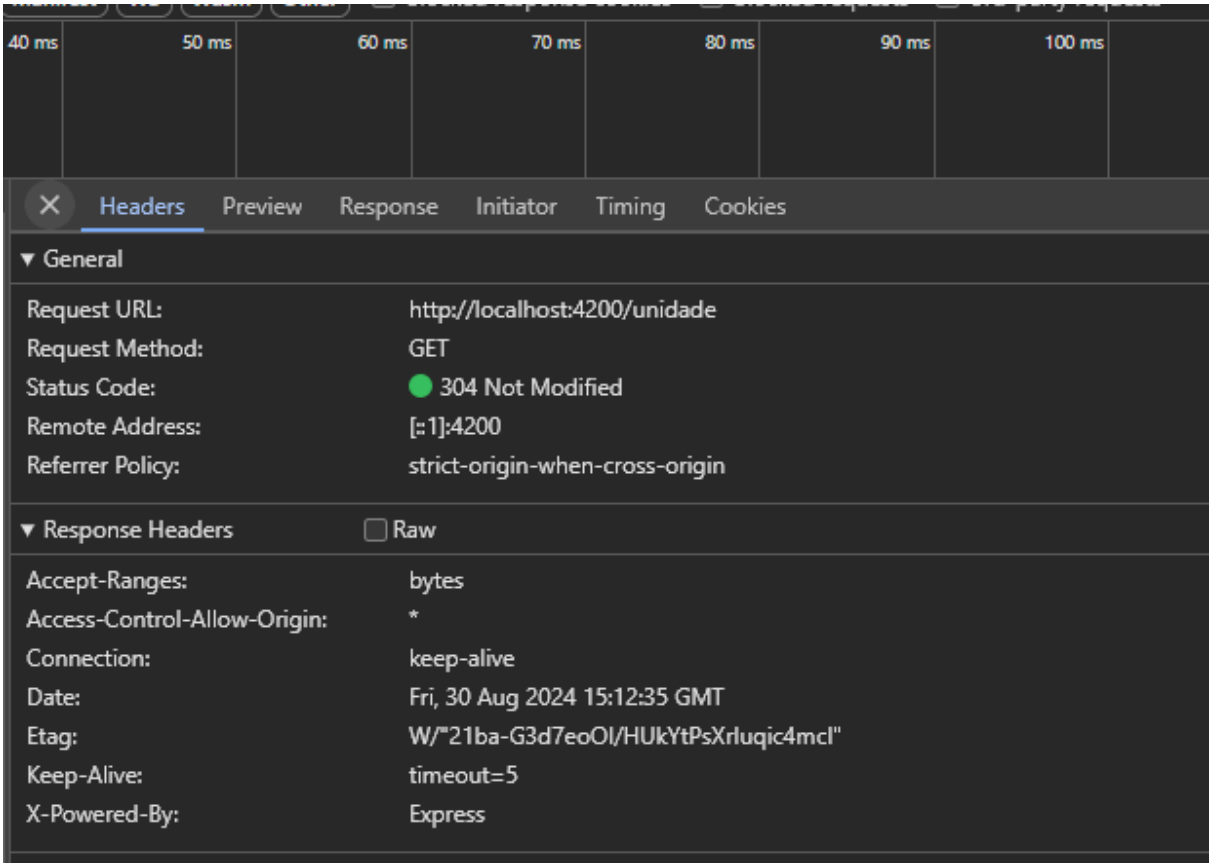
Figura 44 - Consulta realizada no Swagger com ausência de alguns parâmetros.



Fonte: Elaborado pelo autor, (2024).

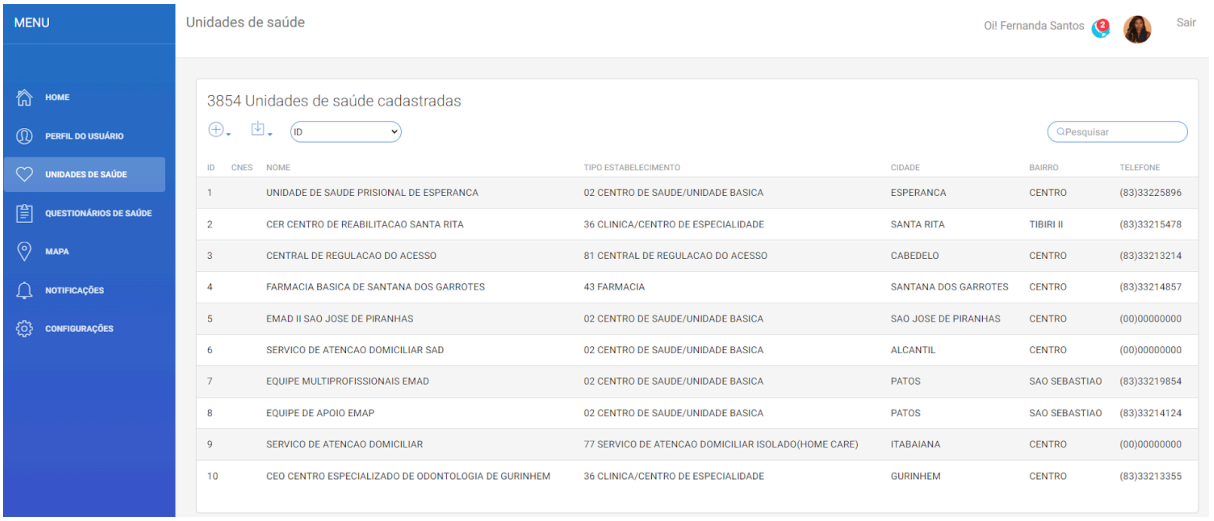
Outra forma de realizar a visualização desses dados se deu pelo uso do *front-end*, Após um processo de conexão e validação da comunicação entre os dois sistemas, também foi necessário a inserção de chaves de acesso a contas *Google.* Para utilização do serviço geográfico do *Google Cloud*. Após isso chegamos nas seguintes demonstrações de resultados de dados acessados no banco de dados do *back-end* e através de requisições do *front-end* é interpretada e mostrada no sistema.

Figura 45 - Requisição do navegador para acesso aos dados.



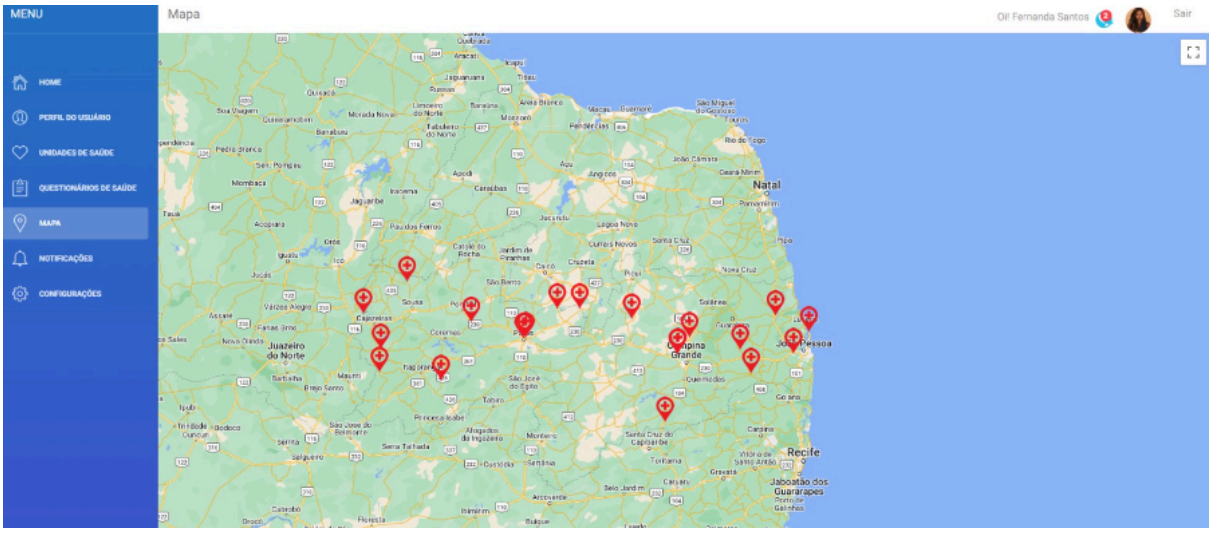
Fonte: Elaborado pelo autor, (2024).

Figura 46 - Página de visualização de unidades de saúde no navegador.



Fonte: Elaborado pelo autor, (2024).

Figura 47 - Visualização geográfica dos dados utilizando ferramentas do Google cloud.



Fonte: Elaborado pelo autor, (2024).

7 CONCLUSÃO

O presente trabalho teve como objetivo o desenvolvimento de um *back-end* robusto para integrar-se a um *front-end* já existente, com o propósito de melhorar o direcionamento de usuários a unidades de saúde próximas e adequadas. Desde o início, o projeto se concentrou em abordar um dos maiores desafios enfrentados pelo sistema de saúde pública brasileira: a superlotação dos hospitais centrais e a dificuldade de acesso a unidades de saúde adequadas. Através do uso de tecnologias modernas e boas práticas de desenvolvimento, buscou-se uma solução que pudesse oferecer aos usuários um sistema eficaz e fácil de usar, contribuindo para a mitigação dos problemas de acesso à saúde.

A primeira etapa do desenvolvimento consistiu em uma análise detalhada do cenário atual, com a coleta de informações a partir de dados existentes sobre a superlotação hospitalar e as dificuldades enfrentadas pelos pacientes na busca por atendimento. A partir desse levantamento, foram estabelecidas as principais demandas e requisitos para o sistema, tanto no que diz respeito às funcionalidades necessárias quanto aos requisitos técnicos e de segurança. Diagramas e especificações detalhadas foram criados para guiar o processo de desenvolvimento, garantindo que todos os aspectos críticos do sistema fossem abordados de maneira eficaz.

Em seguida, iniciou-se o processo de construção de uma base de dados concisa e bem estruturada, que servirá como alicerce para o sistema. Esse processo envolveu a correção de inconsistências nos dados, muitas das quais foram identificadas durante a fase de análise. O objetivo principal foi garantir que o sistema tivesse acesso a informações precisas e atualizadas sobre as unidades de saúde, incluindo localização, tipos de atendimento oferecidos, horários de funcionamento e outros dados relevantes. A precisão desses dados é fundamental para garantir que os usuários sejam direcionados para a unidade de saúde mais adequada às suas necessidades.

A escolha das tecnologias a serem utilizadas no desenvolvimento do *back-end* também foi uma etapa crucial. Optou-se pela utilização do *Spring Boot*, uma tecnologia amplamente reconhecida por sua robustez, flexibilidade e

capacidade de suportar sistemas de grande escala. O uso de *APIs*, foi adotado para garantir que o *back-end* pudesse se comunicar de maneira eficiente e em tempo real com o *front-end*, oferecendo aos usuários uma experiência fluida e responsiva. Além disso, foram implementadas camadas de segurança robustas, garantindo que os dados dos usuários fossem protegidos contra ameaças externas.

Durante o desenvolvimento, diversas adversidades técnicas foram encontradas. Desde a correção de falhas nos dados até a superação de desafios relacionados à escalabilidade e segurança, cada etapa do processo exigiu uma abordagem cuidadosa e baseada em boas práticas de desenvolvimento. A documentação criada ao longo do projeto foi fundamental para manter o fluxo de trabalho organizado e para garantir que todas as decisões tomadas estivessem alinhadas com os objetivos do projeto. O desenvolvimento seguiu um ciclo de iterações, permitindo a correção de problemas e o ajuste fino das funcionalidades conforme o sistema avançava.

Um dos principais resultados deste trabalho foi a construção de uma *API* funcional e alinhada com os requisitos do *front-end*. Além das funcionalidades básicas de comunicação entre o *back-end* e o *front-end*, a *API* inclui funcionalidades adicionais, como a inserção de dados de forma segura e eficiente, e a criação de uma interface baseada em *Swagger*, que permite uma visualização clara e direta das requisições feitas pelo sistema. Essa interface facilita tanto o desenvolvimento contínuo quanto o teste e validação das funcionalidades implementadas

Após a finalização do *back-end*, foi necessário dedicar esforços significativos para garantir a correta integração com o *front-end* existente. Ajustes e melhorias no código do *front-end* foram realizados para assegurar que a comunicação entre as duas partes do sistema fosse fluida e sem falhas. Esse processo de integração foi bem-sucedido, resultando em um sistema coeso e funcional capaz de lidar com as necessidades dos usuários e de fornecer informações precisas e em tempo real sobre as unidades de saúde disponíveis.

A implementação deste sistema, embora já funcional, ainda apresenta algumas lacunas que podem ser abordadas em trabalhos futuros. Um dos principais pontos a serem considerados é a necessidade de garantir a alta disponibilidade do sistema, especialmente em cenários de uso intenso e em áreas onde o acesso à

saúde é crítico. Além disso, a segurança dos dados é uma questão contínua que requer atenção constante. Em um mundo cada vez mais digital, proteger os dados dos usuários contra ameaças externas é uma prioridade. A implementação de soluções mais avançadas de segurança cibernética e a adoção de ferramentas de monitoramento são passos essenciais para garantir a integridade do sistema.

7.1 Trabalhos Futuros

Entre os principais trabalhos futuros, destaca-se a continuidade do desenvolvimento do *front-end*, com foco na implementação de uma funcionalidade que permita a aplicação de questionários personalizados para os usuários. Esses questionários poderão avaliar as condições de saúde dos pacientes e, com base nas respostas, direcioná-los para a unidade de saúde mais adequada às suas necessidades. Essa funcionalidade tem o potencial de melhorar significativamente o processo de triagem e direcionamento dos pacientes, ajudando a reduzir a superlotação em unidades menos preparadas para certos tipos de atendimento.

Além disso, melhorias contínuas no código, são fundamentais para otimizar o desempenho do sistema. A refatoração do código, a adoção de novas tecnologias e o ajuste de processos podem contribuir para tornar o sistema ainda mais eficiente e escalável. Outro ponto a ser explorado em trabalhos futuros é a implementação de práticas de *DevOps*. Ferramentas de *DevOps* permitirão o monitoramento contínuo do sistema, facilitando a identificação e correção de falhas, além de garantir que o sistema funcione de maneira eficiente mesmo sob condições adversas.

Por fim, a implantação do sistema em um ambiente de produção, com testes reais e monitoramento constante, será um passo essencial para validar sua eficácia em situações reais. A expectativa é que, com as melhorias sugeridas e a aplicação de técnicas avançadas de monitoramento e segurança, o sistema possa beneficiar de maneira significativa a população brasileira, ajudando a reduzir os problemas de superlotação e garantindo que os pacientes sejam direcionados para os locais de atendimento mais adequados às suas necessidades.

REFERÊNCIAS

BRASIL. **Portal do Ministério da Saúde**. 2024. Disponível em: <https://www.gov.br/saude/pt-br.> Acesso em: 20 ago. 2024.

GAMA, Alexandre. **O que é JSON**. DevMedia, 2013. Disponível em: <https://www.devmedia.com.br/o-que-e-json/23166.> Acesso em: 20 ago. 2024.

MDN Web Docs. **HTTP Overview**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview.> Acesso em: 20 ago. 2024.

MEDEIROS, Higor. **Application Programming Interface: desenvolvendo APIs de software**. DevMedia, 2024. Disponível em: <https://www.devmedia.com.br/application-programming-interface-desenvolvendo-apis-de-software/30548.> Acesso em: 20 ago. 2024.

PYTHON Software Foundation. **Python.org**. Disponível em: <https://www.python.org/.> Acesso em: 20 ago. 2024.

REZENDE, Ricardo. **Conceitos fundamentais de banco de dados**. DevMedia, 2006. Disponível em: <https://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649.> Acesso em: 20 ago. 2024.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

STACK OVERFLOW. **Stack Overflow Developer Survey 2023**. Disponível em: <https://survey.stackoverflow.co/2023/#most-popular-technologies-language.> Acesso em: 20 ago. 2024.

UZELLI, G. D. P. et al. **Avaliação das dificuldades enfrentadas pelo paciente para realização de uma consulta médica de nível terciário**. *Com. Ciência Saúde*, v. 23, n. 3, p. 207–214, 2013.

VENTURA, Daniele Batista. **Desenvolvimento front-end de aplicativo para busca de serviços de saúde**. [UEPB], 2022. Disponível em:<http://dspace.bc.uepb.edu.br/jspui/handle/123456789/27921>. Acesso em: 20 ago. 2024.

WEISSMANN, Henrique Lobo. **Spring Boot: simplificando o Spring**. DevMedia, 2015. Disponível em: <https://www.devmedia.com.br/spring-boot-simplificando-o-spring/31979>. Acesso em: 20 ago. 2024.