

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Arquitetura de Software Distribuído**

**André Fernando Araújo de Carvalho**

**OPERADORA DE PLANO DE SAÚDE**

Belo Horizonte

2022

**André Fernando Araújo de Carvalho**

**OPERADORA DE PLANO DE SAÚDE**

Trabalho de Conclusão de Curso de  
Especialização em Arquitetura de Software  
Distribuído como requisito parcial à obtenção  
do título de especialista.

Orientador(a): Pedro Alves de Oliveira

Belo Horizonte

2022

*Dedico este trabalho à minha família que sempre me apoiou e incentivou a continuar principalmente nos momentos de estudo em detrimento a horas de lazer.*



## RESUMO

Este trabalho visa apresentar uma solução arquitetural que permita integrar de forma transparente, as ferramentas de software atualmente em uso pela empresa, operadora de planos de saúde e introduzir novas ferramentas. A análise parte do documento de escopo, no qual estão descritas as ferramentas existentes e seu contexto de funcionamento além das necessidades de evolução elencadas como prioridade no processo de adoção de novas tecnologias. Inicialmente, foi delimitado o problema com foco na necessidade de melhoria da integração das diversas soluções existentes, tratadas como sistemas legados. Os requisitos funcionais e não funcionais e as restrições e mecanismos arquiteturais foram identificados, categorizados, descritos, avaliados quanto à prioridade e dificuldade e receberam um código de identificação único. A seguir, a proposta de arquitetura geral é apresentada por meio do diagrama de visão geral, permitindo o entendimento do contexto e de elementos arquiteturais macro. A partir desta análise inicial, definiu-se quatro casos de uso, utilizados para desenvolver a prova de conceito. O detalhamento da visão lógica que inclui diagramas UML com a especificação de classes, componentes, cenário de implantação e modelo de entidades, detalhados com profundidade suficiente para permitir a implementação da POC. Com estes insumos, pode-se implementar as funcionalidades que representam os casos de uso anteriormente definidos e por fim, avaliar os resultados.

**Palavras-chave:** arquitetura de software, projeto de software, requisito arquitetural, serviços Web, API, padronização, camadas, divisão de responsabilidades, componentização, integração, localização geográfica, OpenSource, tratamento de dados, transparência, responsividade, infraestrutura como código, operação, manutenção.

## SUMÁRIO

1. Apresentação.....	7
1.1 Problema.....	7
1.2 Objetivo do trabalho.....	7
1,3 Definições e Abreviaturas.....	7
<b>2. Especificação da Solução.....</b>	<b>7</b>
2.1 Requisitos Funcionais.....	7
2.2 Requisitos Não Funcionais.....	8
2.3 Restrições Arquiteturais.....	8
2.4 Mecanismos Arquiteturais.....	8
<b>3. Modelagem Arquitetural.....</b>	<b>9</b>
3.1 Macroarquitetura.....	9
3.2 Descrição Resumida dos Casos.....	9
3.3 Visão Lógica.....	10
<b>4. Prova de Conceito (POC) e Protótipo Arquitetural.....</b>	<b>12</b>
4.1. Implementação.....	12
4.2 Interfaces e APIs.....	13
<b>5. Avaliação da Arquitetura.....</b>	<b>13</b>
5.1. Análise das abordagens arquiteturais.....	14
5.2. Cenários.....	14
5.3. Evidências da Avaliação.....	15
5.4. Resultados.....	16
<b>6. Conclusão.....</b>	<b>16</b>
<b>REFERÊNCIAS.....</b>	<b>17</b>
<b>APÊNDICES.....</b>	<b>18</b>

## **1. Apresentação**

No atual contexto de grande concorrência entre as empresas, é cada vez mais urgente a necessidade de aprimorar os processos gerenciais e administrativos, permitindo a redução de custos, tanto com otimização de uso dos recursos, quanto na identificação de desperdícios e redução de retrabalho. Outro aspecto importante é a integração da cadeia de insumos e prestação de serviços, elemento primordial nos processos operacionais.

Este cenário não é diferente para as operadoras de planos de saúde, assim, faz-se necessária a evolução tecnológica das ferramentas de software já utilizadas, com maior integração e possibilidade de adoção de novas ferramentas, que permitam ampliar a capacidade de armazenamento e processamento de dados, a uniformização no acesso às informações, de forma ágil, transparente e segura.

### **1.1 Problema**

A Boa Saúde Assistência Médica opera atualmente com diversos sistemas legados, dentre os quais se destacam: SAF, SGPS e SAS. Estas soluções possuem pouco ou nenhuma integração, gerando impacto significativo no tempo de execução dos processos operacionais, administrativos e gerenciais além de ineficiência no uso das informações de colaboradores, prestadores, conveniados e associados. Ainda é possível que ocorram atrasos no tempo de atendimento aos associados, falhas no processo de faturamento, desperdício de recursos por falhas na identificação previa da capacidade de atendimento dos prestadores e conveniados, entre outros.

### **1.2 Objetivo do trabalho**

O objetivo deste trabalho é propor uma arquitetura de integração de sistemas que seja capaz de suportar as necessidades evolutivas da empresa Boa Saúde Assistência Médica, alinhada ao plano de metas e diretrizes na área de tecnologia, incluindo os seguintes objetivos específicos:

- Integrar os processos de atendimento que envolvam os prestadores, conveniados e associados;

- Utilizar geotecnologias em todos os processos que envolvam localização;
- Guiar o processo de integração de novas ferramentas de software para uso gerencial e estratégico;
- Permitir o desenvolvimento e a implantação modular do sistema;
- Desenvolver prova de conceito que permita corroborar parte da solução proposta;
- Servir como ponto de partida para as tarefas de refinamento de especificação, desenvolvimento, codificação e implantação;

### 1.3 Definições e Abreviaturas

OGC: Open Geospatial Consortium, consórcio internacional para definição de padrões de consenso para implementação de softwares que operam com dados geográficos (<https://www.osgeo.org/partners/ogc/>).

WMS: Web Map Service, um padrão OGC para implementação de serviço de mapas em formatos gráficos como png e jpeg, por exemplo.

WFS: Web Feature Service, um padrão OGC para implementação de serviço de mapas em formatos vetoriais como GeoJson, shapefile e GML entre outros.

TMS: Tile Map Service, um serviço padrão OGC para implementação RESTfull de serviço de mapas em formatos gráficos como png e jpeg, por exemplo, com definição de tamanho fixo de partes.

Geocodificação (Geocode): Processo de transformação de um endereço em coordenada geográfica.

## 2. Especificação da Solução

Nesta sessão são apresentadas a lista de requisitos funcionais e não funcionais, as restrições arquiteturais impostas ao projeto e os mecanismos arquiteturais adotados.

A identificação dos requisitos e das restrições arquiteturais será por meio de código formado por prefixo seguido de numeração, específicos de cada subseção, a saber:

- Requisitos Funcionais (RF + número);



- Requisitos Não Funcionais (RNF + número);
- Restrições Arquiteturais (RA + número);

Esta codificação será utilizada em todas as demais sessões, quando for necessário associar quaisquer elementos arquiteturais aos itens aqui identificados.

## 2.1 Requisitos Funcionais

Nesta subseção são apresentados os requisitos funcionais, segmentados por módulos, extraídos do documento de especificação. Cada requisito está identificado por código, de forma unívoca.

Foi atribuída classificação de dificuldade e prioridade a cada um dos requisitos.

A classificação de dificuldade serve como balizador no dimensionamento da equipe e no tempo de execução, e a classificação de prioridade indica a ordem de atendimento. Ambos são itens importantes na definição do cronograma de desenvolvimento.

### Módulo de Informações Cadastrais.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve prover funcionalidade de identificação de usuários por apresentação de credenciais de acesso previamente registradas.	B	A
RF02	O sistema deve solicitar e armazenar credenciais de acesso durante o processo de cadastro de usuários, e prover funcionalidade de recuperação e alteração de senhas.	B	A
RF03	O sistema deve permitir, após verificação de validade, a autorização de usuários por meio da apresentação de <i>token</i> de acesso obtido por processo de identificação.	B	A
RF04	O sistema deve permitir a atribuição de permissões específicas de acesso às suas funcionalidades, por meio de criação e alteração de perfis de usuários.	M	A
RF05	O sistema deve atender, de forma seletiva (por perfil) aos associados, cooperados e conveniados.	M	A
RF06	O sistema deve prover acesso especial às funcionalidades administrativas e financeiras aos colaboradores da Boa Saúde que possuam estas atribuições.	M	A

RF07	O sistema deve permitir a manutenção do cadastro dos prestadores e conveniados da operadora, por meio de funções de criação, modificação e exclusão. Deve incluir dados de tipos de atendimentos que podem ser realizados, local de atendimento (geolocalização), recursos disponíveis, capacidade de vazão de atendimentos, entre outros dados relevantes que permitam a correta atribuição e tramitação do atendimento.	A	A
RF08	O sistema deve permitir o lançamento de atendimentos realizados pelos prestadores e conveniados da operadora aos seus associados, em tempo real, incluindo as fases de apuração, conferência e faturamento, considerando o estado de agendado ou realizado, conforme os dados constantes no cadastro dos prestadores e conveniados.	A	A
RF09	O sistema deve informar a localização em mapa dos prestadores e conveniados disponíveis para a realização de atendimentos no município do associado.	A	A
RF10	O sistema deve informar a localização em mapa dos prestadores e conveniados disponíveis para a realização de atendimentos, por meio de aplicação de filtros avançados: especialidades, capacidade de atendimento (taxa de ocupação), proximidade, menor valor de faturamento.	A	M
RF11	O sistema deve permitir o acompanhamento, em tempo real, dos deslocamentos e localização de pacientes em atendimento na rede conveniada.	A	B
RF12	O sistema deve permitir o registro de <i>check-in</i> e <i>check-out</i> dos associados em processo de atendimento, incluindo a correta identificação e validação de todos os atendimentos realizados pelos prestadores e conveniados. Atendimentos de emergência permitem registro tardio de entrada/saída.	A	M
RF13	O sistema deve permitir a associação de coordenadas geográficas dado um endereço cadastral, nos processos de manutenção do cadastro dos usuários, por meio de integração com serviço de geocodificação de endereços. Deve prever a utilização em lotes para endereços existentes na base atual.	M	A

\*B=Baixa, M=Média, A=Alta.

### Módulo de Serviços ao Associado.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF14	O sistema deve garantir acesso às ferramentas de BPM apenas aos usuários do perfil analista de processos	B	M
RF15	O sistema deve permitir a manutenção dos processos de negócio como: desenho de novos processos e alteração de	M	M

	processos existentes.		
RF16	O sistema deve permitir a consulta aos processos existentes para acompanhamento e análise.	B	M
RF17	O sistema deve garantir o versionamento dos processos criados com finalidade de entendimento da evolução de um processo no tempo.	A	M
RF18	O sistema deve manter a informação do usuário responsável pela criação e pelas alterações realizadas nos processos registrados.	B	M
RF19	O sistema deve permitir a exportação de um processo para algum formato padrão de mercado, como jpeg, png ou pdf, para garantir o compartilhamento e utilização em ambiente externo ao sistema.	M	B

\*B=Baixa, M=Média, A=Alta.

### Módulo de Gestão e Estratégia.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF20	O sistema deve garantir acesso às ferramentas de gestão apenas aos usuários do perfil analista de gestão.	B	A
RF21	A ferramenta deve permitir a coleta de dados sobre projetos, produtos e serviços da empresa.	A	M
RF22	A ferramenta deve ser capaz de gerar indicadores com base nos dados coletados.	M	A
RF23	A ferramenta deve ser capaz de apresentar os indicadores gerados de forma individual e associativa, quando possível.	M	M
RF24	A apresentação dos indicadores deve ser em forma de gráficos, com possibilidade de aplicação de filtros e agregações ou desagregações nas dimensões temporal e espacial.	A	M

\*B=Baixa, M=Média, A=Alta.

### Módulo de Ciência de Dados.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF25	O sistema deve garantir acesso às ferramentas de BI apenas aos usuários do perfil analista de dados	M	A
RF26	O sistema deve prover mecanismos padronizados para obtenção, tratamento e carga de dados provenientes de fontes diversas como: planilhas, bancos de dados relacionais e não relacionais e fontes externas não padronizadas.	A	M

RF27	O sistema deve prover estruturas para o armazenamento, catalogação e recuperação de grandes massas de dados em modelo data warehouse.	A	M
RF28	O sistema deve suportar ferramentas para análise de grandes massas de dados com capacidade de geração de cenários segmentados por setor e cenários globais.	A	M
RF29	O sistema deve suportar ferramentas para geração de relatórios a partir de modelos e cenários pré definidos.	M	M
RF30	O sistema deve prover ferramentas para geração de painéis gráficos para facilitar a apresentação, compartilhamento e entendimento de resultados de análises para cenários pré definidos.	A	M

\*B=Baixa, M=Média, A=Alta.

## 2.2 Requisitos Não Funcionais

Nesta subseção são apresentados todos os requisitos não funcionais, extraídos do documento de especificação. Cada requisito está identificado por código, de forma unívoca.

Foi atribuída classificação de prioridade a cada um dos requisitos, permitindo que a equipe concentre maior ou menor atenção ao cumprimento de cada um deles, além de servir como peso em decisões onde houver conflito entre os requisitos ou ainda o relaxamento na observância de determinado requisito com propósito de redução de custos.

ID	Descrição	Prioridade (B/M/A)*
RNF01	O sistema deve ser acessível nas plataformas web e móvel (multiplataforma)	A
RNF02	Ser testável em todas as suas funcionalidades (qualidade)	A
RNF03	As interfaces devem ser simples e direta, com disposição consistente de informações e dicas de preenchimento em todos os elementos. Todas as entradas devem ser validadas pela interface. O tempo de interação de um usuário com uma tela deve ser, em média, 5 minutos (usabilidade)	A
RNF04	O tempo de reparo de um componente deve ser, em média, de 48 horas (manutenibilidade)	M

RNF05	O tempo de implantação de um componente ou conjuntos de componentes, deve ser, em média, de 2 minutos (fácil implantação).	M
RNF06	O usuário não deve perceber que está utilizando aplicações e tecnologias diferentes (transparência).	M
RNF07	O tempo máximo de resposta do sistema aos usuários deve ser de 5 segundos (bom desempenho).	A
RNF08	Ser recuperável (resiliente) no caso da ocorrência de erro (robustez)	A
RNF09	Estar disponível em horário integral, 24x7 (alta disponibilidade)	A
RNF10	As interfaces devem se adaptar a tamanhos diferentes de tela, exibindo menor ou maior quantidade de elementos quando necessário. (acessibilidade).	A
RNF11	O sistema deve prover respostas aos recursos de APIs em formatos de fácil integração com outros sistemas (interoperabilidade).	A

\*B=Baixa, M=Média, A=Alta.

## 2.3 Restrições Arquiteturais

- **RA01:** Baixo custo de software, preferencialmente sem custo de licença;
- **RA02:** Deve-se utilizar APIs para integração;
- **RA03:** Utilizar arquitetura baseada em serviços, total ou parcialmente;
- **RA04:** Possuir características de aplicação distribuída: abertura, portabilidade, uso de recursos de rede;
- **RA05:** Os componentes de software devem ser encapsulados em *containers* Docker, permitindo implantação padronizada e automatização de implantação via pilha de serviços (IaaS).
- **RA06:** Ser hospedado em nuvem híbrida;

## 2.4 Mecanismos Arquiteturais

Análise	Design	Implementação
Persistência	ORM	Sequelize/Postgres+PostGIS
Front end	Single Page Application	JavaScript+HTML+CSS (jquery+leaflet)

Back end	APIs + GeoService (OGC pattern)	Node.js+Express+axios GeoServer
Integração	Ajax/Promise	jquery/axios
Exceções	try/catch	Manual
Log do sistema	Saída padrão para arquivo	Manual
Teste de endpoints	HTTP requests (GET, POST, PUT)	Postman
Teste de interfaces	Inspect/Debug/Measure	DevTools (navegador)
Deploy	Container	Docker compose (yaml)
Diagramação	UML/C4	Draw.io

### 3. Modelagem Arquitetural

Nesta sessão são apresentados os artefatos da modelagem arquitetural da solução proposta, permitindo identificar de forma adequada os elementos de software necessários à implementação da prova de conceito. A abordagem inicial apresenta uma visão macro da arquitetura, de forma a contextualizar o leitor, em seguida, ajusta o foco nos detalhes dos elementos arquiteturais selecionados para a implementação, definindo e elaborando os casos de uso e os diagramas da visão lógica.

### 3.1 Macroarquitetura

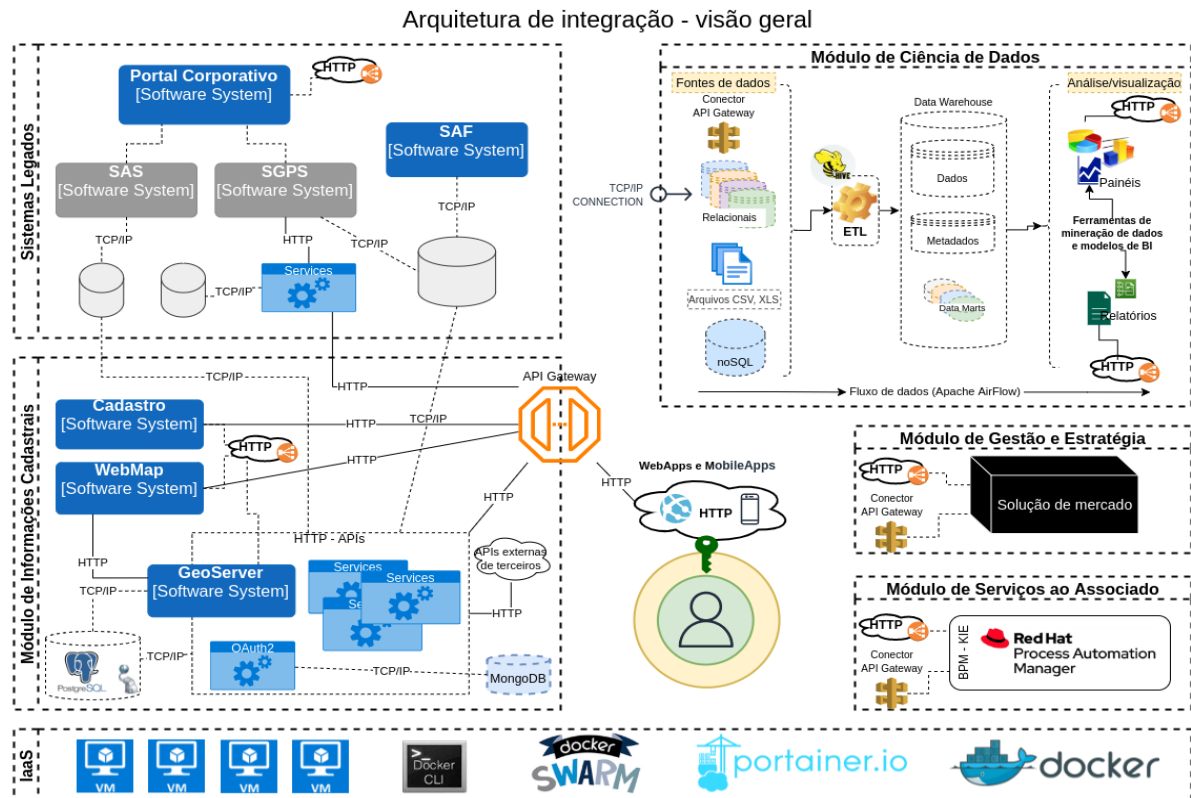


Figura 1 – Visão Geral da Solução.

A figura 1 mostra a especificação do diagrama geral da solução proposta, com todos os módulos, suas interfaces e protocolos de comunicação e elementos que remetem ao ambiente de operação.

Além dos quatro módulos, previamente definidos no documento de escopo, estão representados os componentes dos sistemas legados e os usuários como elementos centrais da solução, denominada Gestão Integral da Saúde do Associado (GISA).

**Sistemas legados** é a representação gráfica das peças de software existentes na arquitetura atual, e procura contextualizar de forma integral, mas simplificada, a integração inicial com os demais módulos da solução. É importante ter em mente a necessidade de se manter a operação da empresa em pleno funcionamento, e portanto, não se propõe uma grande mudança. A abordagem proposta é utilizar a técnica do estrangulamento de monólito, com pequenas e sucessivas implementações a fim de substituir parte das funcionalidades existentes por camadas de serviços e interfa-

ces web responsivas. Esta mesma abordagem serve ao propósito de evolução para implementação de novas funcionalidades. Como descrito no documento de escopo, o conjunto de componentes de software existente é bem heterogêneo, e portanto é indispensável análises exploratórias para se adotar medidas efetivas e de baixo impacto. Pode-se perceber que o componente mais sensível e fechado é o Sistema Administrativo-financeiro (SAF), mas por ter sido implementado em JEE, pode ser possível identificar de forma clara os componentes da camada de negócio e escrever uma camada de serviços logo acima desta, permitindo a reutilização de grande parte do código e introduzindo a tão desejada facilidade de integração.

**Módulo de informações cadastrais** possui grande importância na arquitetura da solução pois é o responsável por obter e manter a massa principal de dados relativos aos usuários. Como dados são o insumo essencial para qualquer sistema de informações, além de obter e manter, disponibilizar de forma adequada e segura para os demais módulos do sistema é tarefa fundamental deste módulo. Para assegurar este propósito, faz-se necessário adotar as melhores práticas no desenvolvimento, manutenção e evolução do módulo, utilizando padrões de comunicação bem definidos, camada de controle de acesso adequada, tecnologias sólidas e estáveis no mercado. O coração do módulo deverá utilizar um gerenciador de banco de dados relacional, estável e escalável, de grande capacidade além de possuir ferramentas de administração de fácil utilização e fornecer suporte a tipos de dados espaciais. A opção indicada é o SGBD PostgreSQL, que além das características elencadas anteriormente, possui versão gratuita e conta com comunidade muito e possibilidade de suporte técnico especializado de nível empresarial. Outra característica essencial deste módulo é a forte presença de camada de serviços baseados em APIs Web que devem suportar todas as operações definidas no módulo.

**Módulo de Serviços ao Associado** pode incrementar de maneira importante o gerenciamento dos processos da empresa pois permite a definição, acompanhamento e evolução dos processos de negócio de forma gráfica e compartilhada. Uma possibilidade que vai de encontro às restrições estabelecidas pela área de TI da empresa, é a adoção do produto *Process Automation Manager* da Red Hat, que inclui muitas



das ferramentas indicadas como necessárias no módulo e traz vantagens como a possibilidade de automação de grande parte dos processos e implantável em plataforma computacional de nuvem híbrida.

**Módulo de Gestão e Estratégia**, sendo composto principalmente por uma ferramenta de gestão corporativa adquirida no mercado, e portanto representado como uma caixa-preta e incluindo algum elemento de integração com capacidade de obter dados de entrada por meio de consumo de APIs padronizadas, arquivos e até entradas manuais.

**Módulo de Ciência de Dados** representa a evolução tecnológica da empresa na direção de identificar cenários de inovação, identificando elementos ocultos nas bases de dados na medida que permite entregar capacidade analítica à empresa. Para o sucesso na implementação deste módulo é imprescindível a composição de equipe com profissionais como analistas de negócio e cientistas de dados para direcionar os esforços da equipe de implementação em direção à escolha das melhores ferramentas e correta captura das especificidades do negócio. Aqui, as recomendações são para o uso de ferramentas já comprovadamente eficientes, principalmente em organização do armazenamento de dados com enfoque na etapa posterior, a do acesso, permitindo a execução das rotinas de análise em tempo razoável. Assim, ferramentas do ecossistema Hadoop devem estar no centro da avaliação, bem como ferramentas como o Apache AirFlow que devem facilitar a implementação de rotinas automatizadas em um ambiente padronizado e que permite bom acompanhamento da operação.

### 3.2 Descrição Resumida dos Casos de Uso

Nesta sessão são apresentados os casos de uso escolhidos para a implementação da POC. A figura 2 representa o diagrama de casos de uso para facilitar a contextualização e na sequência a descrição resumida de cada um deles.

Diagrama de casos de uso

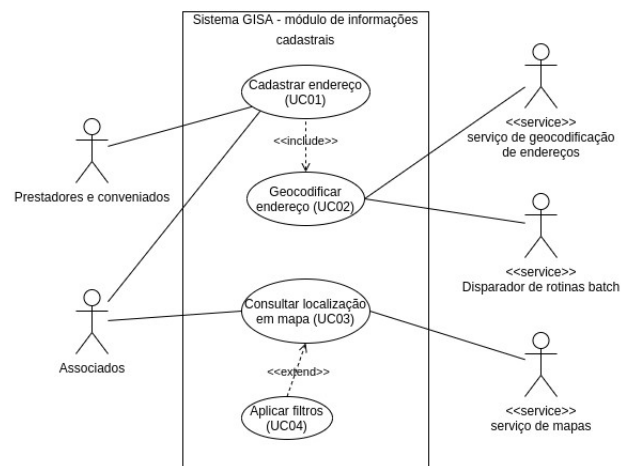


Figura 2 – Casos de uso selecionados para a POC.

UC01 – CADASTRAR ENDEREÇO	
<b>Descrição</b>	Cadastro de endereço incluindo localização geográfica
<b>Atores</b>	Prestadores, conveniados e associados
<b>Prioridade</b>	A
<b>Requisitos associados</b>	RF07
<b>Fluxo Principal</b>	<p>Esta funcionalidade permite ao usuário cadastrar seu endereço, por meio dos seguintes passos:</p> <ol style="list-style-type: none"> <li>1) O usuário acessa o endereço HTTP da aplicação de cadastro;</li> <li>2) O sistema exibe a tela para entrada de endereço;</li> <li>3) O usuário digita o endereço, preenchendo os campos obrigatórios e envia os dados;</li> <li>4) O sistema valida o preenchimento e desvia o fluxo para UC02, geocodificação de endereços;</li> <li>5) O sistema recebe a resposta de UC02 e exibe a localização no mapa;</li> <li>6) O usuário avalia se a localização encontrada corresponde ao endereço fornecido, corrige a posição no mapa, caso necessário, e confirma;</li> <li>7) O sistema armazena o endereço relacionado ao usuário;</li> <li>8) O caso de uso é encerrado;</li> </ol>

#### UC02 – GEOCODIFICAR ENDEREÇO

<b>Descrição</b>	Gerar localização geográfica para um dado endereço
<b>Atores</b>	Prestadores, conveniados e associados
<b>Prioridade</b>	A
<b>Requisitos associados</b>	RF13
<b>Pré-requisito</b>	Depende de UC01 ou uma rotina de geolocalização em batch
<b>Fluxo Principal</b>	<p>Esta funcionalidade permite obter a coordenada geográfica com base em um endereço, por meio dos seguintes passos:</p> <ol style="list-style-type: none"> <li>1) O sistema recebe uma requisição com as partes que compõe um endereço;</li> <li>2) O sistema valida as partes e requisita ao serviço de geocodificação de endereços, a localização geográfica para o endereço fornecido;</li> <li>3) O sistema devolve a localização geográfica encontrada ou um código de falha;</li> <li>7) O caso de uso é encerrado;</li> </ol>

<b>UC03 – CONSULTAR LOCALIZAÇÃO EM MAPA</b>	
<b>Descrição</b>	Consulta de prestadores e conveniados mais próximos
<b>Atores</b>	Associados
<b>Prioridade</b>	A
<b>Requisitos associados</b>	RF09
<b>Fluxo Principal</b>	<p>O usuário poderá consultar mapa de prestadores e conveniados localizados no mesmo município de sua localização geográfica, por meio dos seguintes passos:</p> <ol style="list-style-type: none"> <li>1) O usuário acessa o endereço HTTP do painel de consulta em mapa;</li> <li>2) O sistema exibe ao usuário possibilidade de acessar a consulta básica ou avançada;</li> <li>3) O usuário seleciona a consulta básica;</li> <li>4) O sistema identifica o município onde se encontra a coordenada geográfica relacionada ao endereço do usuário e oferece possibilidade de identificar prestadores e conveniados dentro da área do município;</li> <li>5) O usuário confirma a solicitação de exibição do mapa;</li> <li>6) O sistema exibe o mapa do município onde o usuário está localizado, incluindo todos os prestadores e conveniados localizados no município;</li> <li>7) O caso de uso é encerrado;</li> </ol>

UC04 – APLICAR FILTROS AO MAPA	
<b>Descrição</b>	Definir filtros ao consultar prestadores e conveniados
<b>Atores</b>	Associados
<b>Prioridade</b>	A
<b>Requisitos associados</b>	RF10
<b>Fluxo Principal</b>	<p>O usuário poderá definir e aplicar filtros ao consultar mapa de prestadores e conveniados, por meio dos seguintes passos:</p> <ol style="list-style-type: none"> <li>1) O usuário acessa o endereço HTTP do painel de consulta em mapa;</li> <li>2) O sistema exibe ao usuário possibilidade de acessar a consulta básica ou avançada;</li> <li>3) O usuário seleciona a consulta avançada;</li> <li>4) O sistema exibe os controles de filtro;</li> <li>5) O usuário seleciona as opções disponíveis e aplica os filtros (especialidades, proximidade (raio em km));</li> <li>6) O sistema exibe as coordenadas geográficas dos prestadores e conveniados que satisfazem os filtros do associado e o círculo de abrangência utilizado;</li> <li>7) O caso de uso é encerrado;</li> </ol>

### 3.3 Visão Lógica

Esta seção mostra a especificação dos diagramas da solução proposta, com todos os seus componentes, propriedades e interfaces. Nas subseções a seguir são apresentados os diagramas de Classes, Componentes e Implantação. Também é apresentado um diagrama Entidade-Relacionamento para representar o modelo de dados.

#### 3.3.1 Diagrama de Classes

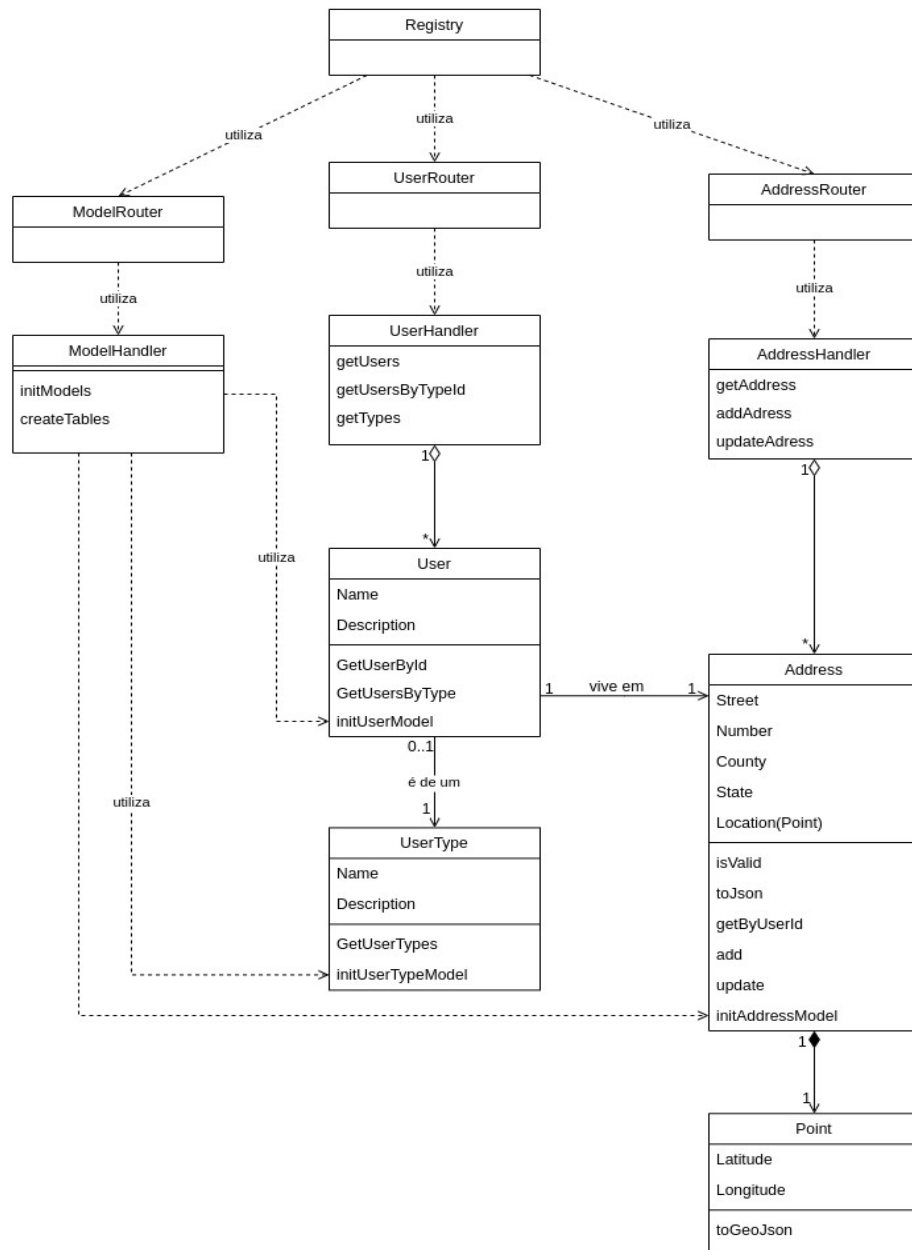


Figura 3 – Diagrama de classes (API de Registro de endereço).

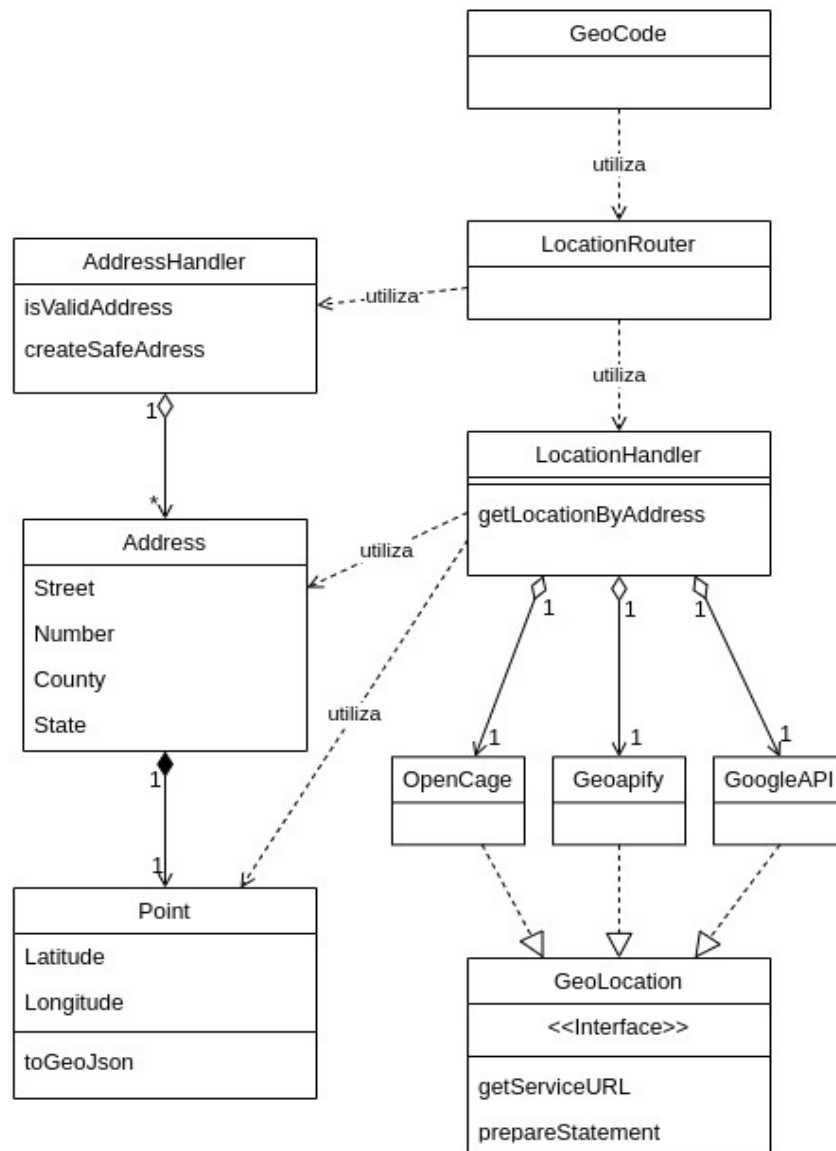


Figura 4 – Diagrama de classes (API de Geocodificação de endereço).

As figuras 3 e 4 representam as classes identificadas como parte da solução que deve ser construída para compor a prova de conceito aqui proposta.

As duas figuras definem dois serviços Web representados por APIs Rest.

O primeiro serviço, denominado API de Registro de endereço (Registry) dará suporte às operações de consulta e armazenamento de endereços textuais, relacionados a cada usuário do sistema.

O segundo serviço, denominado API de Geocodificação de endereço (Geocode), fará a conversão do endereço textual para uma coordenada geográfica, consultando um serviço de terceiro.

### 3.3.2 Diagrama de Componentes

Nesta sessão é apresentado o diagrama de componentes necessários para a implementação da solução que permita demonstrar os casos de uso escolhidos para a prova de conceito.

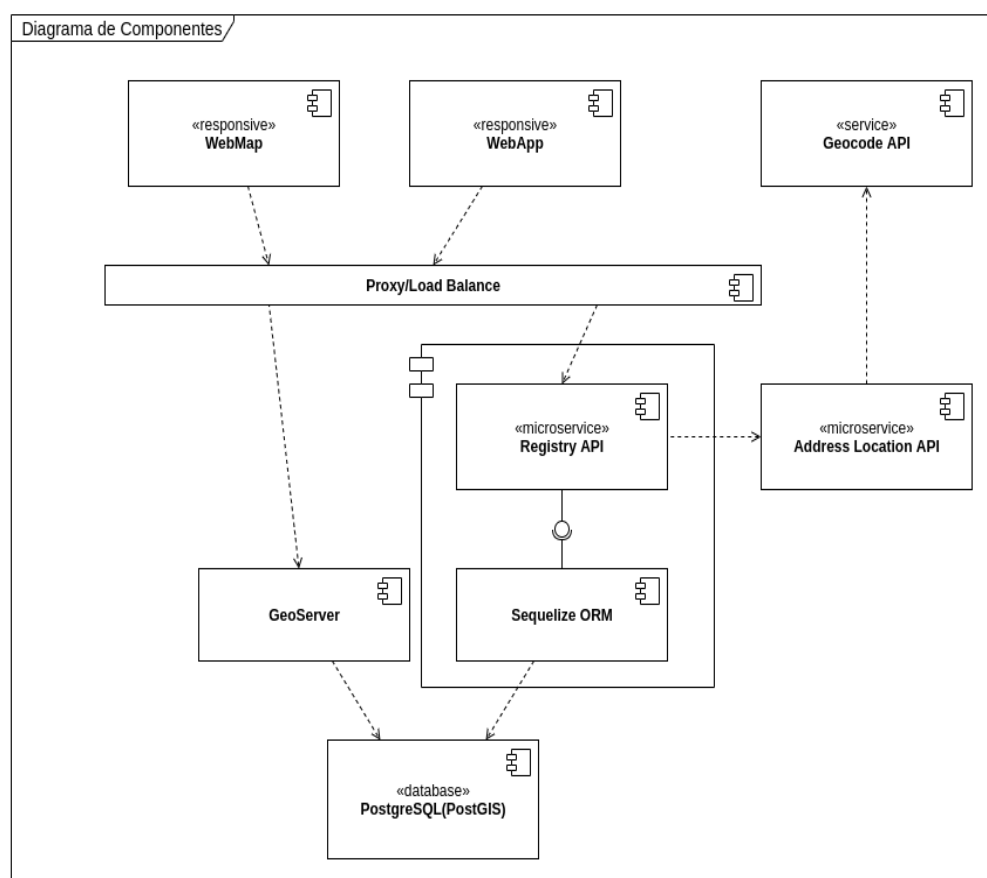


Figura 5 – Diagrama de Componentes

Os componentes arquiteturais apresentados na figura 5, interoperam para proporcionar a execução dos fluxos de fornecimento e consumo de informações, e estão dispostos nas camadas de apresentação, lógica de negócio e persistência de dados. Os detalhes relacionados à implementação de cada um deles, são apresentados a seguir.

**WebApp:** trata-se de um componente que representa a camada de apresentação, com elementos de formulário Web, que utiliza as tecnologias HTML, CSS e JavaScript e deve ser construído seguindo o padrão *Single Page Application*. Assim, deve possuir recurso para acesso a dados via API Rest e capacidade de alteração de partes do conteúdo da página com base nestes dados. É esperada alguma capacidade de validação de entradas do usuário e de dados recebidos por consulta às APIs a fim de melhorar a interatividade. Este componente executará em um navegador Web. Pode-se utilizar bibliotecas JavaScript como JQuery e Bootstrap a fim de facilitar e padronizar a implementação.

**WebMap:** o segundo componente que integra a camada de apresentação, com elementos como painel de apresentação de mapas e controles para realização de consultas incluindo formulário Web, que utiliza as tecnologias HTML, CSS e JavaScript e deve ser construído seguindo o padrão *Single Page Application*. Assim, deve possuir recurso para acesso a dados via API Rest e APIs de dados Geográficos, nos padrões OGC como WMS, WFS e TMS, e capacidade de alteração de partes do conteúdo da página com base nestes dados. É esperada alguma capacidade de validação de entradas do usuário e de dados recebidos por consulta às APIs a fim de melhorar a interatividade. Este componente executará em um navegador Web. Pode-se utilizar bibliotecas JavaScript como Leaflet, JQuery e Bootstrap a fim de facilitar e padronizar a implementação. A biblioteca Leaflet em especial, adota os padrões de acesso a mapas conforme especificação OGC, já mencionada.

**Geocode API:** trata-se de serviço Web de fornecedor externo, para transformar um endereço textual, formatado adequadamente, em uma coordenada geográfica. Uma série de serviços similares estão disponíveis no mercado, mas cada um deles possui suas próprias políticas de disponibilização e sua própria padronização, tanto na formatação das requisições quanto das respostas. Portanto, a fim de padronizar a utilização interna deste tipo de serviço, facilitar a troca de fornecedor e proteger as credenciais de acesso, a proposta inclui um componente extra que encapsule este serviço. As opções testadas foram OpenCage, Geoapify e Google Maps API.



**Address Location API:** este componente deve encapsular o serviço de geocodificação de endereços, de forma a permitir a troca de fornecedor, quando necessário, sem afetar o funcionamento dos demais componentes que dependam deste tipo de serviço. O modelo arquitetural adotado deve ser API Rest.

**Registry API:** este componente deve disponibilizar um serviço de registro de endereços textuais e sua respectiva coordenada geográfica, armazenado em banco de dados com capacidade para tipos de dados espaciais em modelo de dados adequado, conforme definido na sessão 3.3.4. Este serviço deve incluir a disponibilização destes mesmos dados para consulta e atualização. É imprescindível a utilização de um *middleware* para acesso ao SGBD a fim de permitir a troca do gerenciado de bases de dados, se necessário. O modelo arquitetural adotado deve ser API Rest.

**Geoserver:** representado por um software de terceiro, este componente é responsável por disponibilizar API padronizada, conforme especificações do Consórcio Geoespacial Aberto (OGC, na sigla em inglês). O interesse específico para este projeto é relacionado aos padrões Web Map Service (WMS) e Web Feature Service (WFS), APIs para acesso a dados geoespaciais em formato gráfico de imagens, png ou jpeg, e em formato GeoJson. Estes padrões de serviço permitem a realização de consultas parametrizadas, facilitando o processo de disponibilização e obtenção de mapas obedecendo a filtros.

**PostgreSQL/PostGIS:** SGDB consolidado no mercado como uma das melhores, se não a melhor opção de gerenciador de bancos de dados relacionais com capacidades de armazenar, manter e disponibilizar grandes volumes de dados incluindo tipos de dados geográficos, dentre as opções OpenSource. Além destas características, a escolha se deve pelo fato de que o Geoserver possui drive nativo para este SGDB, facilitando a integração.

**Proxy/Load balance:** aqui, o representante é o Nginx, um servidor Web com muitas características interessantes, também consolidado no mercado, e incluindo as op-

ções de operação como *proxy* leve de requisições HTTP e balanceamento de carga de fácil configuração. A adoção deste componente é peça chave na intermediação da comunicação entre os componentes internos da arquitetura e os componentes da camada de apresentação, uniformizando as URLs e permitindo a distribuição das requisições incluindo métodos como: ip-hash, round-robin e least-connected, entre réplicas de instâncias de um serviço, admitindo ampliação da vazão.

### 3.3.3 Diagrama de Implantação

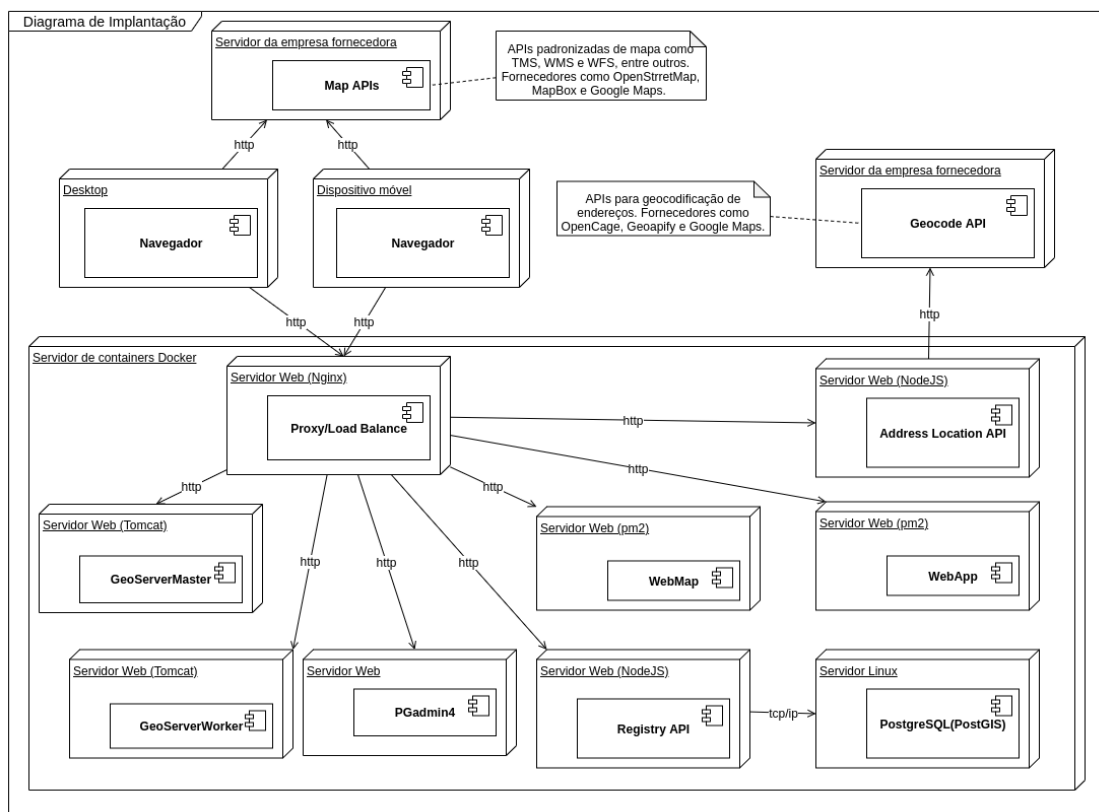


Figura 6 – Diagrama de Implantação.

A aplicação poderá ser implantada em qualquer plataforma que suporte a execução de *containers* Docker, podendo ocorrer em um servidor único ou em vários servidores, físicos ou virtualizados, dentro do mesmo *datacenter* ou em *datacenters* distribuídos geograficamente, bastando possuir comunicação de rede entre eles, com possibilidade de configuração de rotas de acesso via TCP/IP e HTTP.

É destaque a utilização de componentes de software gratuitos como a própria plataforma Docker que pode incluir a orquestração dos serviços empacotados em *containers*, ferramentas como Node.js para o ambiente de execução de JavaScript no servidor, os servidores Web Nginx, pm2 e Tomcat, as aplicações pgAdmin4 e GeoServer além do SGBD PostgreSQL com suporte espacial PostGIS, são todos Open-Source, com boa documentação, alto grau de maturidade e consolidados no mercado.

Os componentes externos como APIs de mapas e de Geocodificação de endereços possuem versões de teste gratuitas com limitações, geralmente em número de requisições diárias, sendo um ponto de atenção e que pode exigir pagamento caso o cenário de execução atinja esses limites. Cada fornecedor externo possui sua política de gratuidade, tempo de teste e planos de pagamento, devendo-se avaliar caso a caso.

A adoção de infraestrutura como código (IaC, na sigla em inglês), adotada como parte do desenvolvimento dos componentes aqui propostos, é enorme facilitador do processo de implantação e exige pouco trabalho para colocar toda pilha de serviços em operação.

Cada componente implementado deve incluir o código necessário para a criação da imagem de *container* Docker de forma a disponibilizar um pacote autossuficiente com relação ao ambiente de execução, prevendo alguma parametrização que permita a injeção de dados sensíveis como *tokens* de acesso a APIs, credenciais de acesso a bancos de dados entre outras, que os torne flexível o suficiente para que possam ser executados em qualquer infraestrutura que disponibilize estes ambientes, como Docker compose, Docker Swarm, Kubernetes entre outros.

Foi gerado um repositório de código específico para manter o projeto de implantação que inclui a pilha de serviços completa e arquivos de configuração e *script* para execução. Mais informações estão na sessão 4.1, Implementação.

### 3.3.4 Modelo de Dados

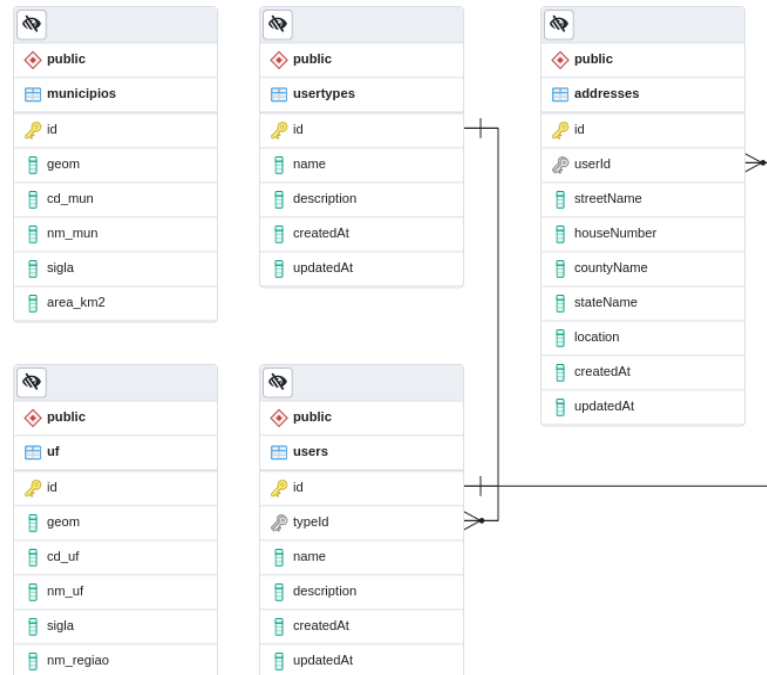


Figura 7 – Diagrama de Entidade Relacionamento (ER)

A figura 7 mostra as entidades que fazem parte do modelo de dados adotado para compor a camada de persistência. Nele podemos observar a existência das entidades User, UserType e Address, que mapeiam as mesmas classes definidas nos diagramas de classe anteriormente apresentados.

Somam-se a estas entidades, mais duas representações utilizadas para compor os mapas de Estados e Municípios, e que participam do processo de identificação geográfica do Município onde se encontra uma coordenada de endereço de usuários. Neste contexto há uma relação geográfica implícita, mas que não se materializa no modelo relacional, apenas de forma lógica em operação geográfica realizada por meio de funções do banco de dados, sob demanda.

## 4. Prova de Conceito (POC) e Protótipo Arquitetural

Nesta sessão serão apresentados os artefatos gerados na implementação da prova de conceito, incluindo descrição de quais componentes foram desenvolvidos, as tecnologias adotadas e as ferramentas utilizadas.

## 4.1. Implementação

A implementação dos artefatos de software seguiu as boas práticas de implementação de código com separação adequada de responsabilidades, simplicidade, uso equilibrado de bibliotecas de terceiros, inserção de comentários em código, adoção de ferramentas que facilitem a escrita e o processo de depuração, ferramenta de versionamento de código, ferramenta de gerenciamento de dependências e ferramenta de empacotamento de artefatos.

O ambiente de desenvolvimento foi montado em plataforma Linux (Ubuntu 16.04), e foram adotadas as seguintes ferramentas:

- Visual Studio Code (Microsoft);
- Navegador Google Chrome (teste e depuração com DevTools);
- Node.js e NPM (Execução de JavaScript e gerenciamento de dependências);
- Geoserver (provedor de APIs de mapas);
- Postman (teste de APIs);
- GitHub (versionamento de código);
- DockerHub (repositório de imagens Docker);
- pgAdmin4 (administração do banco de dados PostgreSQL);
- PostgreSQL/PostGIS (gerenciado de bancos de dados relacionais com extensão geográfica);
- Docker e Docker compose (gerar imagens e executar *containers*);

As bibliotecas e linguagem utilizadas foram:

- JavaScript, HTML e CSS;
- SQL;
- yaml;
- Sequelize;
- Leaflet;

- JQuery;
- Axios;
- Shell Script;
- Json e GeoJson;
- Nginx;
- pm2;
- Tomcat;

Os casos de uso implementados foram: **UC01**, **UC02**, **UC03** e **UC04**, conforme descrito na sessão 3.2 deste documento.

A seguir são apresentados *prints* das telas de interface de usuário que representam os casos de uso.

**Formulário de registro de endereço**

Dr. Machuca, seu endereço já foi cadastrado.

Todos os campos são obrigatórios.

usuários (selecione) ▼ Dr. Machuca

logradouro  
Rua Luis Carlos Samartini

número  
126

município  
São José dos Campos

estado  
São Paulo

procurar localização

Seu endereço é aqui?  
Caso necessário, arraste para ajustar o local.  
Rua Luis Carlos Samartini, 126  
São José dos Campos - São Paulo

Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox, POC of TCC © PUC

A simple client to test the Address and Geocode APIs  
— Fork me on GitHub

Figura 8 – Tela de cadastro de endereço e localização geográfica (**UC01** e **UC02**)

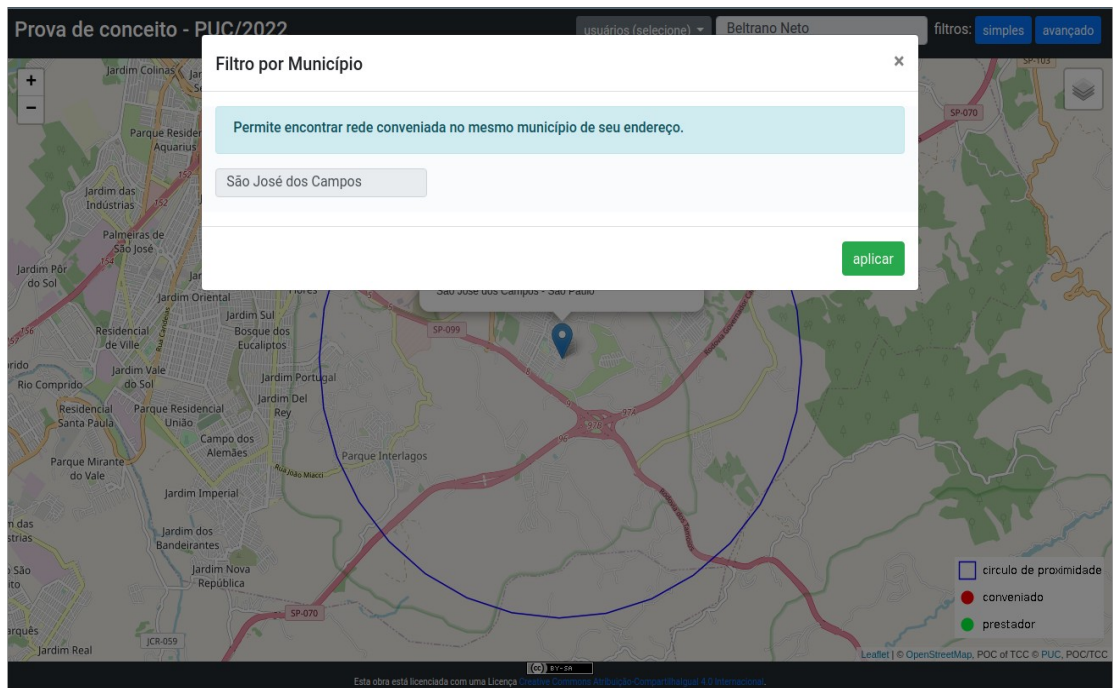


Figura 9 – Tela para consultar localização de prestadores e conveniados (UC03)

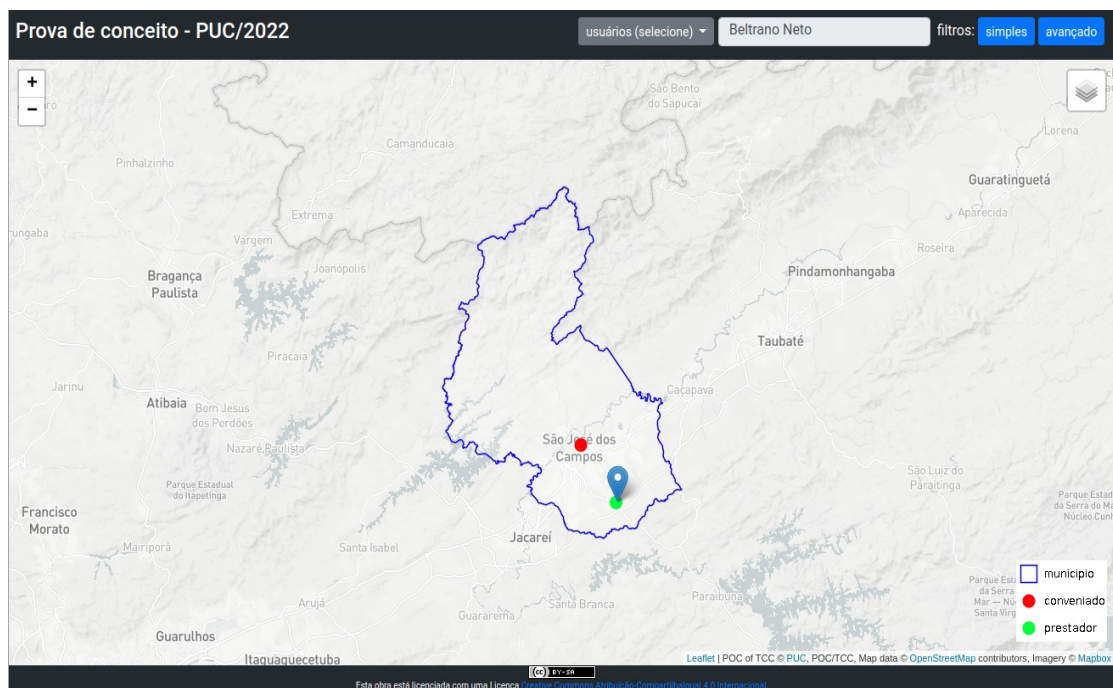


Figura 10 – Tela de resultado da localização de prestadores e conveniados (UC03)

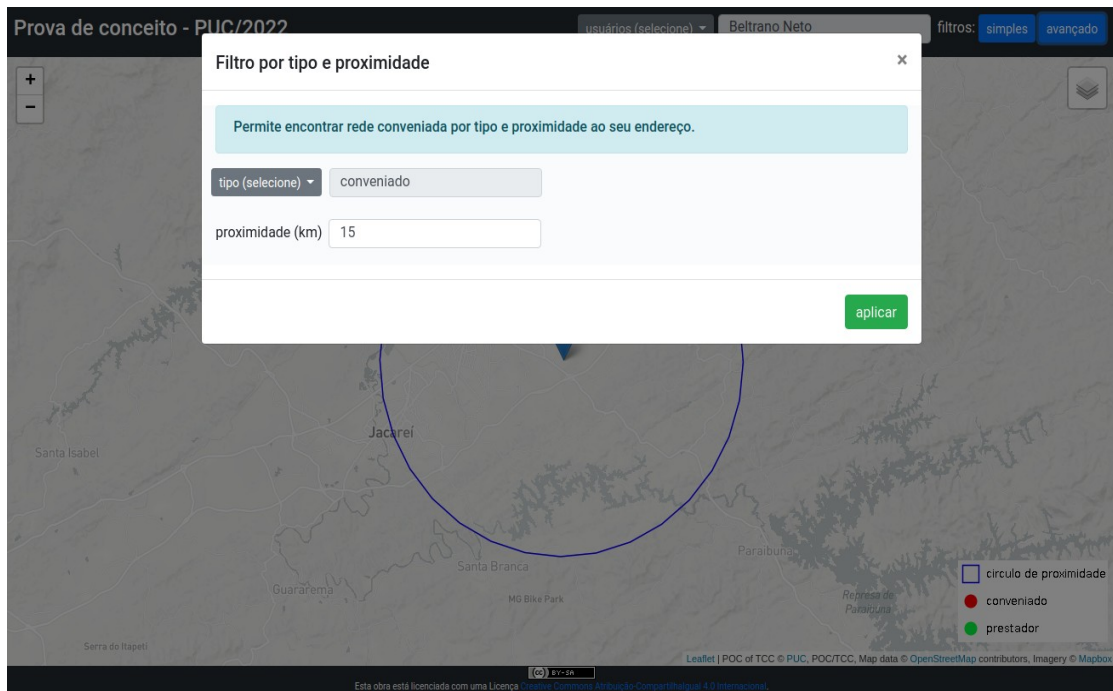


Figura 11 – Tela para filtrar localização de prestadores e conveniados (UC03)

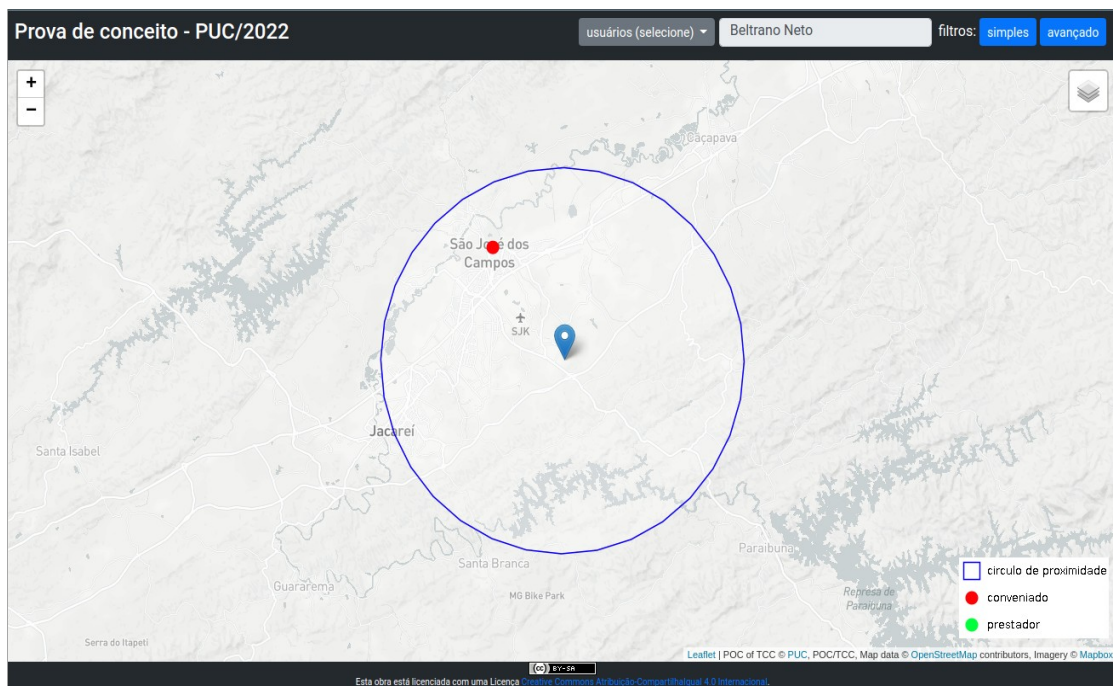


Figura 12 – Tela de resultado da localização de prestadores e conveniados (UC04)

Os requisitos não funcionais escolhidos para avaliação foram: **RNF05**, **RNF07** e **RNF10**. A seguir são apresentados os critérios de aceitação para eles.



RNF07: Desempenho – O sistema deve apresentar as interfaces de usuário e responder às requisições de operações subsequentes em tempo razoável.

Critérios:

- A apresentação inicial de telas pode ocorrer em tempos inferiores a 1 segundo;
- O tempo médio de resposta do sistema aos usuários deve ser de 1 segundo;
- O tempo máximo de resposta do sistema aos usuários deve ser de 5 segundos;

RNF10: Acessibilidade – As interfaces devem se adaptar a tamanhos diferentes de tela, exibindo menor ou maior quantidade de elementos quando necessário.

Critérios:

- As interfaces do sistema devem apresentar facilidade de navegação e os elementos gráficos devem se adaptar de acordo com a resolução do dispositivo;
- O sistema deve manter o padrão gráfico para cores e escalas em suas interfaces de usuário;
- O sistema deve ser compatível com os principais navegadores do mercado como: Google Chrome e Firefox;

RNF11: Interoperabilidade – As camadas de serviços, representadas por APIs, devem prover formatos de saída que facilite a integração entre componentes e sistemas de diferentes tecnologias.

Critérios:

- Os recursos de API do sistema devem retornar as informações em formato JSON;

- Devem ser apresentados códigos de status da resposta compatíveis com o padrão de códigos HTTP, como 200 em sucesso e 404 quando o recurso não existir;

O processo de implantação da prova de conceito foi implementado utilizando IaaS, e os códigos necessários para a execução estão disponíveis no GitHub em repositório específico denominado “**tcc-poc-stack**” e o link de acesso foi incluído no apêndice. Neste repositório estão descritos os elementos e configurações necessárias para a execução de toda a pilha de serviços. São necessários conhecimentos de ambiente de execução Docker e gerenciamento de servidores Linux para completar esta tarefa.

## 4.2 Interfaces e APIs

As APIs implementadas neste trabalho foram testadas e documentadas usando o software Postman. Cada repositório de código de API inclui um arquivo em formato JSON que representa a coleção de testes exportada pelo Postman e que pode ser importada para a repetição dos testes em ambiente local ou ajustando a URL base caso a implantação tenha sido feita em outro endereço.

Estas coleções foram publicadas e estão disponíveis em:

- <https://documenter.getpostman.com/view/6754794/Uyr5nykq>
- <https://documenter.getpostman.com/view/6754794/Uyr5nykr>

## 5. Avaliação da Arquitetura

A seguir, são apresentados os cenários e resultados das avaliações de conformidade das abordagens arquiteturais e da implementação dos casos de uso propostos.

### 5.1. Análise das abordagens arquiteturais

Os atributos de qualidade analisados foram:

Atributos de Qualidade	Cenários	Importância	Complexidade
------------------------	----------	-------------	--------------

Desempenho	Cenário 1: O sistema deve responder em tempo razoável.	A	M
Acessibilidade	Cenário 2: O sistema deve prover boa usabilidade.	M	B
Interoperabilidade	Cenário 3: O sistema deve se comunicar com sistemas de outras tecnologias.	A	M

## 5.2. Cenários

Cenário 1: Desempenho: Ao acessar as interfaces gráficas de usuário, o sistema deve responder em tempo aceitável, estando disponível para interação com o usuário em no máximo 5 segundos e em média em até 1 segundo.

Cenário 2: Acessibilidade: Ao utilizar dispositivos com tamanhos de tela diferentes o sistema deve adaptar os componentes de tela automaticamente de modo que não haja prejuízo no acesso à funcionalidade.

Cenário 3: Interoperabilidade: Ao acessar a URL do serviço de registro de endereços via HTTP GET, o mesmo deve retornar as informações no formato JSON.

## 5.3. Evidências da Avaliação

Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O sistema deve rápido.
Preocupação:	
O sistema deve responder em tempo razoável às requisições.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Um usuário requisita a interface do sistema de cadastramento de endereços.	
Mecanismo:	
Criar interface Web para atender às requisições do sistema de cadastramento de endereços.	
Medida de resposta:	
Renderizar a interface em no máximo 5 segundos e em média em 1 segundo.	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade na rede pode deixar a conexão lenta ou mesmo a perda de pacotes.
Pontos de Sensibilidade:	A configuração do servidor de páginas Web e o proxy devem estar configurados para

	compactar os dados antes do envio ao navegador.
Tradeoff:	Não há

Evidências do cenário 1:

A ferramenta Lighthouse, que é parte do kit de ferramentas de desenvolvimento, DevTools do navegador Google Chrome, foi utilizada para esta avaliação.

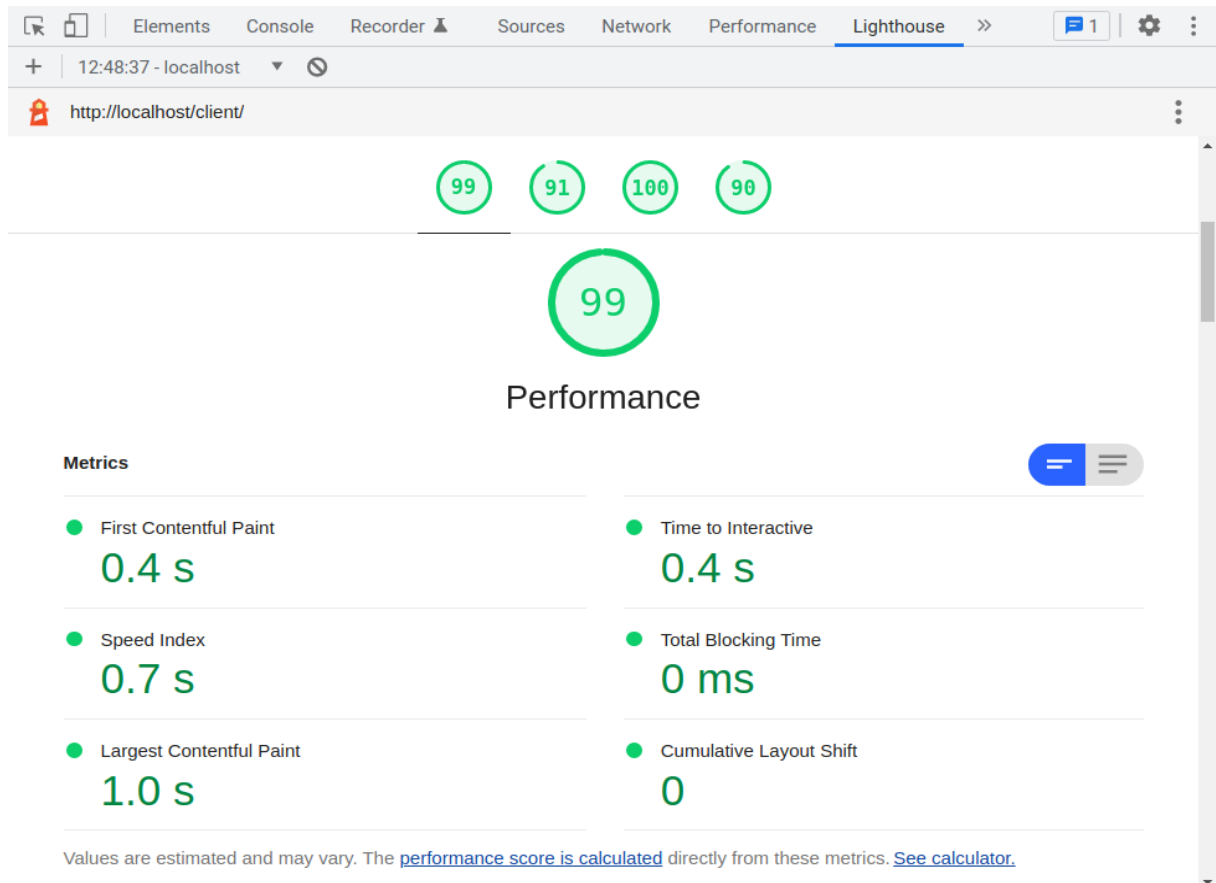


Figura 13 – Tela de resultado da avaliação de desempenho

Atributo de Qualidade:	Acessibilidade
Requisito de Qualidade:	O sistema deve adaptar as interfaces a tamanhos de tela diferentes.
Preocupação:	O sistema deve redimensionar os elementos de tela de acordo com as dimensões de tela do dispositivo do usuário.
Cenário(s):	
Cenário 2	
Ambiente:	

Sistema em operação normal	
Estímulo:	
O usuário entra com a URL do recurso que deseja acessar.	
Mecanismo:	
Criar interfaces de usuário utilizando práticas de detecção de dimensões e bibliotecas de suporte a responsividade incluindo uso de CSS.	
Medida de resposta:	
A interface não deve impedir a visualização e interação de seus elementos e cores.	
Considerações sobre a arquitetura:	
Riscos:	Alguns dispositivos podem apresentar falhas na apresentação dos elementos de tela devido a utilização de navegadores que não respeitem os padrões W3C.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

#### Evidências do cenário 2:

Outra ferramenta, Device Toolbar, que é parte do kit de ferramentas de desenvolvimento do navegador Google Chrome, foi utilizada para esta avaliação. Foram utilizados 3 dispositivos distintos no simulador, segue os resultados.

Dimensions: Nest Hub ▾ 1024 x 600 100% ▾ No throttling ▾

## Formulário de registro de endereço

⚙️

**Todos os campos são obrigatórios.**


usuários (selecione) ▾

logradouro

número

município

estado



Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox, POC of TCC © PUC

A simple client to test the Address and Geocode APIs  
— [Fork me on GitHub](#)

Figura 14 – Tela de resultado da avaliação de acessibilidade (Tablet Nest Hub)

Dimensions: Surface Duo ▾ 540 x 720 100% ▾ No throttling ▾

## Formulário de registro de endereço

⚙️

**Todos os campos são obrigatórios.**

usuários (selecione) ▾

logradouro

número

município

estado




Figura 15 – Tela de resultado da avaliação de acessibilidade (Surface Duo)

Dimensions: iPhone SE ▾ 375 x 667 100% ▾ No throttling ▾

### Formulário de registro de endereço

⚙️

**Todos os campos são obrigatórios.**

usuários (selecione) ▾

logradouro

número

município

estado

Figura 16 – Tela de resultado da avaliação de acessibilidade (iPhone SE)

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve se comunicar com outras tecnologias.
Preocupação:	O sistema deve ter como resposta a uma requisição uma saída de fácil leitura por outro componente.
Cenário(s):	
Cenário 3	
Ambiente:	
Sistema em operação normal	
Estímulo:	Uma aplicação de teste de APIs envia uma requisição ao serviço de registro de endereços.
Mecanismo:	
Criar um serviço REST para atender às requisições do sistema de cadastramento de endereços	
Medida de resposta:	
Retornar os dados requisitados no formato JSON	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade na rede pode deixar a

	conexão lenta ou mesmo a perda de pacotes.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Evidências do cenário 3:

Para este teste foi utilizado o software Postman que permitiu apresentar a seguinte evidência.

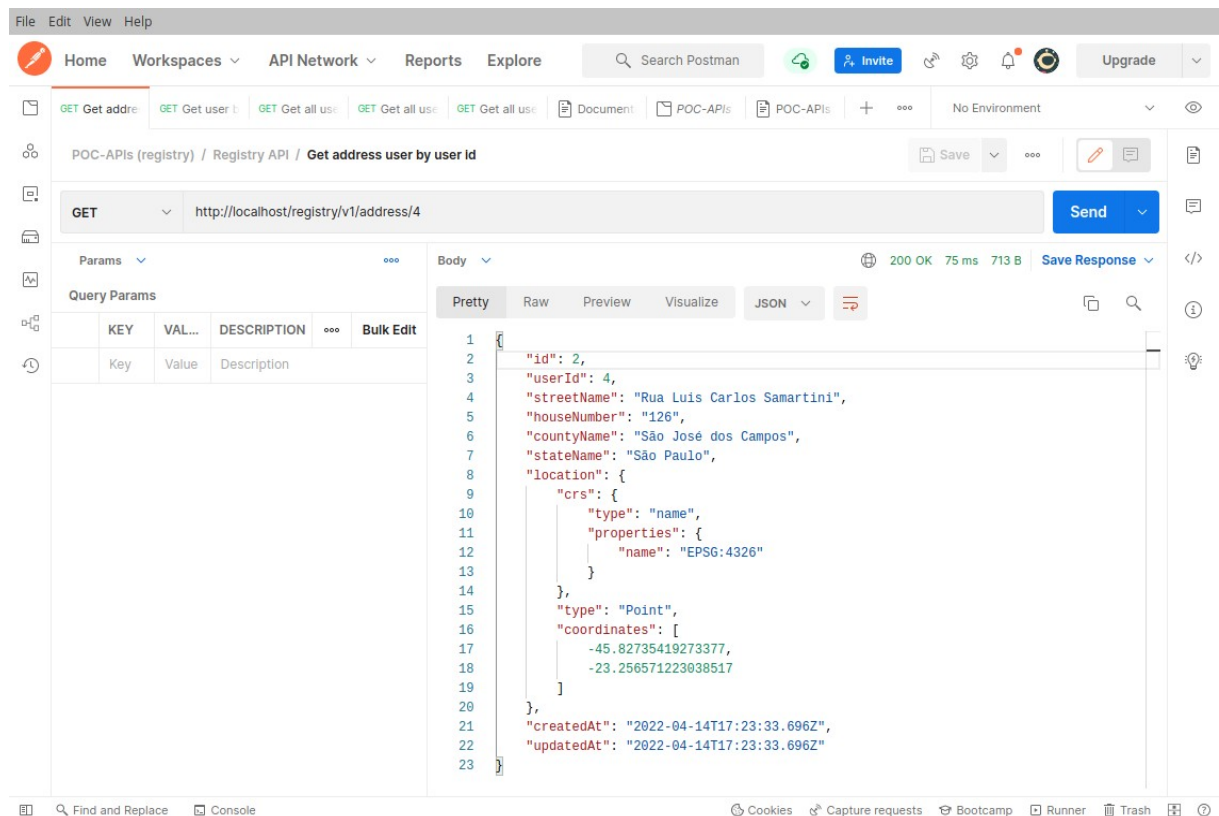


Figura 17 – Tela de resultado da avaliação de interoperabilidade

## 5.4. Resultados

De forma geral a arquitetura se mostrou eficiente para resolver os problemas apresentados, conforme demonstrado nas evidências apresentadas na sessão anterior.

Embora as interfaces gráficas tenham apresentado responsividade ao renderizar em telas de dispositivos diferentes, neste tópico cabe uma ressalva. Dependendo das funcionalidades demandadas, pode ser mais adequado que se implemente aplicati-



vos utilizando recursos nativos das plataformas móveis a fim de alcançar melhores resultados na experiência dos usuários ao utilizar certos recursos.

Outros cenários devem ser identificados e incluídos nos testes a fim de avaliar requisitos de qualidade como facilidade na manutenção, na implantação e na usabilidade.

Requisitos Não Funcionais	Testado	Homologado
RNF06: O sistema deve apresentar as interfaces de usuário e responder às requisições de operações subsequentes em tempo razoável.	SIM	SIM
RNF10: As interfaces devem se adaptar a tamanhos diferentes de tela, exibindo menor ou maior quantidade de elementos quando necessário.	SIM	SIM
RNF11: As camadas de serviços, representadas por APIs, devem prover formatos de saída que facilite a integração entre componentes e sistemas de diferentes tecnologias.	SIM	SIM

## 6. Conclusão

Foi um desafio interessante desenvolver software realizando grande parte das etapas formais indicadas na literatura e comprovadamente eficazes para redução dos riscos associados com esta atividade. Os resultados alcançados superaram minhas expectativas principalmente com relação ao atendimento pleno dos objetivos sem necessidade de retrabalho. Infelizmente não é prática comum em boa parte das empresas no Brasil, ao menos é o que percebo ao conversar com colegas de profissão, as vezes por desconhecimento, mas principalmente pela grande escassez de profissionais, que cada vez mais pressionados por resultados rápidos, são forçados a pular muitas das etapas aqui exercitadas.

Embora os resultados tenham se mostrado promissores, foi exercitada apenas uma pequena parte da arquitetura proposta, cuja amplitude exige várias iterações e refinamentos para melhor o entendimento das reais necessidades, contemplando outros cenários a serem prototipados, a fim de guiar de forma mais assertiva as equipes de implementação e operação. Uma série de desafios podem surgir ao se analisar mais detalhadamente os demais módulos do sistema, principalmente em se tra-

tando da adoção de softwares de terceiros por serem tratados como caixas preta neste momento.

Um ponto negativo da solução proposta é o fato de tratar o módulo de informações cadastrais como o novo guardião de grande parte das informações que provavelmente já existem espalhadas nas bases dos sistemas legados. Caso este desafio não seja bem endereçado no sentido de construir acessos e operacionalizar a migração, pode ser mais um problema do que solução.

## REFERÊNCIAS

Martin, Robert C. **Arquitetura limpa: O guia do artesão para estrutura e design de software**. Cidade: Rio de Janeiro ALTA BOOKS, 2019.

## APÊNDICES

Este é o link da organização que contem os 6 repositórios criados para organizar e manter os seguintes projetos:

<https://github.com/Practice-of-PUC-course>

“tcc-docs”: contem os arquivos originais associados a este documento;

“tcc-webmap-app”: contem os códigos do projeto da aplicação de mapas;

“tcc-address-form”: contem os códigos do projeto da aplicação de cadastro de endereços;

“tcc-address-geocode”: contem os códigos do projeto da api de geocodificação de endereços;

“tcc-registry-api”: contem os códigos do projeto da api de cadastro de endereços;

“tcc-poc-stack”: contem os códigos do projeto para a implantação da pilha de serviços;

Link para o vídeo da apresentação.

[https://drive.google.com/file/d/1CQjIRYMUG\\_U-RqmWX1--tSl0tQQGgKaa/view?usp=sharing](https://drive.google.com/file/d/1CQjIRYMUG_U-RqmWX1--tSl0tQQGgKaa/view?usp=sharing)

