PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído

Projeto Integrado

Relatório Técnico

Rede Social de Distribuição de Cestas Básicas

Txai Vieira Garcia

Belo Horizonte
Outubro, 2022

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

1. Introdução

Ocupando a nona posição no ranking global, o Brasil figura entre os dez países mais desiguais do mundo.

Esta situação agravou-se com o surgimento da pandemia de COVID-19, a qual trouxe, além das implicações sanitárias, consequências econômicas e sociais, como o aumento do desemprego, que em 2020 atingiu 14,4%, o que elevou ainda mais a situação de desigualdade do país e trouxe muitas famílias a uma situação de vulnerabilidade social, sendo que, segundo o último levantamento, 27.7% dos brasileiros vivenciavam situação de insegurança alimentar grave ou moderada em 2020. Neste contexto, torna-se ainda mais importante o trabalho de distribuição de cestas básicas, promovido, geralmente, por instituições religiosas, organizações não governamentais e por iniciativas comunitárias ou individuais.

Este trabalho envolve a participação de três atores: os distribuidores, que concentram as atividades de arrecadação de alimentos, montagem e distribuição das cestas; os doadores, que são as pessoas interessadas em doar alimentos e os recebedores, que são as famílias auxiliadas. Não havendo meios de coordenar estes agentes, é comum encontrar alguns problemas principais em todas as iniciativas de distribuição de cestas:

- A divulgação da arrecadação e da distribuição é restrita;
- Os doadores, por não terem informações suficientes, acabam doando itens de maneira desproporcional (e.g. doam muito feijão, mas o distribuidor está com falta de arroz), causando desperdício;
- Em cada local é necessário que o recebedor faça um cadastro e passe por uma avaliação, o que é um desperdício de tempo para os recebedores e para os distribuidores;
- A demanda de cestas geralmente excede a oferta destas, e sem um meio de divulgar esta informação, muitos recebedores acabam indo ao local de distribuição sem saber se poderão receber uma cesta.

Uma plataforma, que permitisse a centralização de informações e a interação entre todos os três atores envolvidos no processo, permitiria redução dos desperdícios de recursos e permitiria que mais famílias fossem auxiliadas, sobretudo em situações de picos de demanda, como no caso de desastres ou eventos como a recente pandemia. Além disso, as informações coletadas, poderiam servir de auxílio para estudos e na tomada de decisões de ações de combate à fome.

O objetivo do presente trabalho é apresentar a descrição arquitetural de uma plataforma deste tipo, denominada ReCeBa (Rede social de distribuição de Cestas Básicas), que deve atuar como centralizadora a fim de resolver os problemas elencados acima.

Os objetivos específicos a serem alcançados são:

- Elencar as restrições arquiteturais e os requisitos necessários à construção do sistema;
- Desenhar uma arquitetura capaz de prover a segurança necessária dos dados dos usuários e de ser escalável, principalmente em picos de utilização, como durante a ocorrência de desastres;
- Desenhar uma interface que seja intuitiva e leve, considerando o público-alvo da aplicação;
- Demonstrar a viabilidade de uma plataforma do tipo, por meio de prova de conceito e posterior protótipo;
- Documentar a construção do sistema, a fim de permitir um trabalho de expansão deste no futuro;

2. Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

Datas		**Atividade / Tarefa**	**Produto / Resultado**
De	**Até**		
03/05/2022	07/05/2022	1. Levantamento de ideias de possíveis projetos	Tema do projeto integrado
08/05/2022	10/05/2022	2. Estudo do contexto e problemática do tema proposto	Introdução deste documento
08/05/2022	10/05/2022	3. Elaboração da introdução	Introdução deste documento
11/05/2022	11/05/2022	4. Elaboração do cronograma	Cronograma definido
12/05/2022	15/05/2022	5. Levantamento das restrições arquiteturais	Lista de restrições arquiteturais
16/05/2022	20/05/2022	6. Levantamento dos requisitos funcionais	Lista de requisitos funcionais
21/05/2022	24/05/2022	7. Levantamento dos requisitos não funcionais	Lista de requisitos não funcionais
25/05/2022	30/05/2022	8. Definição dos mecanismos arquiteturais	Lista de mecanismos arquiteturais
01/06/2022	03/06/2022	9. Elaboração do diagrama de contexto	Diagrama de contexto da solução
04/06/2022	05/06/2022	10. Revisão deste documento	Etapa 1 do Relatório técnico revisada
06/06/2022	15/06/2022	11. Elaboração do vídeo de apresentação	Vídeo de 3 minutos apresentando o contexto do projeto e a solução proposta
20/06/2022	23/06/2022	12. Elaboração do diagrama de container	Diagrama de container da solução
24/06/2022	26/06/2022	13. Elaboração do diagrama de componentes	Diagrama de componentes da solução
27/06/2022	08/07/2022	14. Elaboração do wireframe da prova de conceito	Wireframe navegável da prova de conceito com a implementação de 3 requisitos funcionais
09/09/2022	22/09/2022	15. Criação do front end da prova de conceito	Frontend disponibilizado em nuvem, implementado conforme wireframe
23/09/2022	06/10/2022	16. Criação do back end da prova de conceito	Backend disponibilizado em nuvem, com todas as interfaces necessárias para a integração com o frontend
07/10/2022	12/10/2022	17. Integração do backend e do frontend	Frontend e backend integrados
13/10/2022	15/10/2022	18. Revisão do relatório técnico	Etapa 2 do relatório técnico revisada
16/10/2022	18/10/2022	19. Levantamento dos atributos de qualidade e cenários da avaliação da arquitetura	Lista de itens de análise das abordagens arquiteturais
19/10/2022	26/10/2022	20. Detalhamento dos cenários levantados	Lista detalhada de cenários

		na etapa anterior	
27/10/2022	16/10/2022	21. Levantamento das evidências relativas a cada cenário	Lista de evidências
17/11/2022	20/11/2022	22. Construção da seção de resultados obtidos	Texto e tabela de resultados obtidos, evidenciando RNFs homologados e pontos fortes e fracos da arquitetura
21/11/2022	23/11/2022	23. Avaliação dos resultados obtidos	Quadro resumo contendo as avaliações
24/11/2022	27/11/2022	24. Elaboração da conclusão	Conclusão do relatório
28/11/2022	30/11/2022	25. Revisão do relatório técnico	Relatório técnico completo e revisado
01/12/2022	15/12/2022	26. Elaboração do vídeo explicativo	Vídeo de 5 minutos apresentando o projeto implementado e a arquitetura desenhada

3. Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macro arquitetura da solução.

3.1 Restrições Arquiteturais

As restrições arquiteturais foram levantadas, considerando como público-alvo pessoas de baixo poder aquisitivo. Além disso, também foi considerado que o projeto não possui um objetivo comercial, então deve permitir envolvimento da comunidade para expansão e manutenção.

ID	Descrição
R1	Por não ser uma aplicação com clientes empresariais, o sistema deve permitir acesso público através da internet
R2	O público-alvo possui, majoritariamente, acesso à internet através de celulares. Então o sistema deve prover uma solução de frontend mobile, ou através de aplicativos ou através de web apps
R3	Em muitos lugares, o acesso à internet pode ser instável. Portanto, o sistema deve prover algumas funcionalidades offline
R4	Serão trafegados dados pessoais identificáveis, portanto o sistema deve prover mecanismos de segurança para proteção destes dados
R5	Não dispondo de investimentos, o sistema deve utilizar ao máximo tecnologias open source e gratuitas
R6	O desenvolvimento do sistema deve utilizar controle de versão através de repositório Git público e deve ser estruturado para permitir o envolvimento da comunidade open source

3.2 Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve permitir o acesso de usuários não cadastrados	B	A
RF02	O sistema deve permitir o auto cadastramento do usuário	B	A
RF03	O sistema deve permitir que usuários cadastrem iniciativas de distribuição de cestas (IDC)	M	A
RF04	As IDCs devem conter nome, endereço, meta de cestas do mês, quantidade de cestas disponíveis, itens faltantes, composição das cestas e data de entrega para o mês corrente	B	A
RF05	As IDCs devem ser acessíveis por usuários cadastrados e não	B	A

	cadastrados		
RF06	O sistema deve permitir que os criadores das IDCs definam outros usuários como administradores destas últimas. Os administradores terão as mesmas atribuições e permissões dos criadores	A	B
RF07	O sistema deve permitir que os administradores das IDCs cadastrem metas de cestas básicas	B	M
RF08	O sistema deve permitir que os administradores das IDCs cadastrem composições de cestas básicas	B	M
RF09	O sistema deve permitir que os administradores das IDCs cadastrem itens recebidos por doação	B	M
RF10	O sistema deve montar cestas básicas, automaticamente, conforme a composição da cesta, a meta de distribuição e a quantidade de itens recebidos	M	B
RF11	O sistema deve exibir as IDCs cadastradas através de mapa e lista	A	B
RF12	O sistema deve permitir que usuários cadastrados, criem pedidos de recebimento de cestas básicas	B	M
RF13	O sistema deve permitir que o usuário envie o pedido para até 5 IDCs	M	M
RF14	O sistema deve exibir, para os administradores das IDCs, os pedidos de recebimento pendentes	B	M
RF15	O sistema deve permitir que os administradores das IDCs aceitem o pedido de recebimento, rejeitem-no ou o coloquem em fila de espera	M	M
RF16	Ao aceitar um pedido de recebimento, o administrador da IDC terá 24h de tolerância para poder desfazer a ação	A	B
RF17	O sistema deve avisar o administrador da IDC caso ele aceite um pedido de recebimento e não tenha cestas suficientes para atender o pedido	B	B
RF18	Quando uma cesta for completada, o sistema deve exibir a fila de pedidos pendentes para que ela possa ser reavaliada pelo administrador	M	B
RF19	O sistema deve notificar o usuário por e-mail, notificação push e SMS quando um pedido de recebimento for aceito	A	A
RF20	O sistema deve notificar o usuário por e-mail, notificação push e SMS 3 dias antes da data de entrega da cesta básica	A	M
RF21	O sistema deve destacar, para o administrador da IDC, usuários que já tiveram seus pedidos aceitos por outras IDCs	M	A
RF22	O sistema deve retirar esta marcação um mês após o	M	A

	recebimento da cesta pelo usuário		
RF23	O sistema deve permitir que o administrador de IDC registre o recebimento da cesta pelo usuário	B	A

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

ID	Descrição	Prioridade B/M/A
RNF01	O sistema deve estar disponível das 6h às 24h (18h) durante os sete dias da semana	A
RNF02	A interface de usuário deve ser acessível, devendo ser atendidos os níveis A e AA definidos no WCAG 2.1	M
RNF03	A interface de usuário deve possuir design responsivo	A
RNF04	O sistema deve atender até 2000 usuários simultaneamente sem degradação da performance	B
RNF05	A interface web deve rodar nas versões mais recentes do Microsoft Edge, Mozilla Firefox e Google Chrome	B
RNF06	O tempo de resposta da interface de usuário deve ser de até 1 segundo 90% do tempo	M
RNF07	O sistema deve prover segurança dos dados pessoais identificáveis, permitindo que apenas os donos desses dados possam lê-los e alterá-los	M
RNF08	100% dos dados pessoais identificáveis devem ser encriptados em repouso e em trânsito	A

3.4 Mecanismos Arquiteturais

Análise	*Design*	Implementação
Persistência	Bando de dados NOSQL	MongoDB
Persistência	Banco de dados relacional	PostgreSQL
Front end	Single Page Application	Vue.js
Back end	Micro serviços	Spring Boot
Desenvolvimento	Plataforma de desenvolvimento orientada a objetos	Java
Integração	API Restful	HTTP+JSON
Integração	API Gateway	Kong
Autorização e Autenticação	OIDC/OAuth2	Keycloack
Log do sistema	Plataforma de gerenciamento de logs	Graylog

Teste de Software	Testes de unidade	JUnit
Deploy	Conteinerização	Docker/Kubernets
Deploy	CI/CD	Jenkins

4. Modelagem Arquitetural

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à implementação da prova de conceito (seção 5).

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: https://c4model.com/ e aqui: https://www.infoq.com/br/articles/C4-architecture-model/. Dos quatro nível que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

4.1 Diagrama de Contexto

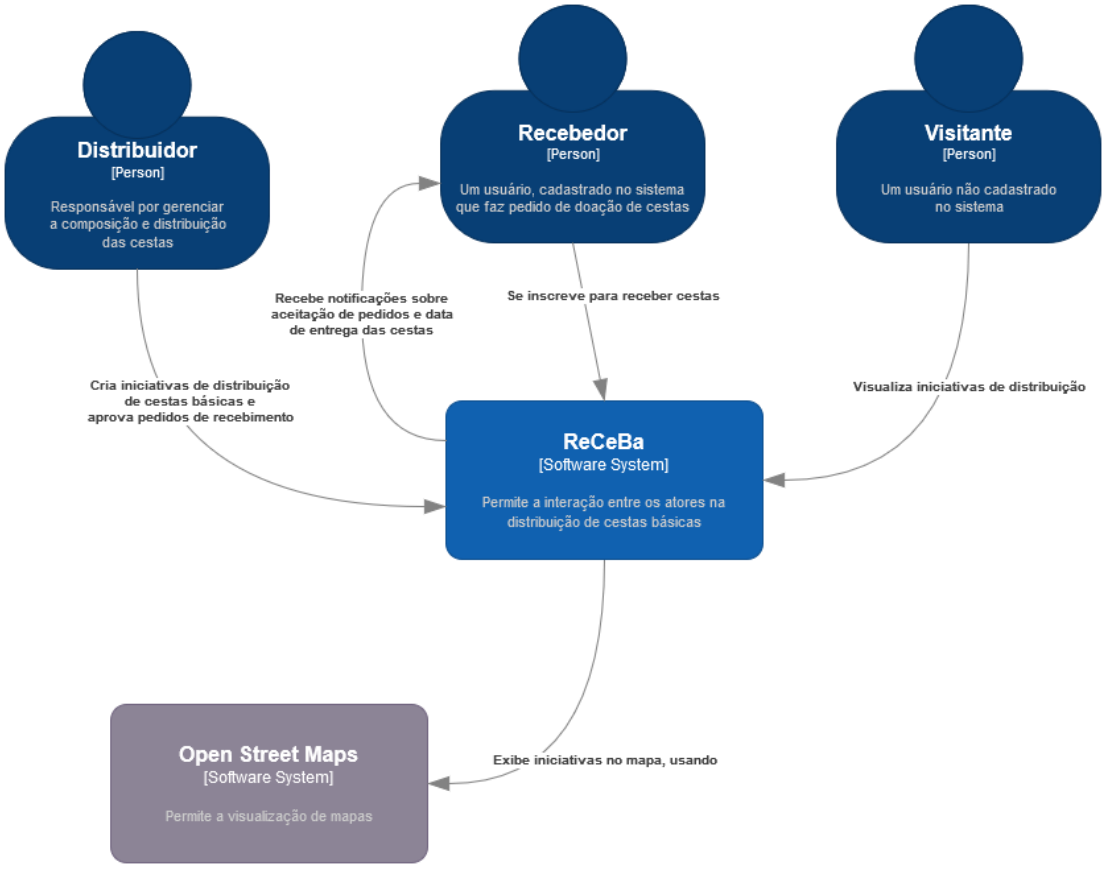

Figura 1 - Visão Geral da Solução

A figura 1 mostra o diagrama de contexto da solução proposta, com todos os usuários envolvidos e suas interações com o sistema. Nele é possível perceber os três perfis presentes no sistema: distribuidores, recebedores e visitantes, bem como suas atribuições. Também é possível notar que o ReCeBa atua como um centralizador destas interações.

Link para o vídeo de apresentação da Etapa 1:

https://youtu.be/LrCGrWfXn34

4.2 Diagrama de Container

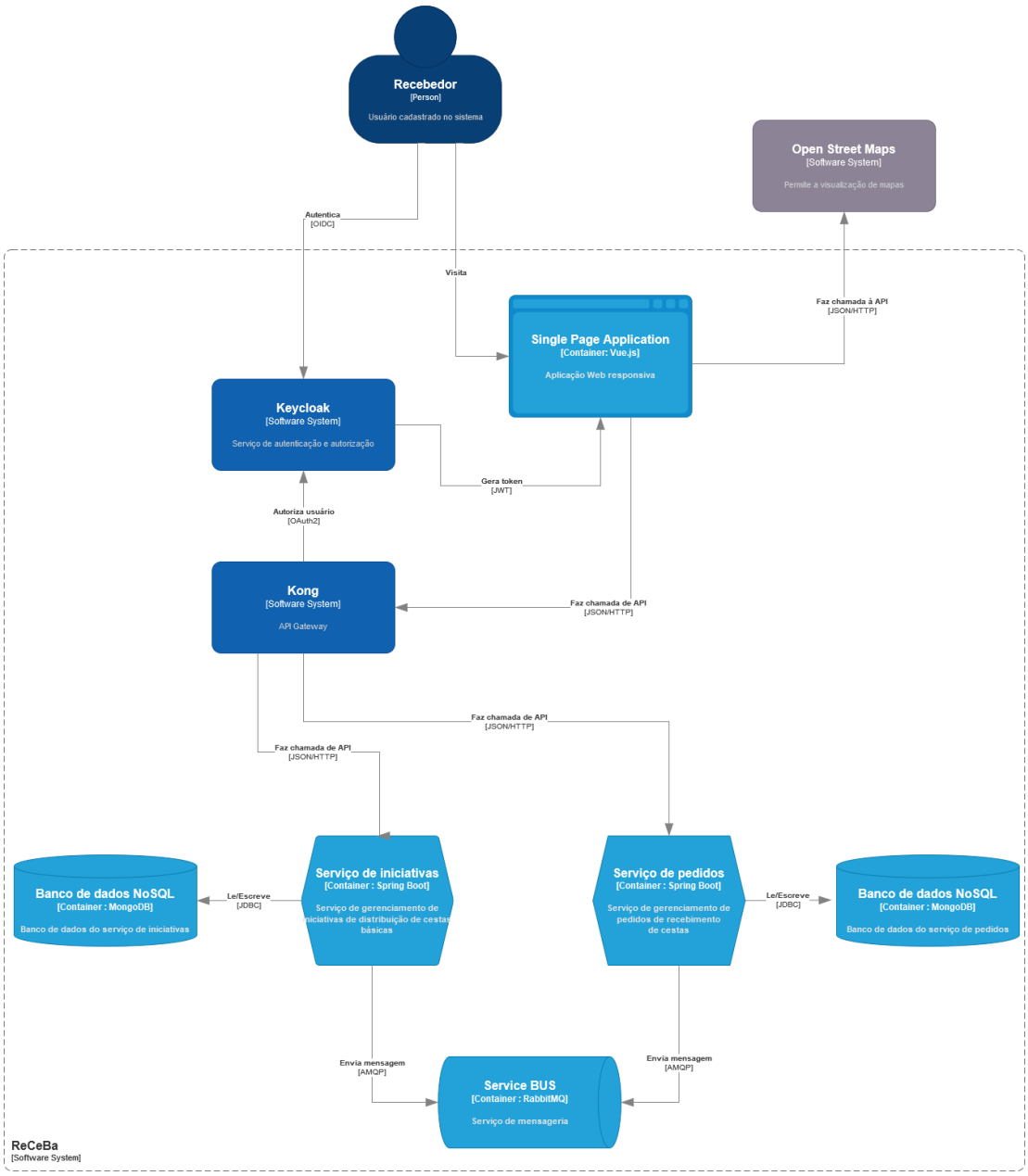

Figura 2 – Diagrama de container

A figura 2 apresenta os *containers* da aplicação e as conexões entre eles.
Foi adotado um padrão arquitetural de micro serviços visando maior flexibilidade no desenvolvimento das funcionalidades e a possibilidade de escalar apenas algumas partes mais utilizadas da aplicação.
Cada micro serviço possui seu próprio banco de dados, sendo que, para os serviços de iniciativas e de pedidos optou-se por utilizar bancos de dados não relacionais de documentos, pois estes se encaixavam melhor com a natureza não estruturada dos dados tratados nestes serviços.

A comunicação entre os serviços é feita através de um serviço de mensageria, utilizando filas de mensagem e padrão publicador-assinante, a fim de evitar acoplamento entre os componentes.
Para desacoplar o acesso do frontend aos serviços foi utilizado um API Gateway, responsável por rotear as requisições, além de garantir que apenas usuários autenticados e autorizados possam ter acesso aos recursos do backend
Para realizar a autenticação optou-se por uma ferramenta de Single Sign On (SSO) que gerencia a criação, manutenção e autenticação dos usuários.
Por fim a interface com o usuário se dá através de uma Single Page Application responsiva, que deve permitir o acesso tanto por aparelhos móveis quanto por desktop.

4.3 Diagrama de Componentes

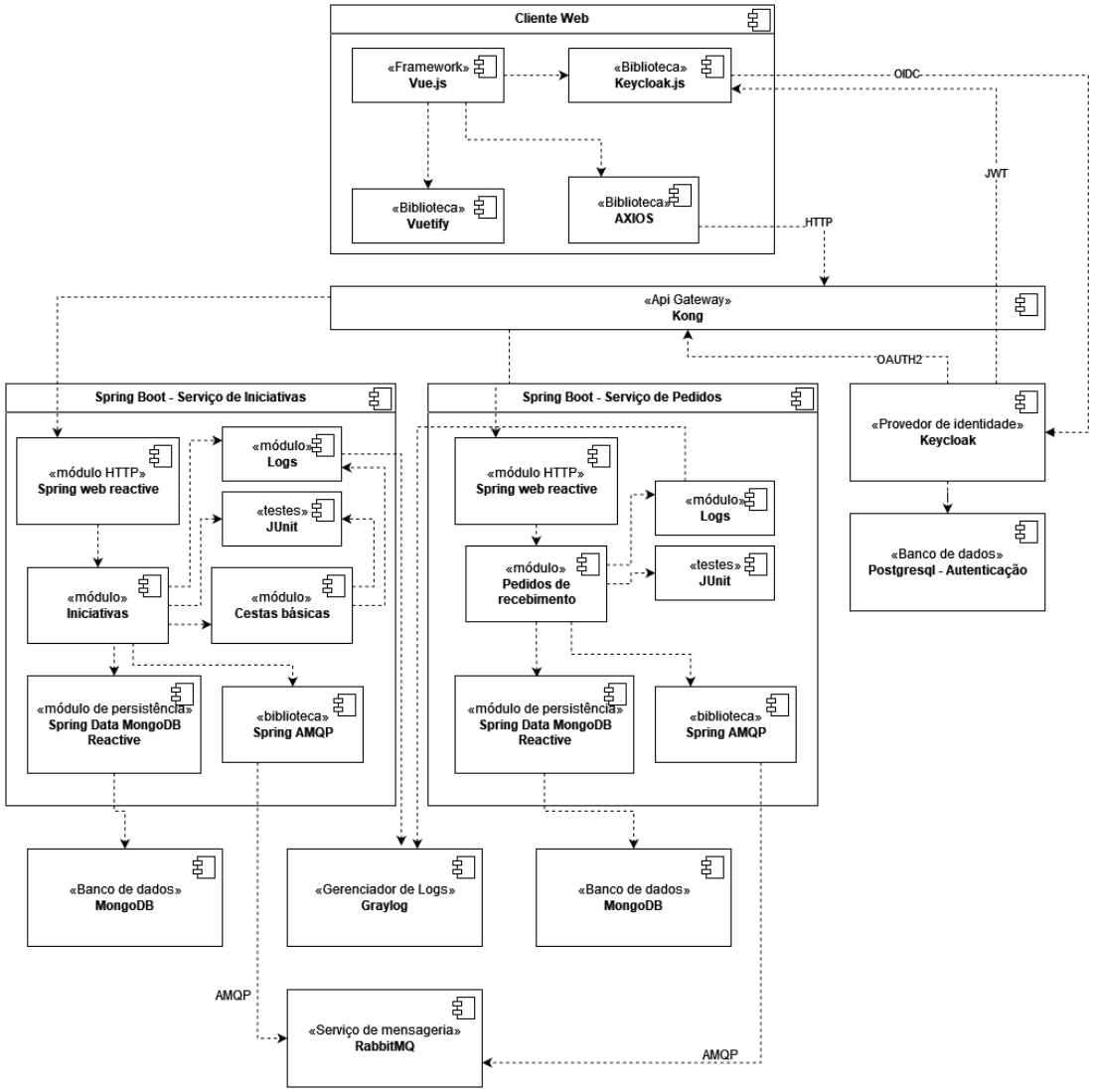

Figura 2 – Diagrama de Componentes

A figura 3 apresenta o diagrama de componentes da solução. Através dele, podemos ver que a solução é composta pelos seguintes componentes:

- **Cliente Web:** aplicação, rodando num navegador, responsável por exibir a interface de usuário, composta pelos subcomponentes:
    - **Vue.js:** framework de desenvolvimento utilizado na construção do Single Page Application (SPA);
    - **Vuetify:** biblioteca de componentes visuais para o Vue.js;
    - **Keycloak.js:** responsável pela interface do SPA com o servidor de autenticação Keycloak;
    - **AXIOS:** biblioteca responsável por criar, enviar e receber requisições HTTP;
- **Keycloak:** servidor de autenticação e autorização. Responsável pela criação, autenticação e autorização do usuário. Também é quem gera o token para ser utilizado nas chamadas ao API Gateway;
    - **PostgreSQL – Autenticação:** banco de dados utilizado pelo Keycloak;
- **Kong:** API Gateway responsável pelo controle e redirecionamento das requisições HTTP enviadas aos serviços;
- **Graylog:** servidor responsável pela coleta e disponibilização de logs para consulta;
- **RabbitMQ:** serviço de mensageria responsável pela comunicação entre os serviços;
- **Serviço de iniciativas:** responsável pelo gerenciamento das iniciativas de distribuição de cestas cadastradas pelos usuários. Nele ocorrem as ações de cadastro das iniciativas, bem como o cálculo de quantidade de cestas disponíveis. É composto pelos seguintes subcomponentes:
    - **Spring Boot:** plataforma Java para a construção e execução de micro serviços;
    - **Spring Web Reactive:** módulo do Spring responsável pelo recebimento e envio de chamadas HTTP;

  - **Spring Data MongoDB Reactive:** módulo do Spring responsável pela interface com o banco de dados MongoDB;
  - **Spring AMQP:** módulo do Spring responsável pelo envio e recebimento de mensagens AMQP;
  - **JUnit:** framework de testes unitários;
  - **Iniciativas:** módulo responsável pelo cadastro de iniciativas de distribuição de cestas básicas;
  - **Cestas básicas:** módulo responsável pelo cálculo de cestas disponíveis para distribuição e controle dos itens recebidos;
  - **Logs:** módulo responsável pelo envio de logs ao Graylog;
  - **MongoDB:** banco de dados não relacional utilizado nos micro serviços;

- **Serviço de pedidos:** responsável pelo gerenciamento dos pedidos de recebimento de cestas básicas efetuados pelos usuários. Além dos componentes externos já explicitados no serviço de iniciativas, é também composto por:
  - **Pedidos de recebimento:** módulo responsável pela vinculação dos pedidos às iniciativas de distribuição e também pela gestão de aprovações e fila de espera;

5. Prova de Conceito (PoC)

Nesta seção está detalhada a construção de uma prova de conceito que permitirá avaliar as funcionalidades relativamente aos requisitos arquiteturais definidos, segundo o modelo ATAM (seção 6).

Esta PoC implementa os requisitos funcionais RF01, RF02, RF03 e RF05.

É possível acessa-la através do link: http:// http://receba.s3-website-sa-east-1.amazonaws.com/

O sistema já possui um usuário cadastrado com dados de acesso:
Login: txaivg
Senha: 123456

5.1 Integrações entre Componentes

Para melhor visualização de como seria implementada a interface de usuário, foi desenvolvido um protótipo navegável que pode ser acessado pelos links abaixo:

Wireframe: https://www.figma.com/file/Uc1o1IxLpyMsBZWepsCfso/Receba?node-id=5%3A18

Protótipo: https://www.figma.com/proto/Uc1o1IxLpyMsBZWepsCfso/Receba?node-id=5%3A18&starting-point-node-id=5%3A18

5.2 Código da Aplicação

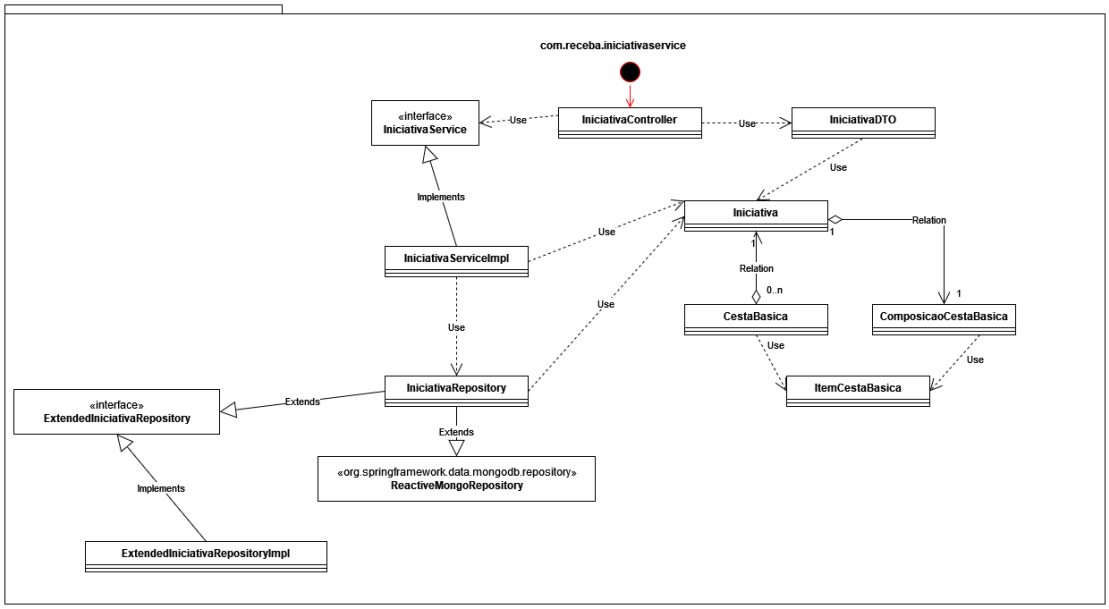

Figura 4 – Estrutura de código da aplicação

A estrutura da aplicação mostrada na Figura 4 apresenta os componentes do código do serviço de iniciativas, desenvolvido nesta prova de conceito:

- IniciativaController: esta classe é responsável por criar a API e gerenciar as requisições feitas ao serviço, bem como as respostas;

- IniciativaService: esta é a classe que implementa a lógica de negócios do serviço. Responsável por gerenciar o ciclo de vida tanto das iniciativas quanto das cestas básicas associadas;

- IniciativaRepository: esta classe é a responsável por fazer a conexão com o banco de dados e implementa os métodos necessários para armazenar e ler os dados de iniciativa;

- ExtendedIniciativaRepository: esta classe implementa queries que não são implementadas pro padrão por ReactiveMongoRepository

- Iniciativa, ComposicaoCestaBasica, CestaBasica e ItemCestaBasica: estas classes são os modelos de domínio utilizados para operações das regras de negócios;

- IniciativaDTO: é um data transfer object que representa o objeto a ser enviado como resposta às chamadas de API.

O código pode ser encontrado no repositório do GitHub: https://github.com/txai/receba

6. Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

6.1. Análise das abordagens arquiteturais

Atributos de Quailidade	**Cenários**	**Importância**	**Complexidade**
Confiabilidade	Cenário 1: O sistema deve estar disponível das 6h às 24h (18h) durante os sete dias da semana	A	B
Usabilidade	Cenário 2: A interface de usuário deve ser acessível Cenário 3: A interface de usuário deve ser responsiva	M	M
Eficiência	Cenário 4: O sistema deve atender até 2000 usuários simultaneamente sem degradação da performance Cenário 5: O tempo de resposta da interface de usuário deve ser de até 1 segundo 90% do tempo	B	A
Compatibilidade	Cenário 6: O sistema deve ser	A	M

	compatível com as versões mais recentes dos navegadores: Microsoft Edge, Mozilla Firefox e Google Chrome		
Segurança	Cenário 7: O sistema deve prover segurança dos dados pessoais identificáveis, permitindo que apenas os donos desses dados possam lê-los e alterá-los Cenário 8: Os dados pessoais identificáveis devem ser encriptados	A	A

6.2. Cenários

Cenário 1 - Disponibilidade: Ao acessar o sistema no horário das 6h às 24h em qualquer dia, aleatóriamente, deve ser possível utilizar suas funções principais: login, cadastro de iniciativas de distribuição de cestas e mapa de iniciativas cadastradas. Dentro desta janela, o sistema pode ficar indisponível por no máximo 60h no ano.

Cenário 2 - Usabilidade: A navegação pela interface gráfica do sistema deve ser possível através de teclado e leitor de telas. Com isso uma pessoa com baixa ou ausência total de visão deve conseguir realizar autenticação e cadastro.

Cenário 3 – Usabilidade: Ao navegar pela interface gráfica utilizando um navegador móvel, deve ser possível executar todas as principais funções do sistema.

Cenário 4 – Eficiência: Ao acessar o sistema com 2000 usuários simultâneos o tempo de resposta da operação de cadastro de iniciativas de distribuição deve ser igual ou inferior ao tempo normal de 1 segundo.

Cenário 5 – Eficiência: Ao acessar as telas de edição de perfil e cadastro de iniciativas de distribuição, o tempo de resposta deve ser igual ou inferior a 1 segundo.

Cenário 6 – Compatibilidade: Ao acessar a interface gráfica do sistema nos navegadores Microsoft Edge, Mozilla Firefox e Google Chrome não deve ser notado diferenças entre os elementos gráficos.

Cenário 7 – Segurança: Um usuário somente deve precisar de autenticação para acessar os dados do sistema, seja através da interface gráfica, seja através da API dos serviços que compõem o backend

Cenário 8 – Segurança: O banco de dados deve ser encriptado em repouso e os dados devem ser encriptados em trânsito

6.3. Evidências da Avaliação

Atributo de Qualidade:	Confiabilidade
Requisito de Qualidade:	O sistema deve estar disponível das 6h às 24h (18h) durante os sete dias da semana
Preocupação:	
Garantir que o sistema esteja disponível para o público durante o período em que se espera maior movimento	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Acesso ao sistema dentro da janela de disponibilidade	
Mecanismo:	
Hospedar a aplicação em infraestrutura de nuvem que garanta a disponibilidade e permita monitoramento dos servidores	
Medida de resposta:	
O monitoramento dos servidores deve indicar que eles estiveram indisponíveis apenas 60h no ano quando a falha ocorrer dentro da janela de disponibilidade. Ao realizar um acesso aleatório, dentro da janela de disponibilidade, o sistema deve estar disponível	
Considerações sobre a arquitetura:	
Riscos:	Lock In em relação ao serviço de nuvem utilizado
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

A validação deste cenário foi feita implementando uma PoC no serviço de nuvem da AWS, que garante disponibilidade de 99% para os servidores, e monitorando a

ocorrência de falhas ao longo de 1 mês. Não houve registros de falhas conforme mostra a figura abaixo:

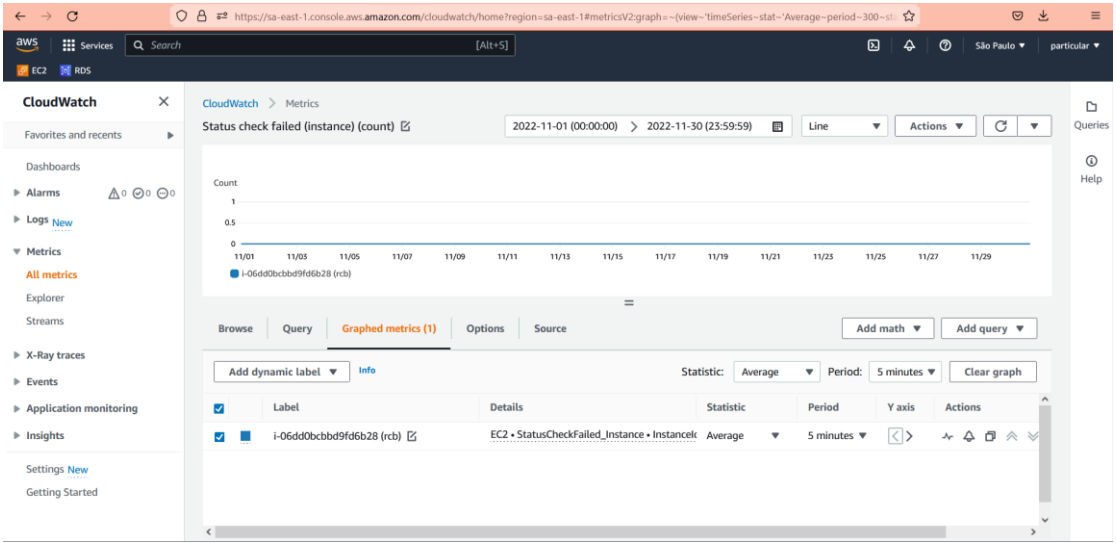

Figura 5 – Evidência de disponibilidade

Também foi realizado acesso em momento aleatório e comprovado que o sistema estava disponível.

Figura 6 – Evidência de disponibilidade

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A interface de usuário deve ser acessível, devendo ser atendidos os níveis A e AA definidos no WCAG 2.1
Preocupação:	
Garantir que pessoas com limitação visual possam utilizar o sistema normalmente	

Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Usuário com baixa visão, nenhuma visão ou com problemas visuais realicionados à distinção de cores, utilizando o sistema	
Mecanismo:	
Desenvolver a interface gráfica com elementos ARIA e prestar atenção ao esquema de cores para que tenham contraste suficiente	
Medida de resposta:	
Verificar que a navegação com o leitor de tela NVDA ocorre normalmente. Verificar que é possível distinguir os elementos graficamente mesmo que o usuário tenha problemas para distinguir as cores	
Considerações sobre a arquitetura:	
Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	Elementos gráficos presentes na interface tais como mapas comprometem a acessibilidade, havendo então a necessidade de implementar alternativas, como listas. Isto aumenta o custo de implementação

A validação deste cenário foi realizada utilizando o leitor de tela NVDA para acessar o aplicativo. Foi possível ouvir a descrição de todos os elementos relevantes. Infelizmente não é possível prover evidências gráficas para esta validação. Também foi validado a questão da disposição das cores. Através de ferramentas de simulação, foi possível simular a visão de um usuário com Acromatopsia (visão com ausência de cores). Neste cenário é possível navegar pelo sistema sem perda

significativa da usabilidade.

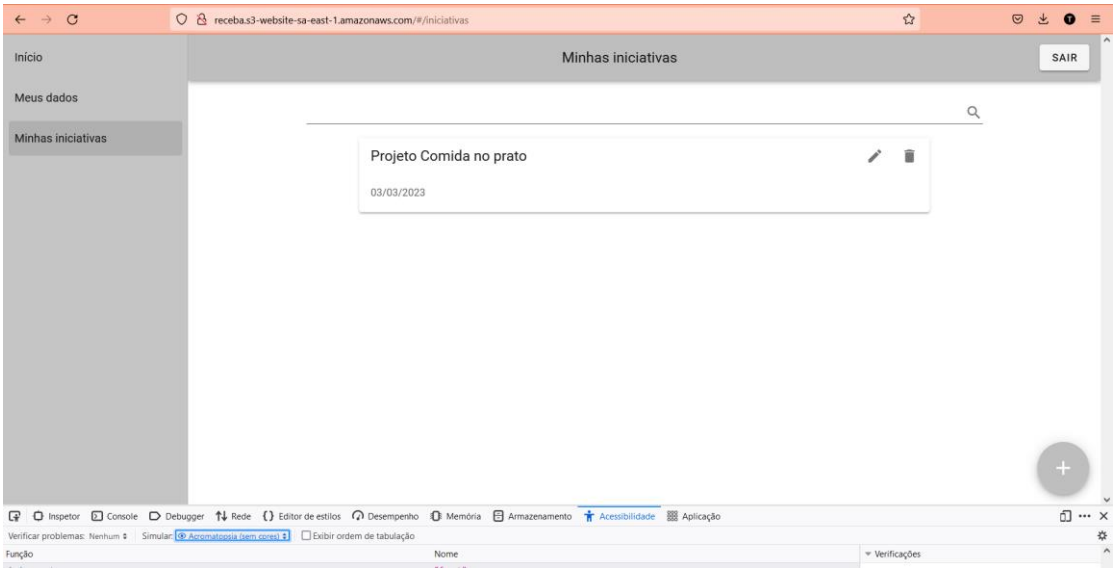

Figura 7 – Evidência de usabilidade

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	A interface de usuário deve possuir design responsivo
Preocupação:	
Garantir que usuários que disponham apenas de aparelhos móveis possam acessar e utilizar o sistema	
Cenário(s):	
Cenário 3	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Usuário acessando o sistema em um dispositivo móvel	
Mecanismo:	
Utilizar técnicas de responsividade no desenvolvimento da interface gráfica	
Medida de resposta:	
Verificar que os elementos gráficos são visíveis em diferentes tamanhos de tela	
Considerações sobre a arquitetura:	
Riscos:	Pode não ser possível colocar todas as funcionalidades em telas muito pequenas.

Pontos de Sensibilidade:	Não há
Tradeoff:	O desenvolvimento da interface gráfica responsiva envolve maior complexidade e tempo. A opção por fazer um app responsivo ao invés de app nativo representa ganho de tempo e custo ao projeto, mas implica perda de performance e de funcionalidades offline

A responsividade foi testada utilizando-se a opção do Mozilla Firefox de simular telas com diversos tamanhos de dispositivos móveis. Pode-se notar que é possível utilizar o sistema em sua totalidade através de um dispositivo móvel.

As figuras apresentadas correspondem a um teste com uma tela de 316 x 480

Figura 8 – Evidência de tela responsiva

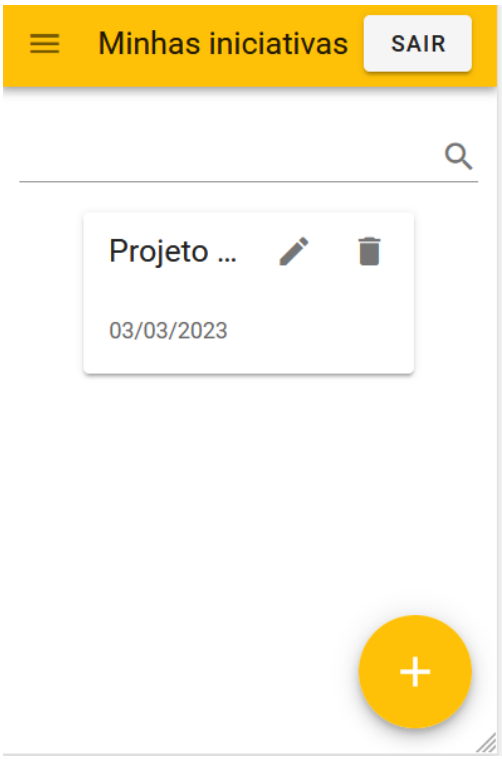

Figura 9 – Evidência de tela responsiva

Atributo de Qualidade:	Eficiência
Requisito de Qualidade:	O sistema deve atender até 2000 usuários simultaneamente sem degradação da performance
Preocupação:	
Possibilidade de degradação considerável da performance do sistema quando houver muitos acesso simultâneos, tais como numa situação de crise	
Cenário(s):	
Cenário 4	
Ambiente:	
Sistema com carga de utilização muito alta	
Estímulo:	
Acesso simultâneo de 2000 usuários, mantido por 1 minuto	
Mecanismo:	
Utilização do Kong atuando como API Gateway e também como Load Balancer	

Medida de resposta:	
O tempo das requisições e das interações de tela devem permanecer inferiores a 1s	
Considerações sobre a arquitetura:	
Riscos:	O Kong pode tornar-se um gargalo e também é um ponto único de falha
Pontos de Sensibilidade:	A API Gateway é um ponto sensível dependendo do número de requisições
Tradeoff:	O uso de um Load balancer aumenta a escalabilidade horizontal do sistema, mas reduz a performance em operações normais

O teste de validação foi realizado utilizando a ferramenta Loader. Com ela foram simulados 2000 clientes conectando ao sistema simultâneamente e foram medidos os tempos de resposta durante 1 minuto. O tempo médio ficou em 126ms e o tempo máximo foi de 1264ms. Na média o sistema comportou-se como esperado, mas é possível notar leve degração no início do teste quando todos os clientes estão em processo de conexão. Como não foi implementado load balancer o tempo ficou aquém do esperado.

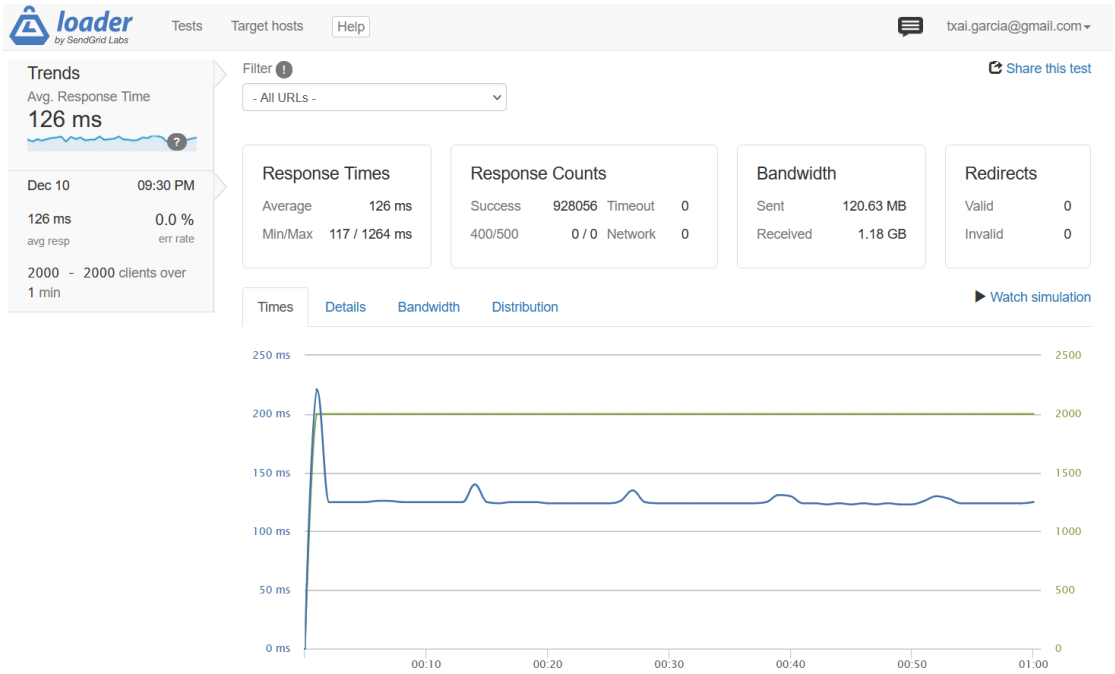

Figura 10 – Evidência do teste de carga

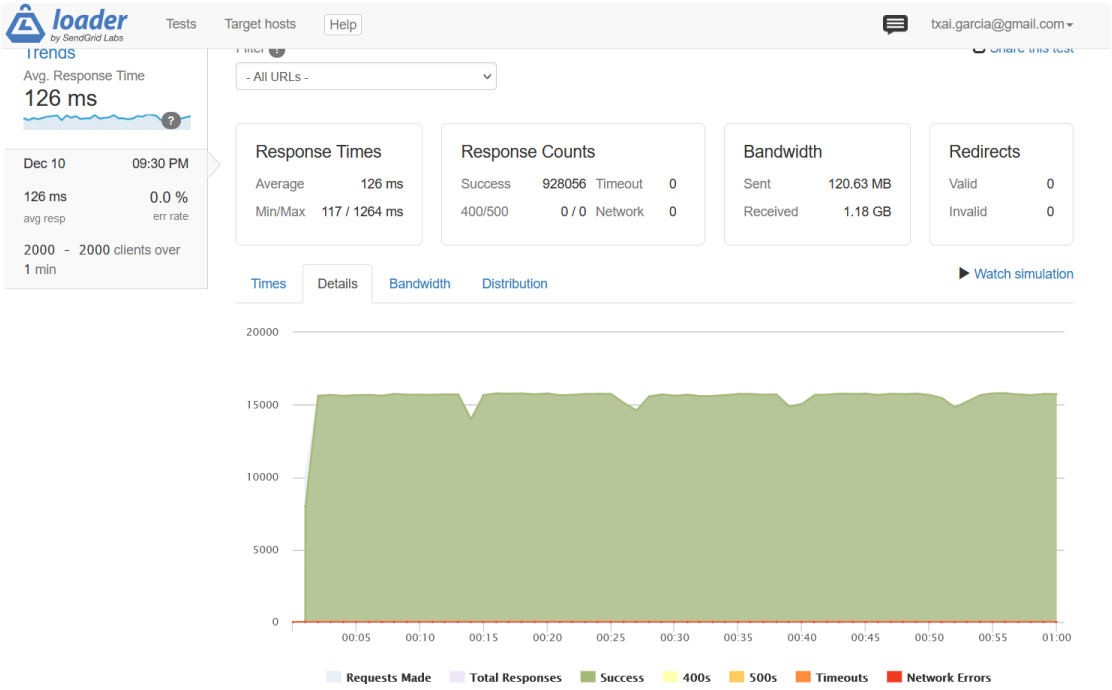

Figura 11 – Evidência do teste de carga

Atributo de Qualidade:	Eficiência
Requisito de Qualidade:	O tempo de resposta da interface de usuário deve ser de até 1 segundo 90% do tempo
Preocupação:	
Se o tempo de resposta da aplicação for muito lento, isto pode interferir na satisfação do usuário	
Cenário(s):	
Cenário 5	
Ambiente:	
Sistema com carga de utilização muito alta	
Estímulo:	
Acesso simultâneo de 2000 usuários, mantido por 1 minuto. Acesso à funcionalidade de cadsatro de iniciativas	
Mecanismo:	

Utilização de SPA e métodos de cache	
Medida de resposta:	
O tempo de carregamento da tela de cadastro de iniciativas deve ser inferior a 1s. Esta é a tela mais pesada da aplicação no momento	
Considerações sobre a arquitetura:	
Riscos:	Instabilidades na rede podem tornar o acesso lento
Pontos de Sensibilidade:	O uso de cache e do SPA pode ser um problema se houver muitas atualizações nos cadastros
Tradeoff:	O uso de SPA impacta no SEO do site, além de ter um carregamento inicial talvez mais lento

Para validar este cenário, aproveitou-se o teste anterior e mediu-se o tempo de resposta da página mais lenta da aplicação durante o período de alta carga. O tempo de carregamento ficou em 473 ms, abaixo de 1s.

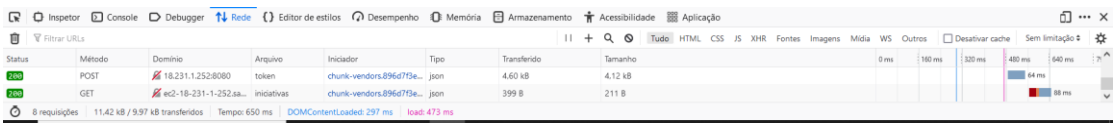

Figura 12 – Evidência do teste de eficiência

Atributo de Qualidade:	Compatibilidade
Requisito de Qualidade:	A interface web deve rodar nas versões mais recentes do Microsoft Edge, Mozilla Firefox e Google Chrome
Preocupação:	
Diferentes usuários devem poder acessar o sistema utilizando o navegador que normalmente eles utilizam	
Cenário(s):	
Cenário 6	
Ambiente:	
Sistema em operação normal	

Estímulo:	
Acesso oriundo de cada um dos diferentes navegadores	
Mecanismo:	
Adaptar a interface gráfica conforme a necessidade em cada navegador	
Medida de resposta:	
O layout e funcionalidades devem permanecer inalterados nos 3 navegadores	
Considerações sobre a arquitetura:	
Riscos:	Não há
Pontos de Sensibilidade:	Não há
Tradeoff:	Atender diversos navegadores pode aumentar o tempo de desenvolvimento e impedir o uso de alguns recursos

Para a realização deste teste foram feitos acessos em cada um dos navegadores e constatou-se que a interface e funcionalidades se mantém inalteradas.

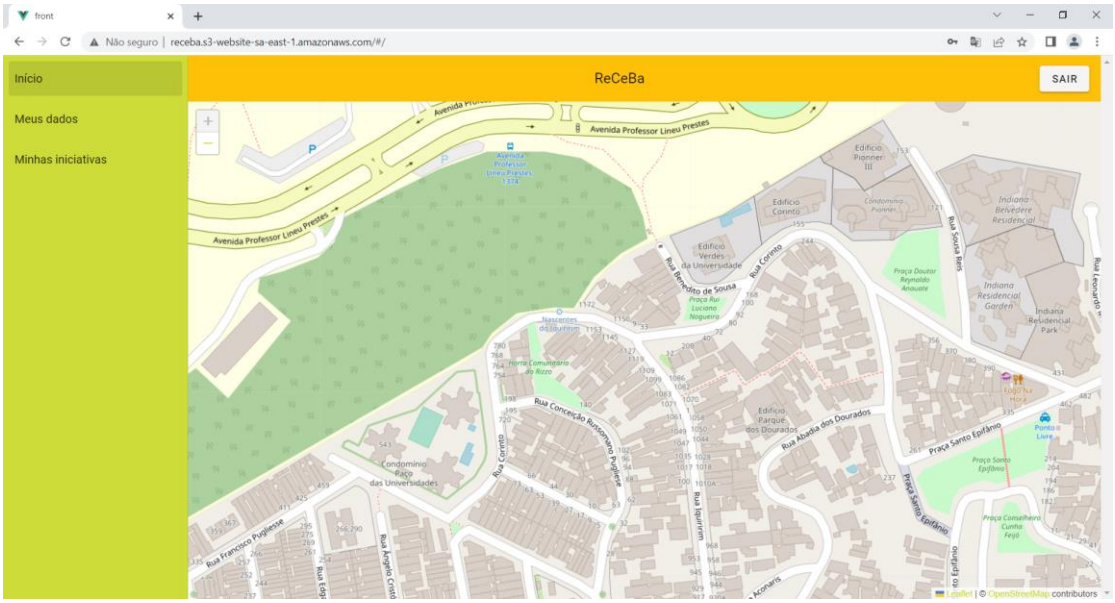

Figura 13 – Evidência de acesso à interface no Chrome

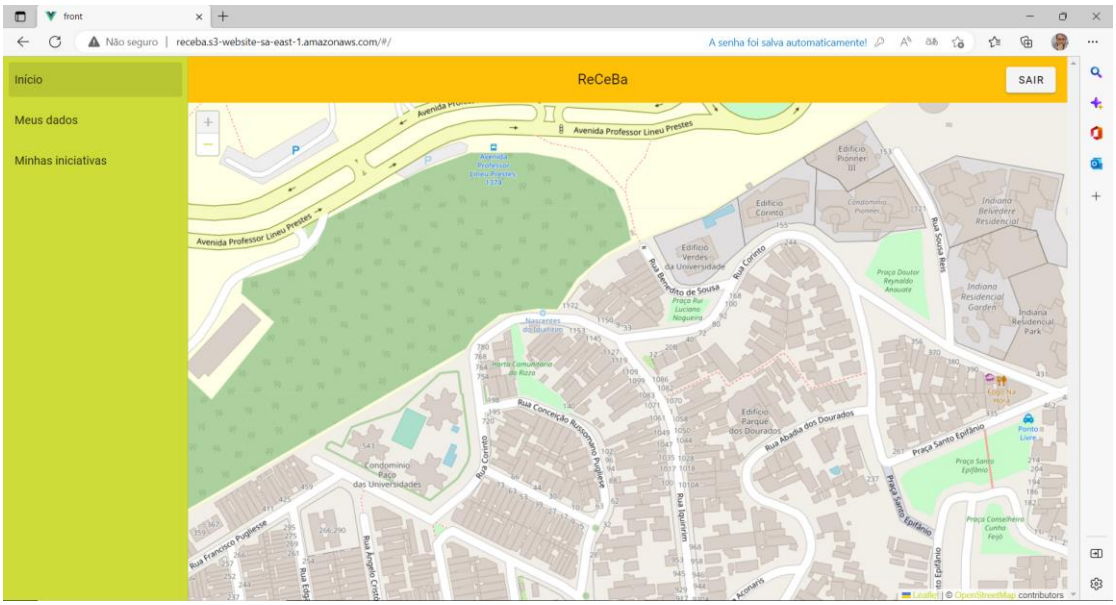

Figura 14 – Evidência de acesso à interface no Edge

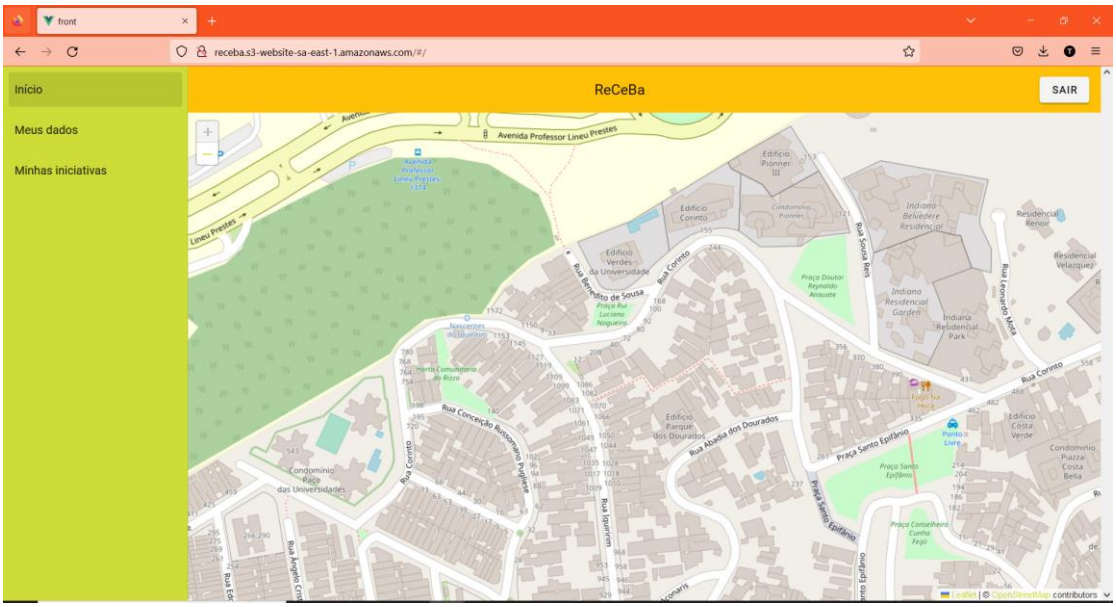

Figura 15 – Evidência de acesso à interface no Firefox

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	O sistema deve prover segurança dos dados pessoais identificáveis, permitindo que apenas os donos desses dados possam lê-los e alterá-los
Preocupação:	
O sistema trabalha com dados sensíveis que devem ser protegidos contra ataques ou	

acessos indevídos	
Cenário(s):	
Cenário 7	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Usuário acessa área protegida do sistema que contém dados particulares	
Mecanismo:	
Implementar servidor de autenticação e autorização e mecanismos de segurança que impeçam que usuários não autorizados tenham acesso a dados privados de terceiros	
Medida de resposta:	
Só deve ser possível acessar seus dados mediante acesso por login e senha	
Considerações sobre a arquitetura:	
Riscos:	Um ataque ao servidor de autenticação pode comprometer a segurança do sistema. Usuários podem acabar compartilhando senhas e expondo dados pessoais
Pontos de Sensibilidade:	O Keycloak, servidor de autenticação, pode ser um ponto único de falha e um gargalo dependendo do pico de utilização
Tradeoff:	A necessiade de autenticação impacta diretamente na performance do sistema de maneira negativa. Também é uma camada a mais de complexidade para o desenvolvimento

Na PoC foi implementado um servidor de autenticação que impede que usuários não autenticados tenham acesso às telas de cadastro de iniciativas e outros dados de usuários. Não foi implementado o bloqueio de API, sendo esse cenário parcialmente atendido.

Nas figuras abaixo é possível notar que o usuário só consegue acesso ao menu se efetuar login primeiro

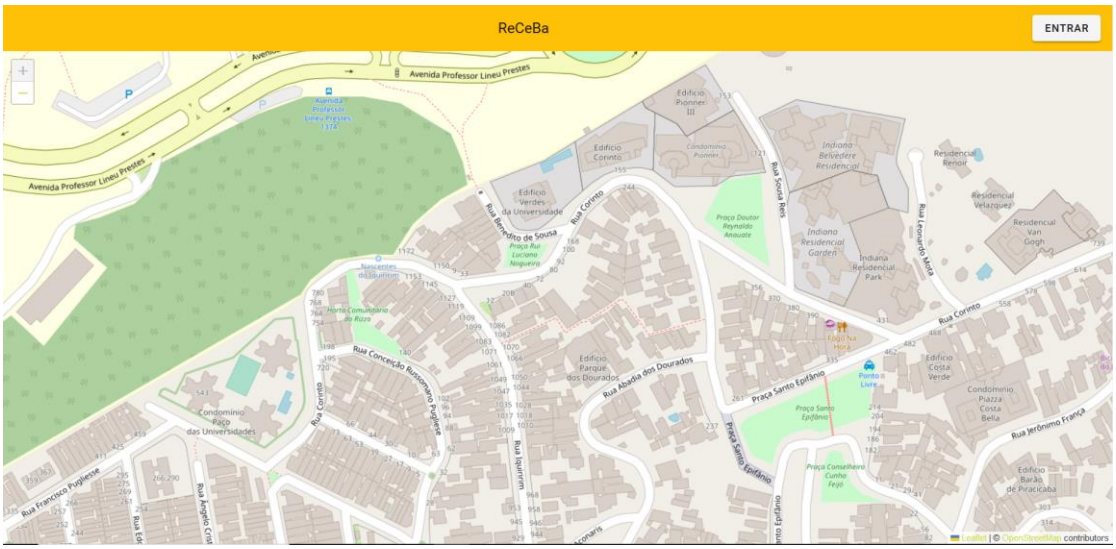

Figura 16 – Evidência de autenticação

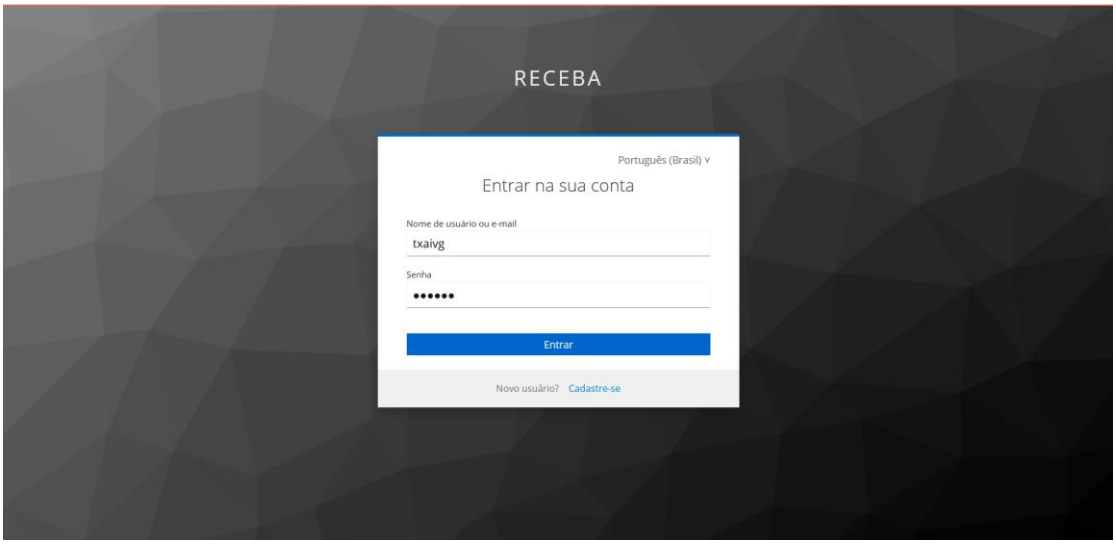

Figura 17 – Evidência de autenticação

Figura 18 – Evidência de autenticação

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	100% dos dados pessoais identificáveis devem ser encriptados em repouso e em trânsito
Preocupação:	
O sistema trabalha com dados sensíveis que devem ser protegidos contra ataques ou acessos indevídos	
Cenário(s):	
Cenário 8	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Invasor tenta capturar os dados em trânsito ou obtém acesso ao banco de dados	
Mecanismo:	
Implementar camada SSL para criptografar os dados em trânsito e hospedar o banco de dados em serviço de nuvem que ofereça criptografia	
Medida de resposta:	
Ao obter acesso aos dados não é possível lê-los pois estão criptografados	
Considerações sobre a arquitetura:	
Riscos:	Método de criptografia pode ser comprometido

Pontos de Sensibilidade:	Não há
Tradeoff:	A necessidade de criptografia possui impacto negativo na performance do sistema como um todo

Para a PoC não foi realizado criptografia em trânsito dos dados, pois isto demandaria investimento em certificados e um tempo maior de desenvolvimento. Porém, o banco de dados foi hospedado na infraestrutura de nuvem da AWS que provê a opção de encriptação do banco.

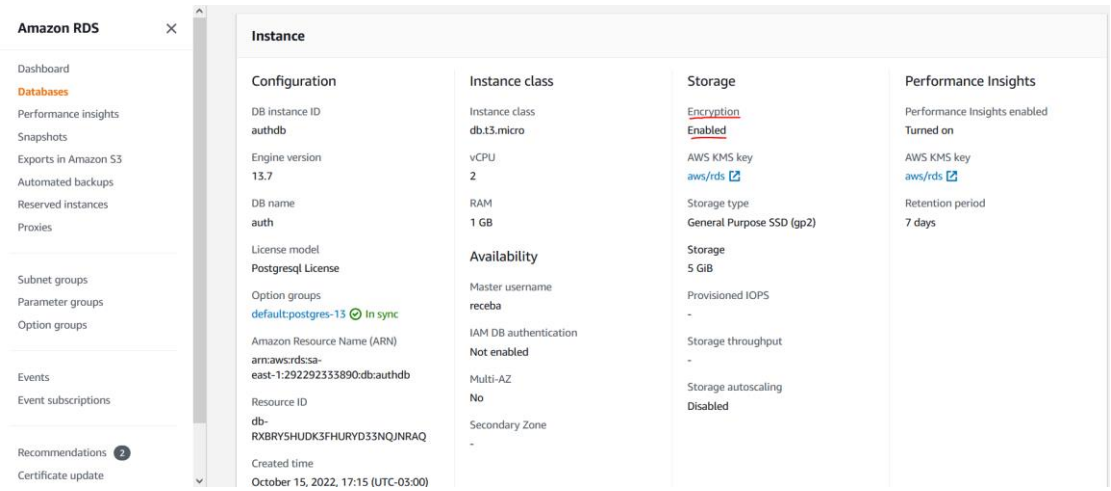

Figura 19 – Evidência de encriptação do banco

6.4. Resultados Obtidos

Requisitos Não Funcionais	Teste	Homologação
RNF01: O sistema deve estar disponível das 6h às 24h (18h) durante os sete dias da semana	OK	OK
RNF02: A interface de usuário deve ser acessível, devendo ser atendidos os níveis A e AA definidos no WCAG 2.1	OK	OK
RNF03: A interface de usuário deve possuir design responsivo	OK	OK
RNF04: O sistema deve atender até 2000 usuários simultaneamente sem degradação da performance	OK	NOK
RNF05: A interface web deve rodar nas versões mais recentes do Microsoft Edge, Mozilla Firefox e Google Chrome	OK	OK
RNF06: O tempo de resposta da interface de usuário deve ser de até 1 segundo 90% do tempo	OK	OK
RNF07: O sistema deve prover segurança dos dados pessoais identificáveis, permitindo que apenas os donos desses dados possam lê-los e alterá-los	OK	NOK
RNF08: 100% dos dados pessoais identificáveis devem ser encriptados em repouso e em trânsito	OK	NOK

7. Avaliação Crítica dos Resultados

Quando analisados todos os atributos arquiteturais deste projeto, dois se destacam como atributos de maior relevância: segurança e usabilidade. Considerando estes critérios, e analisando a realidade orçamentária do projeto, entende-se que a arquitetura proposta atende, de maneira satisfatória, aos requisitos levantados.

A opção por construir uma interface gráfica desacoplada do servidor, permitiu o desenvolvimento independente da interface e maior agilidade na prototipação. Esta agilidade deveu-se sobretudo ao uso do framework Vue.js e sua biblioteca gráfica, Vuetify, pois pode-se aproveitar uma miríade de componentes já prontos para a rápida validação dos conceitos principais do sistema. Além disso, desacoplar a interface do servidor permitirá que outras opções de interface, tais como aplicativos nativos, sejam implementadas no futuro.

Apesar disso, esta decisão também acarretou num aumento considerável da complexidade do projeto, obrigando a equipe desenvolvedora a ter domínio de um número maior de tecnologias.

O uso de microserviços no backend não se mostrou uma decisão arquitetural interessante, por ter dificultado a integração entre os módulos que compõem o sistema, além de introduzir maior possibilidade de falhas. Dado o nível de acoplamento entre os módulos e baixas chances de expansão dos requisitos do sistema, uma nova abordagem arquitetural deve ser pensada.

O Keycloak como servidor de autenticação permitiu implementar uma importantíssima camada de segurança de maneira fácil e rápida, abstraindo toda a parte de autenticação para uma ferramenta já pronta. Além disso, por ser uma ferramenta open source, permite que não haja dependência em relação a nenhum serviço de nuvem específico, permitindo a migração, caso necessário.

Apesar de permitir certa escalabilidade, a performance não foi considerada como um atributo prioritário e, portanto, é um ponto fraco da arquitetura proposta. Notadamente, nota-se certa degradação devido à decisão de privilegiar a segurança do sistema como um todo, e à utilização de Java com Spring Boot no desenvolvimento do backend. Esta última decisão, apesar de impactar a performance, trouxe uma velocidade de desenvolvimento que impacta positivamente o projeto por trazer uma redução de custos de implementação.

8. Conclusão

Pode-se considerar que os objetivos deste trabalho foram cumpridos com êxito, sendo demonstrada a viabilidade de uma plataforma de integração entre os agentes presentes nas iniciativas de formação e distribuição de cestas básicas.

A criação da prova de conceito foi crucial para visualizar qual impacto uma priorização da segurança teria sobre o desempenho do sistema e permitir avaliar quais aspectos de um software seguro poderiam ser implementados sem que a usabilidade do sistema fosse severamente afetada. Foi possível aprender que não é possível construir um sistema que atenda a todos os aspectos desejados simultâneamente, devendo haver uma priorização de alguns aspectos em detrimento de outros.

Também foi possível perceber que aumentar muito a granularidade do sistema, no caso adotando um esquema arquitetural de microserviços, pode impactar negativamente a manutenibilidade, especialmente se os pontos de fronteira entre os serviços não estiverem tão claramente delimitados.

Neste caso, cabe uma reavaliação da arquitetura, estudando a possibilidade de implementar o backend como monólito. Ou ainda, estudar mais a fundo a definição de quais serviços devem compor a arquitetura e quais devem ser as interações sobre eles.

Outro aspecto relevante da arquitetura proposta, foi a capacidade de poder replicar o ambiente de produção em uma máquina de desenvolvimento com recursos reduzidos. Isto deriva diretamente da opção em adotar ferramentas open source e que não fossem atreladas ao provedor de serviços de nuvem. A manutebilidade do sistema melhorou muito por ser possível testar conceitos e problemas ainda durante o desenvolvimento, sem a necessidade de provisionar infraestrutura para isto.

Com isso seria possível para uma equipe descentralizada levar adiante este projeto de modo que ele fosse desenvolvido com licença de código aberto.

Futuras expansões do projeto devem se atentar à definição da API, de modo a avaliar quais dados do usuário podem ser compartilhados e armazenados, visando atender os critérios de segurança. Além disso podem expandir a gama de interfaces gráficas disponívies construindo apps nativos para melhorar a performance e usabilidade em dispositivos móveis.

Referências

AGÊNCIA O GLOBO**. Brasil é nono país mais desigual do mundo, diz IBGE.** São Paulo, 12 de Novembro de 2020. Disponível em: https://exame.com/economia/brasil-e-nono-pais-mais-desigual-do-mundo-diz-ibge/. Acessado em: 08 de Maio de 2022.

UOL. **Desemprego no Brasil cai a 13,2%, mas renda tem a maior queda da história.** São Paulo, 27 de Outubro de 2021. Disponível em: https://economia.uol.com.br/noticias/redacao/2021/10/27/pnad-ibge-desemprego-brasil.htm. Acessado em: 08 de Maio de 2022.

PELLEGRINI, A. **Qual o quadro de insegurança alimentar no Brasil da pandemia.** São Paulo, 13 de Abril de 2021. Disponível em: https://www.nexojornal.com.br/expresso/2021/04/13/Qual-o-quadro-de-inseguran%C3%A7a-alimentar-no-Brasil-da-pandemia. Acessado em: 08 de Maio de 2022.

BROWN, S. **O modelo C4 de documentação para Arquitetura de Software**. 01 de Agosto de 2018. Disponível em: https://www.infoq.com/br/articles/C4-architecture-model/. Acessado em 01 de Junho de 2022.