PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

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

Projeto Integrado

Relatório Técnico

SISTEMA DE PROSPECÇÃO DE LEADS DE SEGUROS

Gabriel de Souza Martins

Belo Horizonte
Agosto de 2023

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

Introdução

Ao longo da história pessoas e empresas sempre estiveram sujeitas a diversos tipos de eventos imprevisíveis, ou seja, a diversos tipos de riscos como por exemplo morte, incêndio, acidente de automóveis e outros. Portanto o seguro pode ajudar pessoas ou empresas a diminuírem as preocupações futuras que poderão gerar prejuízo (ESCOLA NACIONAL DE SEGUROS, 2019).

Segundo a Escola Nacional de Seguros (2019), seguro é um mecanismo de transferência de risco de uma pessoa ou empresa para uma seguradora que assumirá este risco, portanto ao transferir o risco, a pessoa ou empresa irá pagar determinado valor à seguradora e caso este risco aconteça, a seguradora reembolsará as perdas sofridas.

O setor de seguros, segundo a FENACOR (2022), teve arrecadação de R$ 321,05 bilhões no acumulado até novembro de 2022, que representa um crescimento de 16,6% em relação ao mesmo período em 2021, isso evidência sua capacidade de crescimento ano após ano e estima-se que essa tendência deverá se manter fazendo com que o setor mantenha uma participação importante no PIB brasileiro.

Atualmente, a empresa moderna e inteligente sabe que ouvir o cliente e solucionar seus anseios e problemas é a única forma de continuar existindo e para vendas de seguros isto não é diferente ou menos importante, pois a compreensão das tendências atuais ajudará em vários aspectos, sendo assim é importante saber que a nova ordem em vendas de seguros é baseada nos seguintes fatores: clientes mais exigentes e concorrência mais acirrada (SANTOS, 2018).

Com o avanço das tecnologias e o alcance do acesso a à internet, o comércio eletrônico ocupa cada vez mais espaço no mercado e por sua vez as lojas virtuais têm grandes desafios como prover a melhor experiência no processo de compra, agilizar entregas, fidelizar clientes e tornar os preços competitivos.

De acordo com SEBRAE (2022), pesquisas realizadas pelo E-commerce Radar mostraram que 82% dos clientes abrem mão da compra antes de realizarem o seu pagamento, ou seja, desistem de comprar e entre os principais motivos de desistência estão: indecisão, lentidão de internet, preço, frete, condições de pagamento e entrega.

Segundo Lippert (2021), a compra de um usuário no ambiente digital, geralmente, é divida em três fases: lead, oportunidade e venda. No caso de um e-commerce, o lead seria o carrinho, a oportunidade seria o pedido e a venda a fatura, porém 50% dos usuários que colocam os pedidos no carrinho não concluem a venda e da mesma forma, há também usuários que efetuam o pedido, geram boleto de pagamento, mas não o pagam.

Considerando este cenário de forma geral, este trabalho tem por objetivo propor uma solução de arquitetura de software que visa otimizar as vendas de seguros por meio da prospecção de leads durante a desistência por parte do cliente.

O cliente ao iniciar uma simulação de seguro, preencher seus dados pessoais e não finalizar a compra irá gerar um lead para que este, por sua vez, possa ser processado e gerar uma oferta mais atrativa ao cliente informando-o da nova oportunidade.

Como principais benefícios desta solução destacam-se: disponibilidade, escalabilidade, performance, melhoria na experiência do usuário, maximização de lucro e coleta de dados para estudos futuros.

A figura 1 exemplifica por meio de desenho livre a solução proposta.

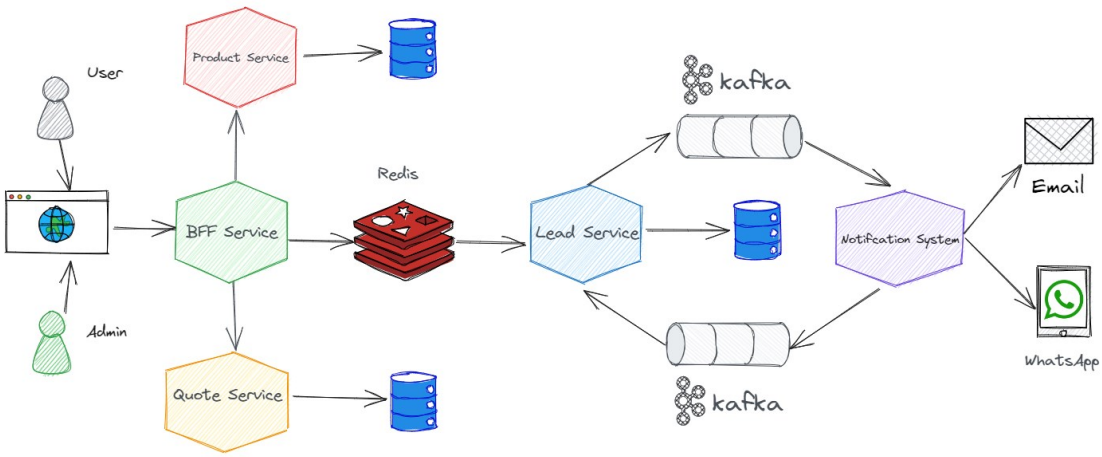

Figura 1- Exemplo da solução Proposta. Fonte: do Autor

O objetivo geral deste trabalho é apresentar a descrição do projeto arquitetural de uma plataforma de prospecção de leads de vendas de seguros. O projeto visa fornecer uma plataforma na qual vendas de seguros não concluídas se transformem em dados que representem vendas em potencial e notifiquem o cliente para concluir a mesma. A plataforma deverá possuir alta disponibilidade, segurança, performance e poderá ser acessada por diversos dispositivos diferentes com acesso à internet através de um browser (desktops, notebooks, tablets e smartphones).

Os objetivos específicos propostos são:

- Criar um módulo responsável por simular vendas de seguros que permita o cliente informar dados pessoais e escolher o produto desejado.
- Criar mecanismos de integração que permitam a captura dos leads durante vendas não concluídas.
- Notificar o cliente da venda não concluída com base no lead capturado.

2. Especificação Arquitetural da solução

Esta seção apresenta a especificação arquitetural da solução proposta a ser desenvolvida, incluindo diagramas, restrições e requisitos funcionais e não-funcionais que permitam visualizar a macroarquitetura.

2.1 Restrições Arquiteturais

Nesta seção são definidos os requisitos arquiteturais da solução proposta levando em consta restrições tecnológicas que precisam ser satisfeitas.

R1: O frontend do software deve ser desenvolvido em linguagem JavaScript, com o *framework* Angular.
R2: O backend do software deve ser desenvolvido em linguagem Java, com o *framework* SpringBoot.
R3: As APIs devem seguir o padrão RESTful.
R4: A captura de leads deve ocorrer primeiramente em um banco de dados em memória para fins de performance.
R5: O processamento dos leads deve ocorrer de forma assíncrona.
R6: O sistema deve ser hospedado em algum provedor de Nuvem.
R7: O backend deve gerar logs estruturados que permitam pesquisa e indexação.

2.2 Requisitos Funcionais

Esta seção tem por objetivo detalhes os requisitos funcionais da solução.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	Efetuar login	M	A
RF02	Manter produtos de seguros	M	A
RF03	O sistema deve permitir a simulação de seguro com base nos dados de CPF, Nome, Sexo e Data de Nascimento	A	A
RF04	O sistema deve validar o CPF informado	M	A
RF05	O sistema deve receber e validar o e-mail informado	M	A
RF06	O sistema deve notificar clientes que não concluírem a simulação com ofertas mais atrativas	A	A

*B=Baixa, M=Média, A=Alta.

2.3 Requisitos Não-funcionais

Esta seção tem por objetivo detalhes os requisitos não-funcionais da solução.

ID	Descrição	Prioridade B/M/A
RNF01	O sistema deve ser apresentar disponibilidade 24 X 7 X 365	A
RNF02	O sistema deve ser escalável	M
RNF03	O tempo de resposta de cada requisição deve ser menor que 500ms	M
RNF04	O sistema deve permitir uma boa manutenção e evolução	M
RNF05	O sistema deve apresentar fácil utilização	M
RNF05	O sistema deve ter comunicação com seus módulos independentemente das tecnologias que cada módulo utilize.	A
RNF06	Não permitir que o cadastro de produtos possa ser feito por usuários não autenticados	A

2.4 Mecanismos Arquiteturais

Nesta seção são apresentados os mecanismos arquiteturais da solução proposta.

Análise	*Design*	Implementação
Frontend	Interface de comunicação com o usuário	Angular (AWS S3)
Backend	Regras de negócio da aplicação	Java com SpringBoot (AWS ECS)
Persistência de Produtos	ORM	JPA + Hibernate (AWS RDS0
Persistência de Leads	ORM	MongoDB (AWS DocumentDB)
Integração com persistência de análise de leads	Publisher/Subscriber	Redis (AWS ElastiCache)
Integração com aplicação de análise de leads	Mensageria	Apache Kafka (AWS MSK)
Governança de APIs	Documentação e versionamento de APIs	OpenAPI 3.0 (AWS API Gateway)
Autenticação e Autorização	Verificação das credenciais para executar ações	OAuth0 (AWS Lambda)
Integração	Mensageria	Apache Kafka (AWS MSK)
Log do sistema	Framework de log	Logstash + Logback (AWS CloudWatch Logs)
Teste de Software	Testes automatizados	Cucumber + TestContainers (Github Actions)
CI/CD	Ferramenta para pipeline de	Github Actions

	integração e entrega contínua	
Deploy	Hospedagem dos sistemas	Terraform (Github Action)
Versionamento	Controle de código-fonte	Git e Github

3. *Modelagem Arquitetural*

Esta seção apresenta a modelagem arquitetural da solução proposta por meio de diagramas com o objetivo de permitir seu completo entendimento.

3.1 Diagrama de Contexto

A figura 2 apresenta o diagrama de contexto da solução proposta exemplificando a utilização do sistema por parte do cliente para simulação, a utilização do administrador para cadastro e manutenção de produtos e a captura e processamento de leads.

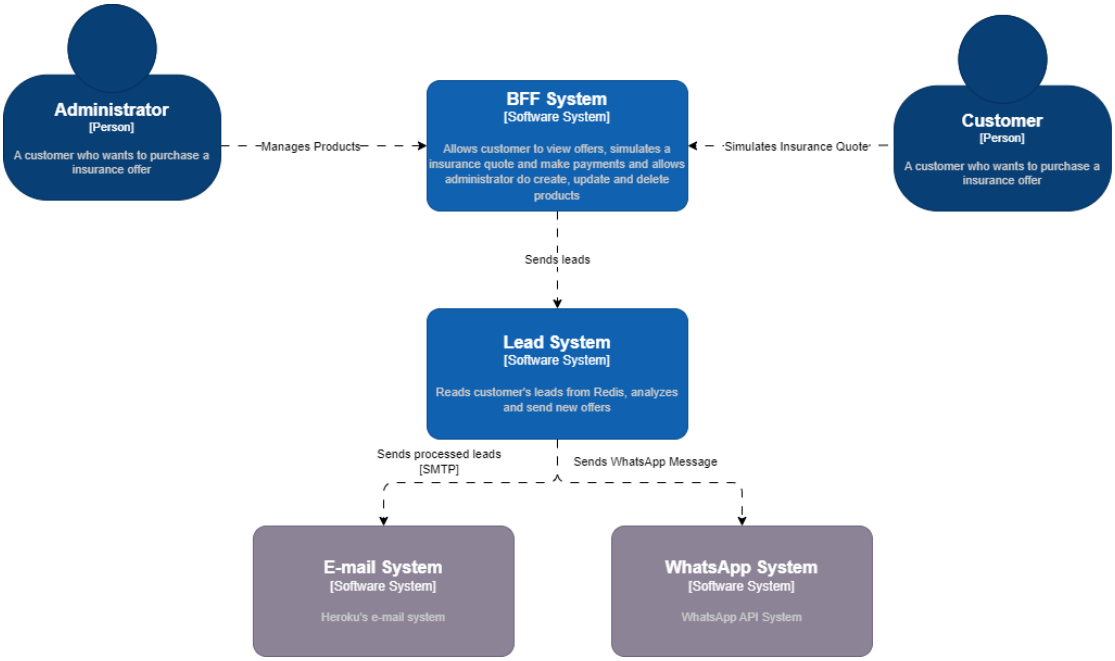

Figura 2 - Diagrama de Contexto da Solução. Fonte: do Autor

Como é possível observar, existem dois tipos de usuários: o cliente que irá realizar simulações/compra de seguros e o administrador responsável por cadastrar produtos.

3.2 Diagrama de Container

A figura 3 apresenta o Diagrama de Container da solução contendo mais detalhes da mesma.

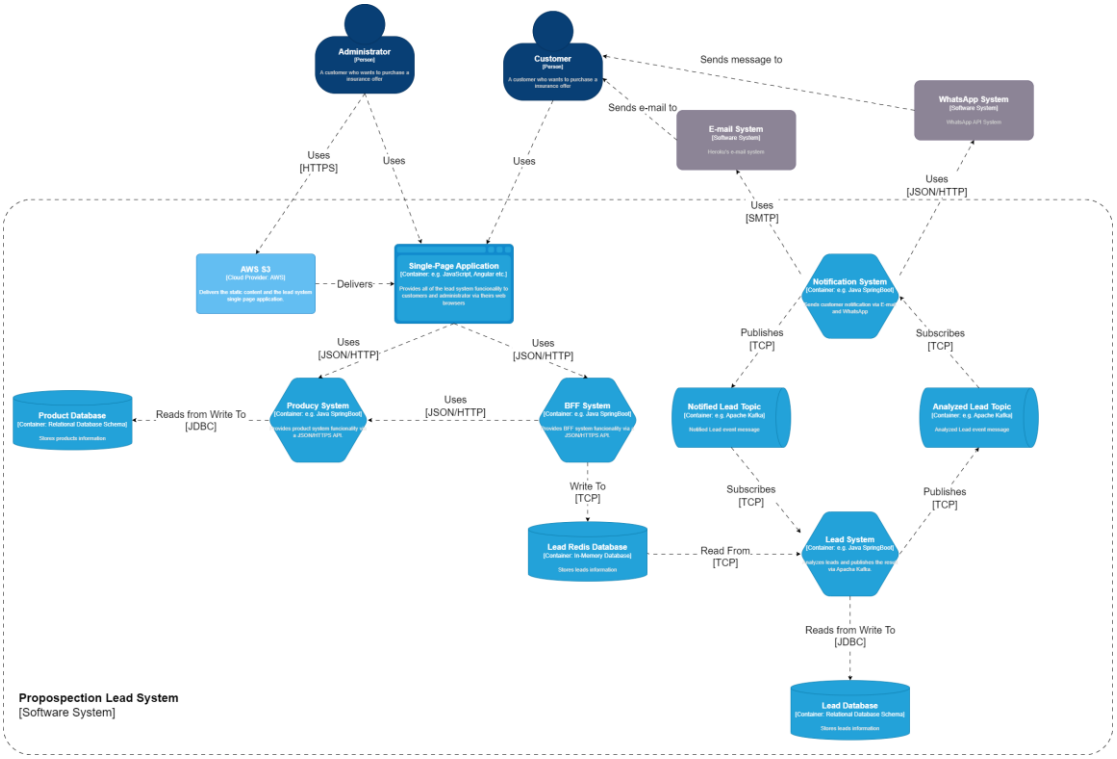

Figura 3 - Diagrama de Container da Solução. Fonte: do Autor

Nesta figura são representados os microsserviços que irão compor a solução, os bancos de dados que irão guardar informações de produto e lead e também a plataforma de mensageria responsável pela integração dos microsserviços.

É possível perceber a existência de dois tópicos: o tópico de lead analisado será utilizado pelo microsserviço de notificação para enviar e-mail e mensagem ao cliente que desistiu da compra e também o tópico de lead notificado que é usado para confirmação de que a notificação (e-mail/mensagem) foi entregue, seguindo assim o padrão SAGA.

3.3 Diagrama de Componentes

A figura 4 apresenta o Diagrama de Componentes da solução contendo mais detalhes.

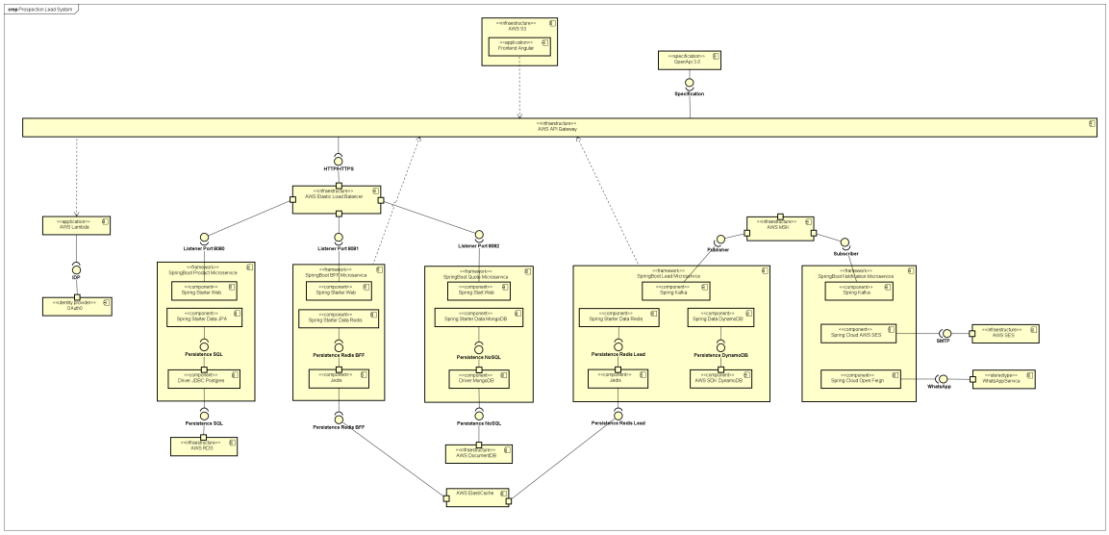

Figura 4 - Diagrama de Componentes da Solução. Fonte: do Autor

Conforme o diagrama apresentado na figura 4, os componentes participantes da solução são:

- Frontend Angular: Interface de comunicação com o usuário que permite que o mesmo crie produtos (quando administrador) e faça simulações de seguro (quando cliente).
- Product Microservice: Microsserviço responsável por prover as funcionalidades de produto
- BFF Microservice: Microsserviço responsável pelos dados que serão exibidos no frontend.
- Quote Microservice: Microsserviço responsável por gerar cotações de seguro.
- Lead Microservice: Microsserviço responsável pela análise dos leads capturados.
- Notification Microservice: Microsserviço responsável por comunicar o cliente via e-mail e mensagem sobre nova ofertas.
- AWS S3: Responsável por armazenar o conteúdo estágio do frontend.

- API Gateway: Responsável por centralizar as requisições HTTP e encaminhar para o backend correspondente
- AWS Lambda: Serviço AWS que será utilizado como autorizador pelo API Gateway.
- AWS Elastic Load Balancer: Responsável por balancear o trafego recebido pelo API Gateway e distribuir nos containers de aplicações.
- AWS SES: Serviço de e-mail da AWS.
- AWS MSK: Serviço da AWS com Apache Kafka gerenciado para comunicação entre microsserviços.
- AWS ElastiCache: Serviço da AWS responsável pelo Redis gerenciado.
- AWS DocumentDB: Serviço da AWS responsável pelo MongoDB gerenciado.
- AWS RDS: Serviço da AWS responsável pelo Postgres gerenciado.
- AWS DynamoDB: Serviço da AWS de banco de dados NoSQL auto gerenciado.
- WhatsApp Service: API do WhatsApp para envio de mensagens.
- OAuth0: Identity Provider que fará a gestão de acessos.
- Spring Data JPA: Módulo responsável pela implementação JPA de Persistência de Dados.
- Driver JDBC Postgre: Driver JDBC responsável pela comunicação com o Postgres.
- Spring Data MongoDB: Módulo responsável pela persistência dos dados no MongoDB.
- Driver MongoDB: Driver responsável pela comunicação com o MongoDB.
- Spring Data Redis: Módulo responsável pela persistência dos dados no ElastiCache Redis.
- Jedis: Java client responsável pela comunicação com o Redis.
- AWS SDK Dynamo: Software Development Kit para desenvolvimento de aplicações Java que utilizam o DynamoDB.

4. Avaliação da Arquitetura (ATAM)

Esta seção apresenta a avaliação da arquitetura do projeto desenvolvido segundo o método ATAM.

4.1 Análise das abordagens arquiteturais

A tabela abaixo traz um resumo das principais características da proposta arquitetural com base na seção 2.3 de requisitos não-funcionais.

Atributos de Quailidade	Cenários	Importância	Complexidade
Disponibilidade	Cenário1: O sistema deve ser apresentar disponibilidade 24 x 7 x 365	B	M
Escalabilidade	Cenário 2: O sistema deve ser escalável	B	M
Desempenho	Cenário 3: O tempo de resposta de cada requisição deve ser menor que 500ms.	A	A
Interoperabilidade	Cenário 4: O sistema deve ter comunicação com seus módulos independentemente das tecnologias que cada módulo utilize.	A	A
Usabilidade	Cenário 5: O sistema deve prover boa usabilidade.	M	M
Manutenibilidade	Cenário 6: O sistema deve permitir uma boa manutenção e evolução.	A	M
Segurança	Cenário 7: Não permitir que o cadastro de produtos possa ser feito por usuários não autenticados.	B	M

4.2 Cenários

Esta seção tem por objetivo detalhar os cenários utilizados na realização dos testes da aplicação.

Cenário 1 - Disponibilidade: Ao acessar as aplicações via requisições HTTP, as mesmas devem encontrar-se disponíveis para utilização de suas funcionalidades sem apresentar interrupções durante 24h por dia em 7 dias da semana nos 365 dias no ano.

Cenário 2 - Escalabilidade: As aplicações devem se adequar a carga de trabalho exigida, ou seja, devem apresentar a capacidade de suportar grande volume de requisições escalando horizontalmente os recursos necessários.

Cenário 3 - Desempenho: As aplicações devem apresentar bom desenho tendo o tempo de resposta menor que 500ms em cada requisição HTTP.

Cenário 4 - Interoperabilidade: Cada módulo da aplicação deve conseguir comunicar-se com os demais independente da linguagem utilizada em cada um deles, garantindo assim que a haja uma integração eficiente no sistema.

Cenário 5 - Usabilidade: Ao navegar na tela, o sistema deve apresentar boa usabilidade para o usuário. A navegação deve apresentar facilidade e o acesso as funcionalidades deve ser bem claro para a função que precisar ser realizada, o usuário deve ser capaz de criar produtos em no máximo 2 minutos e realizar cotações de seguro em no máximo 3 minutos.

Cenário 6 - Manutenibilidade: Havendo a demanda de novos recursos nas aplicações que possibilitem novas funcionalidades a arquitetura deve permitir que isso seja realizado de forma rápida e que não gere grandes impactos no código e funcionalidades existentes.

Cenário 7 - Segurança: Cada funcionalidade deve ser executada somente pelos usuários autenticados e autorizados a realizar determinada operação, como por exemplo inclusão, alteração, consulta e exclusão de produtos.

4.3 Evidências da Avaliação

O código-fonte das aplicações pode ser encontrado através do endereço eletrônico https://github.com/gabrielsmartins/prospect-leads-system.

Os resultados obtidos durante os testes das aplicações são apresentados abaixo, porém alguns dos cenários não foram implementados e, portanto, não foi possível avalia-los.

Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	O sistema deve ser apresentar disponibilidade 24 x 7 x 365.
Preocupação:	
O sistema deve ter alta disponibilidade para evitar prejuízos.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Utilização por parte do usuário.	
Mecanismo:	
Utilização de serviço de nuvem que permita alta disponibilidade	
Medida de resposta:	
SLA fornecido pelo serviço de nuvem para o recurso escolhido de hospedagem da aplicação.	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade no serviço de nuvem pode provocar indisponibilidade.
Pontos de Sensibilidade:	Não há.
Tradeoff:	Talvez 100% de disponibilidade não seja necessário e um valor um pouco menor atenda perfeitamente as necessidades de utilização.

Este atributo de qualidade não foi mensurado devido a sua complexidade de mediação na plataforma de nuvem Railway que foi utilizada para prova de conceito, porém foi implementado um Service Discovery que informa o status de cada aplicação conforme a figura 5.

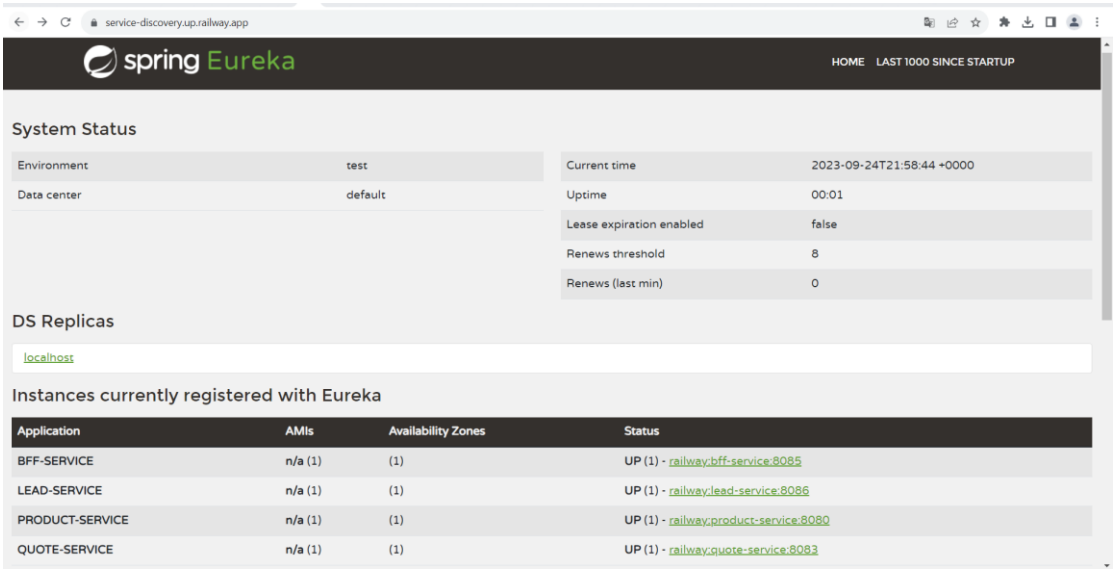

Figura 5 - Service Discovery Eureka. Fonte: do Autor

Atributo de Qualidade:	Escalabilidade
Requisito de Qualidade:	O sistema deve ser escalável.
Preocupação:	
O sistema deve ter a capacidade de atender altas demandas oferecendo o mesmo desempenho de forma contínua.	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Alta demanda de utilização por parte dos usuários.	
Mecanismo:	
Criar um grupo de auto scalling que permita o ajuste da aplicação para suportar altas demandas.	
Medida de resposta:	

Verificação da quantidade de containers em execução na plataforma de nuvem durante o período de alta demanda.	
Considerações sobre a arquitetura:	
Riscos:	Se a plataforma de nuvem não possuir o recurso de auto scalling pode inviabilizar este requisito.
Pontos de Sensibilidade:	Não há.
Tradeoff:	Não há.

Este atributo de qualidade não foi mensurado devido ao plano de utilização escolhido na plataforma de nuvem Railway que oferece somente recursos básicos.

A figura 6 ilustra as aplicações implantadas na plataforma Railway.

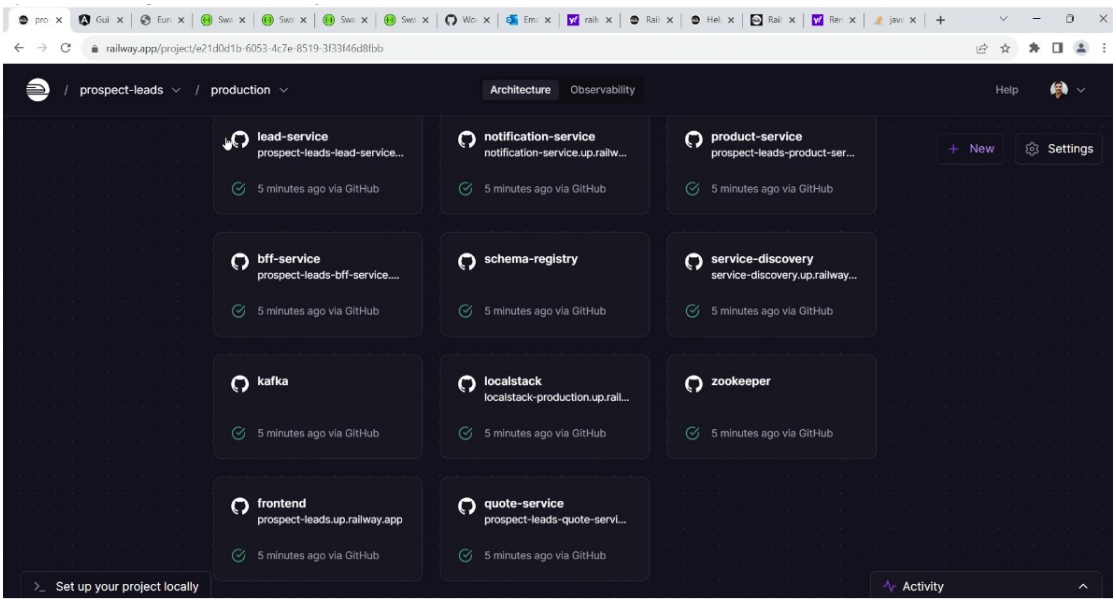

Figura 6 - Aplicações implantadas no Railway. Fonte: do Autor

Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O tempo de resposta de cada requisição deve ser menor que 500ms.
Preocupação:	
O sistema deve ter um bom tempo de resposta para não prejudicar a experiência do usuário.	
Cenário(s):	

Cenário 3	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Envio das requisições HTTP feitas no browser para o módulo REST do BFF.	
Mecanismo:	
Implementação do framework Spring Web Flux que permite o paradigma de programação reativa melhorando significante a performance da aplicação por utilizar fluxos não bloqueantes.	
Medida de resposta:	
Tempo de resposta em milissegundos que pode ser visualizado no browser na aba Network ou na ferramenta Insomnia utilizada para testes.	
Considerações sobre a arquitetura:	
Riscos:	Uma má utilização do Spring Web Flux pode comprometer o desempenho desejado.
Pontos de Sensibilidade:	Não há.
Tradeoff:	Não há.

As figuras 7, 8 e 9 exemplificam chamadas HTTP e o tempo de resposta com a ferramenta Insomnia.

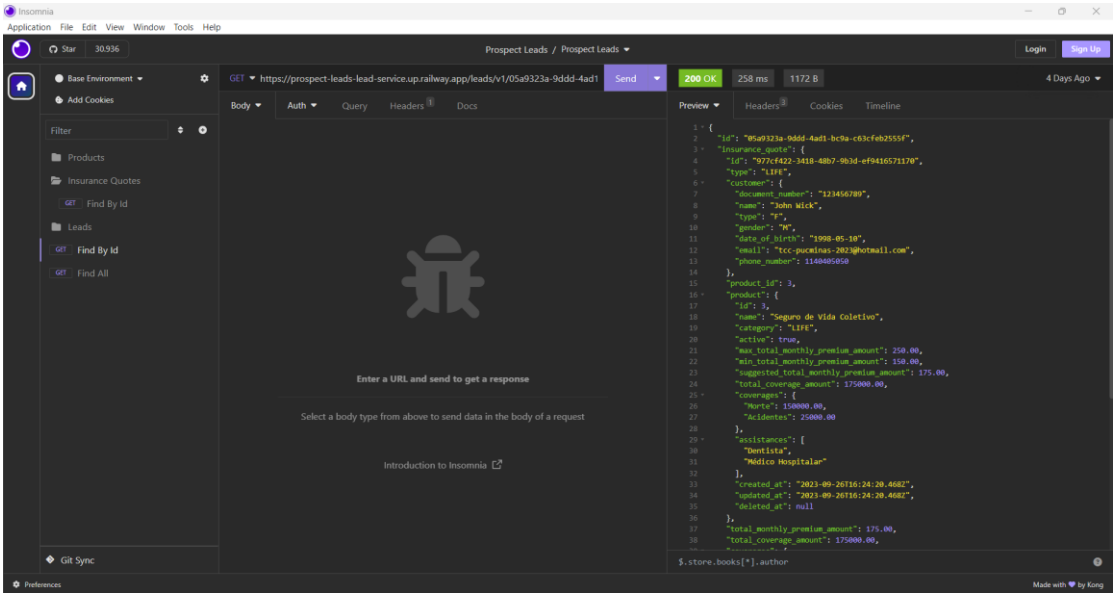

Figura 7 - Requisição HTTP para API de Leads. Fonte: do Autor

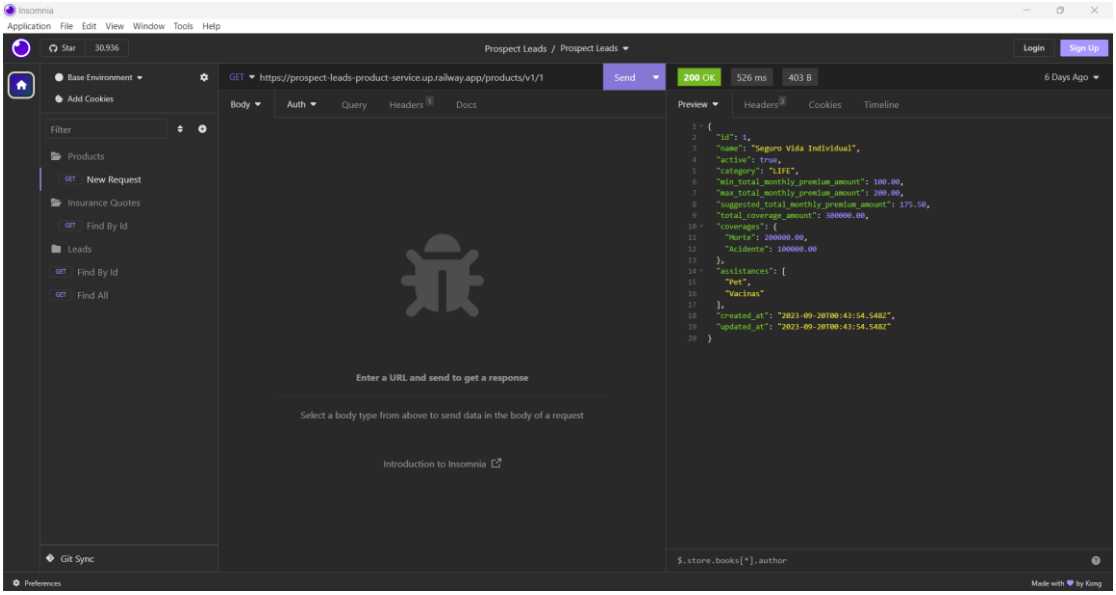

Figura 8 - Requisição HTTP para API de Produtos. Fonte: do Autor

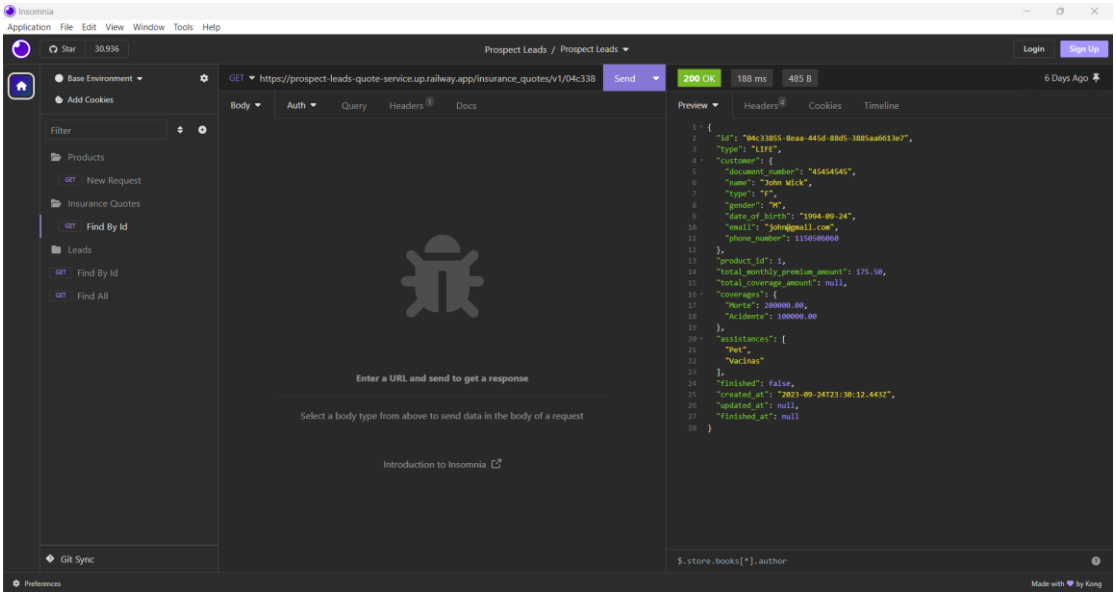

Figura 9 - Requisição HTTP para API de Cotações. Fonte: do Autor

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve ter comunicação com seus módulos independentemente das tecnologias que cada módulo utilize.
Preocupação:	
O sistema deve realizar integrações com os diferentes módulos independente da linguagem.	
Cenário(s):	
Cenário 4	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Requisições HTTP enviada para os módulos REST e envio de mensagens e streams nos middlewares Redis e Apache Kafka.	
Mecanismo:	
Utilização de APIs REST como interface de comunicação e middlewares de mensageira.	
Medida de resposta:	
Envio das requisições HTTP em formato JSON e mensagens em formato AVRO e JSON.	
Considerações sobre a arquitetura:	

Riscos:	Não há riscos mapeados.
Pontos de Sensibilidade:	Não há.
Tradeoff:	Não há.

As figuras 10, 11, 12 e 13 mostram os contratos de API REST e de mensageria.

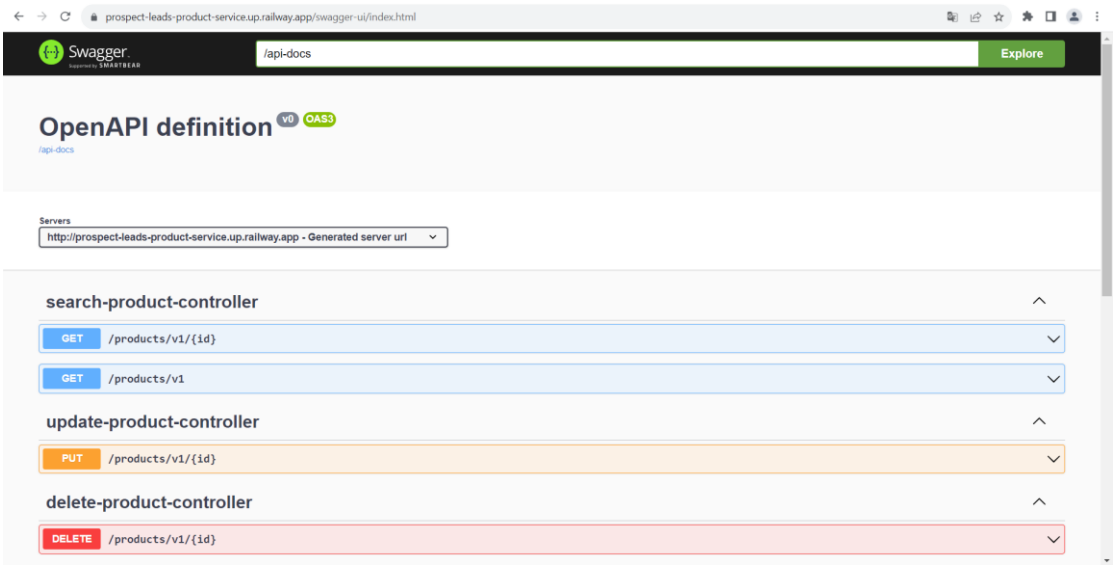

Figura 10 - Especificação OpenAPI da API de Produtos. Fonte: do Autor

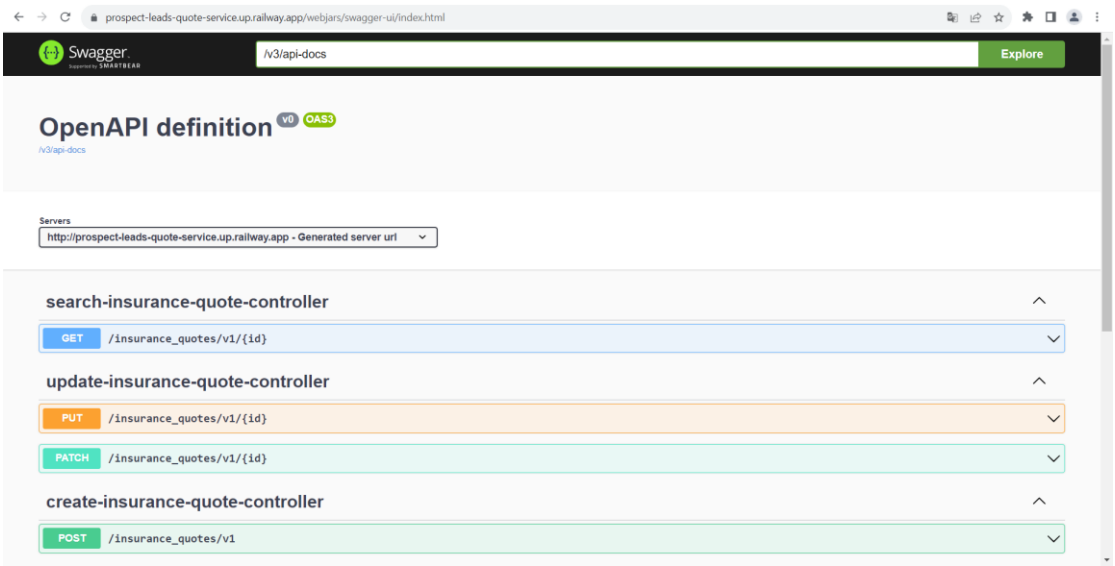

Figura 11 - Especificação OpenAPI da API de Cotações. Fonte: do Autor

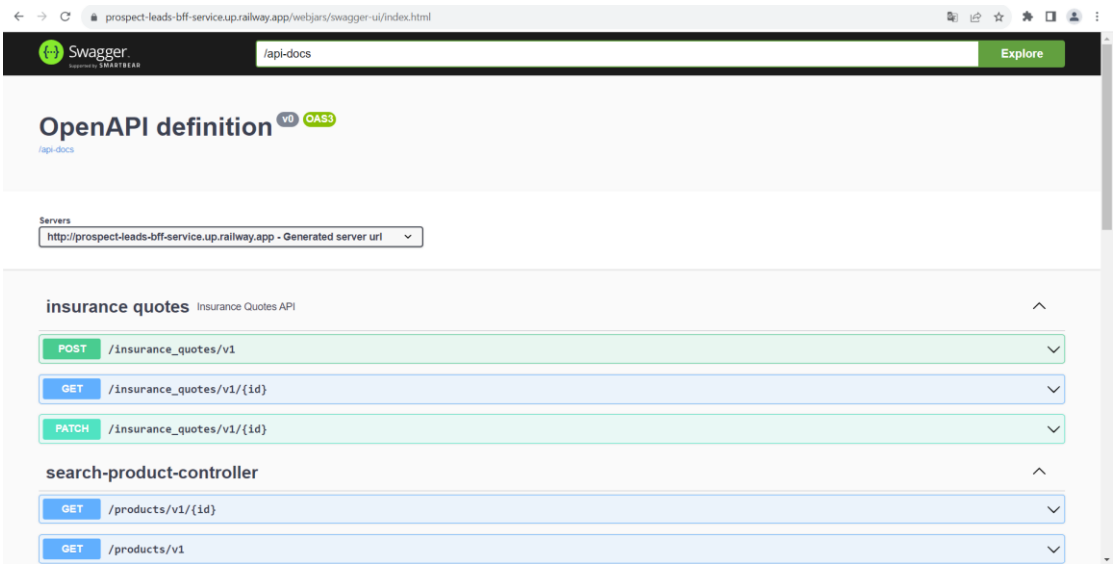

Figura 12 - Especificação OpenAPI da API de BFF. Fonte: do Autor

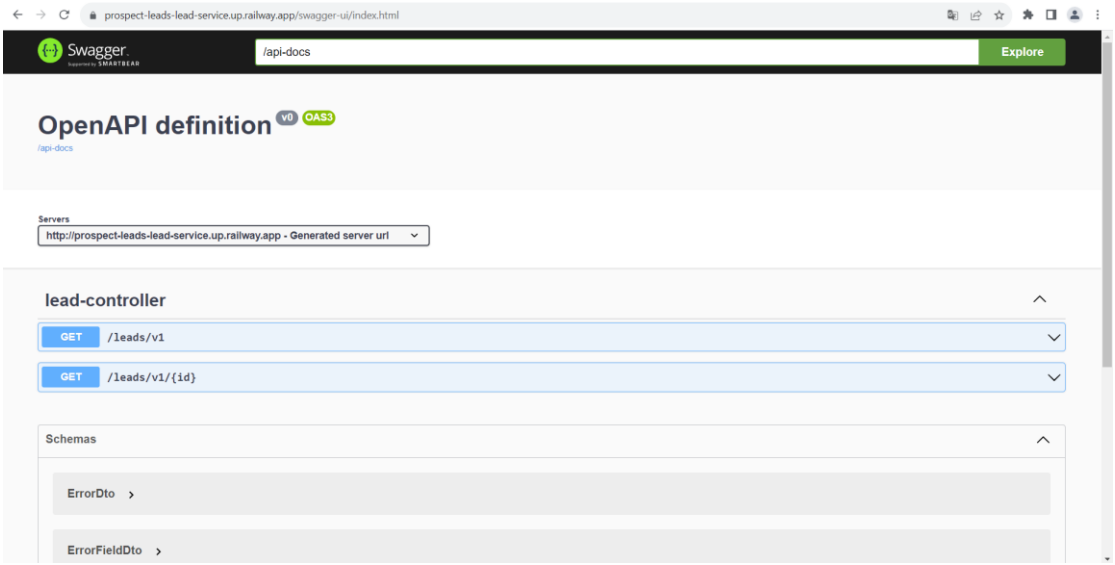

Figura 13 - Especificação OpenAPI da API de Leads. Fonte: do Autor

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	O sistema deve prover boa usabilidade.
Preocupação:	
Uma usabilidade ruim pode prejudicar a experiência e utilização do usuário.	
Cenário(s):	
Cenário 5	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Utilização por parte do usuário através do browser escolhido.	

Mecanismo:	
Criar uma interface amigável com o framework Angular.	
Medida de resposta:	
Tempo gasto para executar uma funcionalidade.	
Considerações sobre a arquitetura:	
Riscos:	A utilização errada de componentes para exibição no browser pode prejudicar a utilização.
Pontos de Sensibilidade:	Não há.
Tradeoff:	Não há.

As figuras 14, 15, 16 e 17 ilustram as telas das funcionalidades da aplicação:

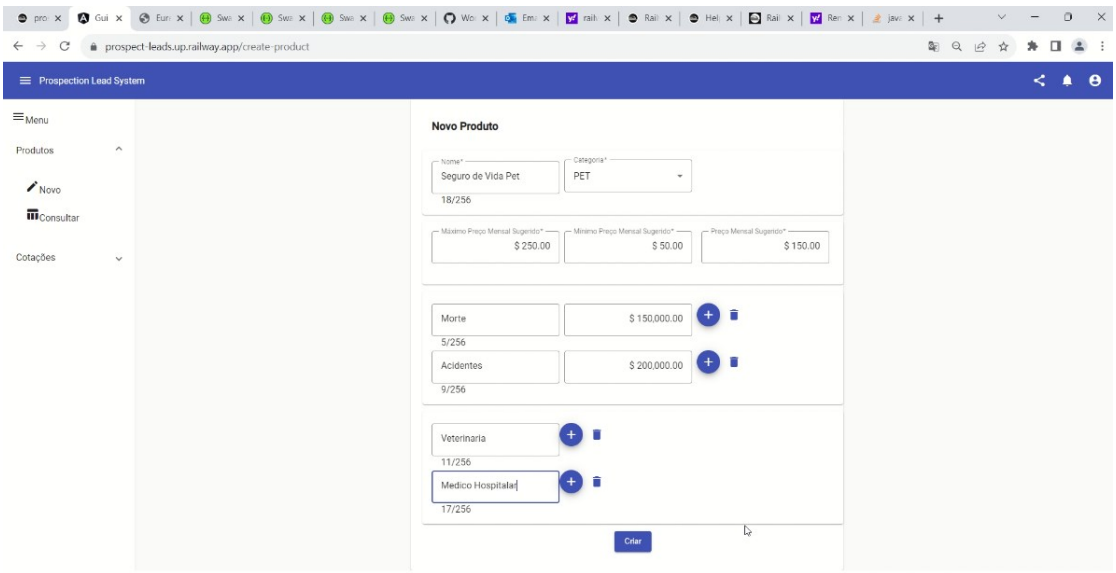

Figura 14 - Tela de Cdastro de Produto. Fonte: do Autor

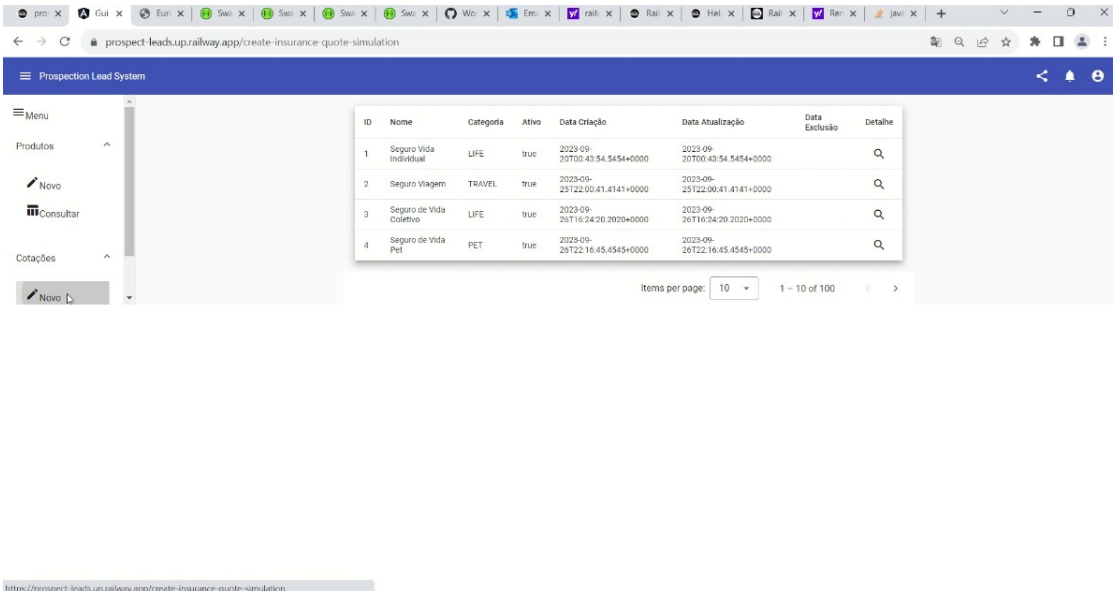

Figura 15 - Tela de Consulta de Produto. Fonte: do Autor

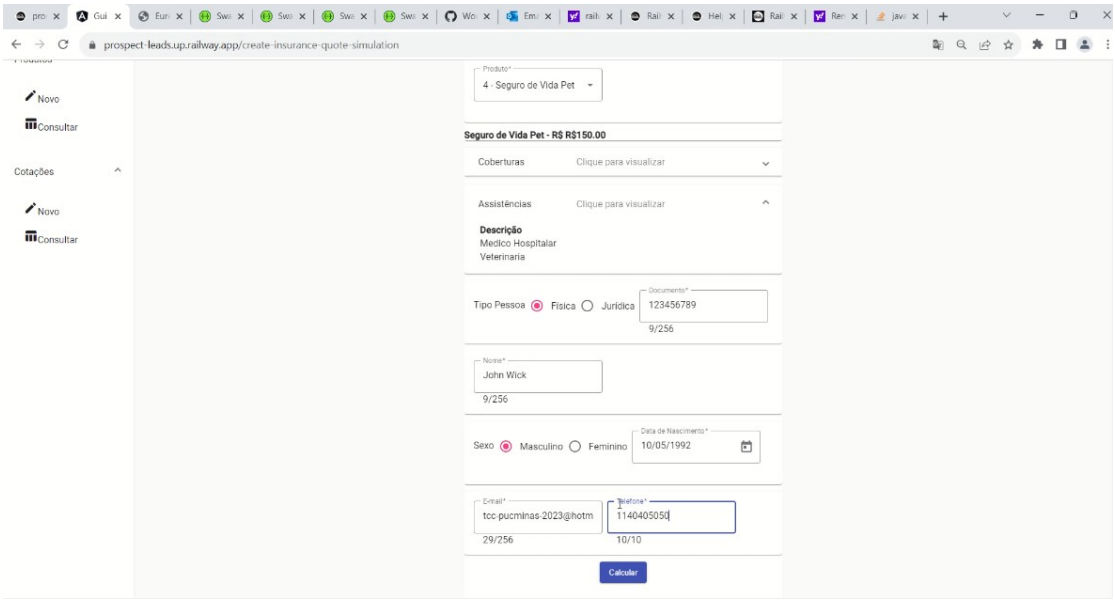

Figura 16 - Tela de simulação de cotação. Fonte: do Autor

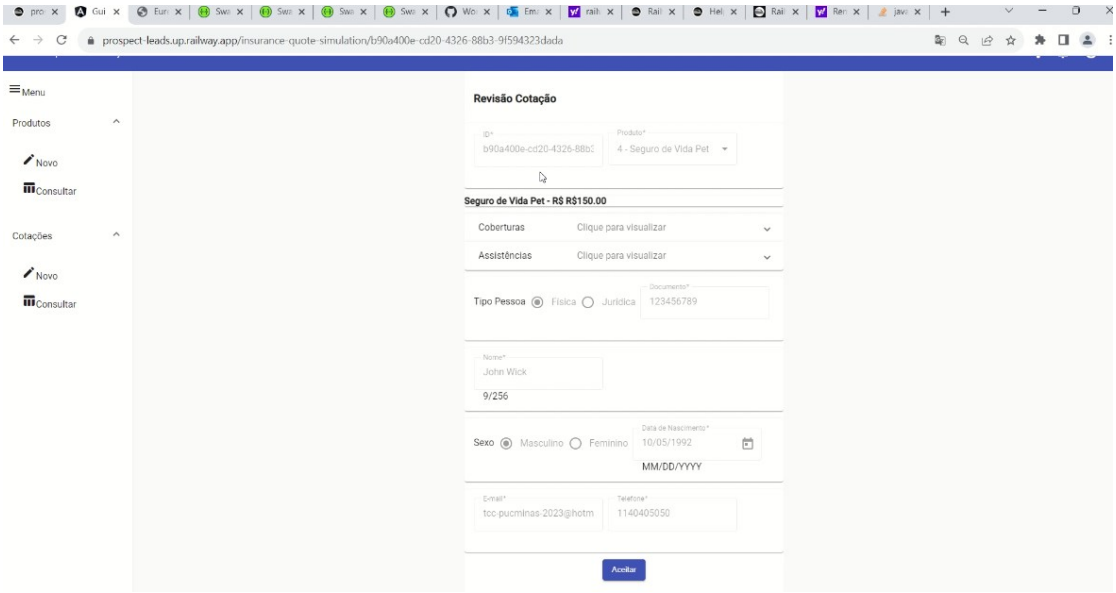

Figura 17 - Tela de resumo de cotação. Fonte: do Autor

Atributo de Qualidade:	Manutenibilidade
Requisito de Qualidade:	O sistema deve permitir uma boa manutenção e evolução.
Preocupação:	
O sistema deve apresentar boa manutenibilidade para permitir evoluções futuras e manutenções rápidas.	
Cenário(s):	
Cenário 6	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Implementação de código-fonte por parte do desenvolvedor.	
Mecanismo:	
Utilização de testes de arquitetura com framework ArchUnit.	
Medida de resposta:	
Resultado dos testes unitários.	
Considerações sobre a arquitetura:	
Riscos:	Uma má arquitetura pode prejudicar evoluções futuras.
Pontos de Sensibilidade:	Não há.

Tradeoff:	Não há.

As figuras 18 exemplifica um teste de arquitetura com o framework ArchUnit.

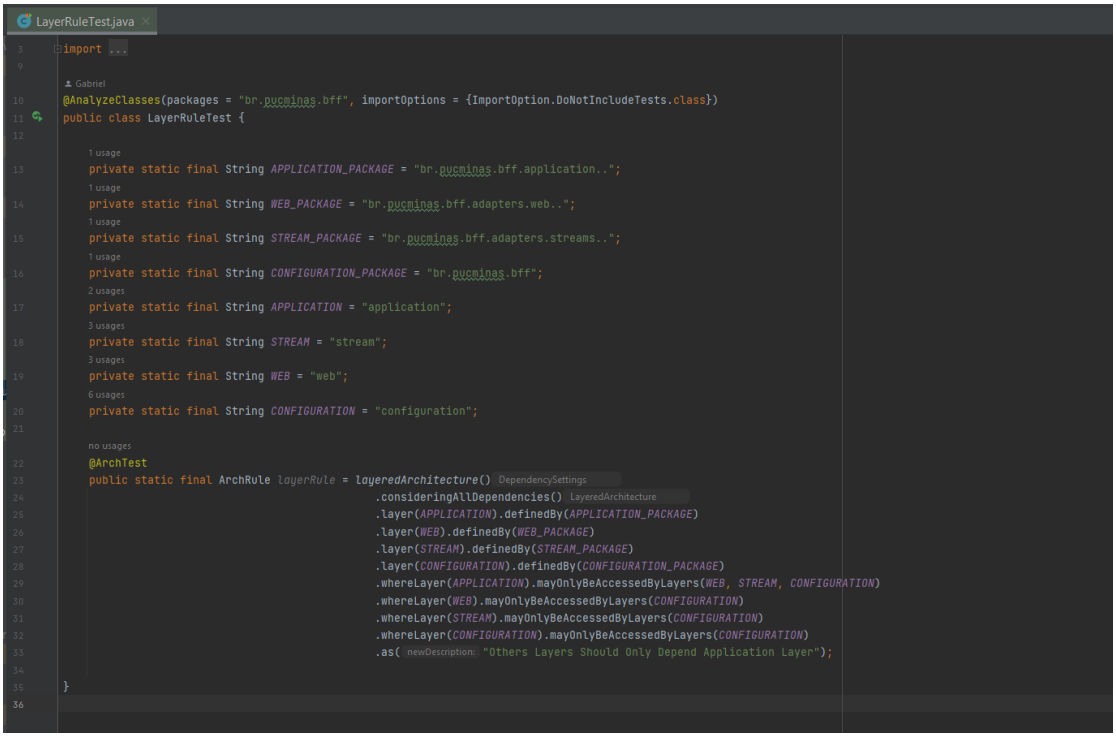

Figura 18 - Exemplo de teste de arquitetura com ArchUnit. Fonte: do Autor

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	Não permitir que o cadastro de produtos possa ser feito por usuários não autenticados.
Preocupação:	
Um sistema sem a correta implementação de autenticação e autorização pode expor funcionalidades sensíveis e causar danos.	
Cenário(s):	
Cenário 7	
Ambiente:	
Sistema em operação normal na plataforma de nuvem Railway.	
Estímulo:	
Requisições HTTP enviadas via browser pelo usuário ou via Insomnia.	
Mecanismo:	

Utilização do framework Spring Security com um Authorization Server com OAuth2.	
Medida de resposta:	
Requisições liberadas ou restritas por funcionalidade	
Considerações sobre a arquitetura:	
Riscos:	A falta de autenticação e autorização expõe funcionalidades e torna o sistema mais vulnerável.
Pontos de Sensibilidade:	Não há.
Tradeoff:	Não há.

Este requisito não foi implementado devido ao grau de complexidade e o tempo disponível para desenvolvimento.

5. Avaliação Crítica dos Resultados

Esta seção apresenta a avaliação dos principais pontos positivos e negativos da arquitetura proposta.

Ponto avaliado	Descrição
Separação de domínios	A separação de domínios proposta parece fazer sentido com cada entidade envolvida no processo de captura de leads, porém uma abordagem com monlito modular poderia apresentar também resultados significativos.
Manutenibilidade	A arquitetura hexagonal utilizada no desenvolvimento das aplicações mostrou uma boa manutenibilidade.
Autenticação e Autorização	A autenticação e autorização poderiam melhorar a segurança do sistema apresentado
Escalabilidade	A substituição da plataforma de nuvem Railway por outra como AWS, Azure ou GCP provavelmente traria melhores recursos de escalabilidade.
Desempenho	A escolha de programação com paradigma reativo para aplicações que teriam a comunicação síncrona (por exemplo o BFF) foi um fator importante para uma boa performance.
Integrações	A comunicação entre sistema via API REST atendeu aos requisitos e a comunicação via mensageria promoveu o desacoplamento das aplicações, porém outras formas de integração como gRPC poderiam ter sido utilizadas, bem como outros serviços de mensageria como AWS SQS, RabbitMQ, ActiveMQ etc.

De forma geral a aplicação funcionou dentro do proposto sendo capaz de captura o lead de simulações de cotação não finalizadas e enviar e-mail para o cliente com uma nova oferta conforme a figura 19.

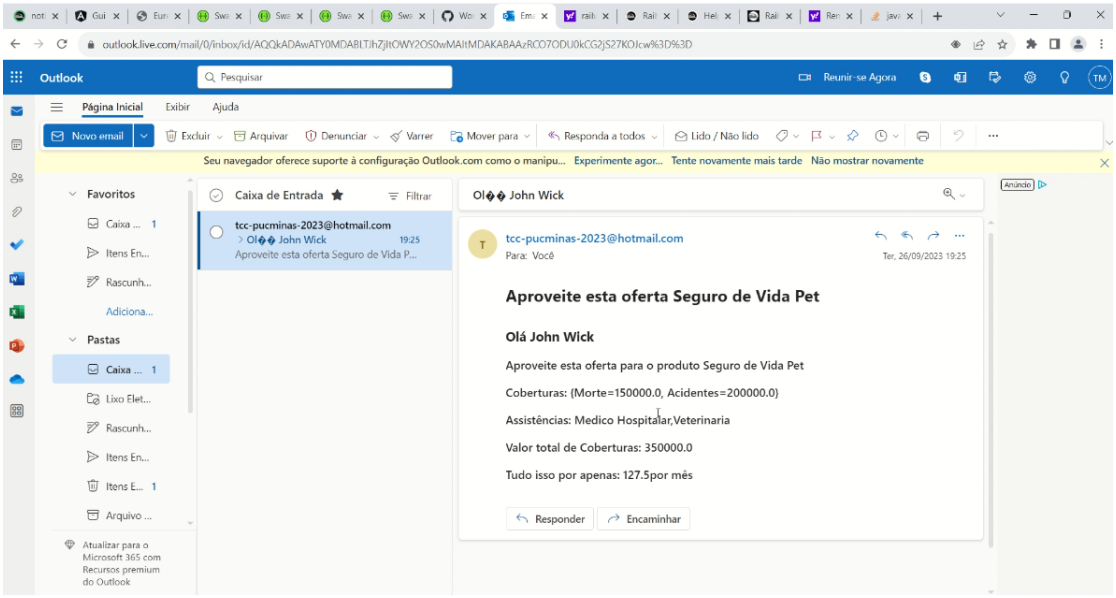

Figura 19 - Exemplo do e-mail de oferta Fonte: do Autor

6. Conclusão

Este trabalho apresentou uma proposta arquitetural de comunicação assíncrona para captura e processamento de leads de seguros. Foram exploradas diversas tecnologias como Java, Spring Boot, Angular, Redis, Apache Kafka, MongoDB, DynamoDB e PostgresSQL. Considerando estas tecnologias o uso da ferramenta LocalStack juntamente com o Docker viabilizou os testes.

Foram apresentadas também sugestões de implantação em nuvem utilizando a plataforma Railway, em que foi possível verificar que os componentes podem funcionar de forma independente, ou seja, poderiam estar parcialmente uma plataforma de nuvem ou outra.

A forma de divisão do microsserviços também demonstrou uma facilidade de manutenção, devido cada aplicação tratar de um domínio específico como produtos, leads e cotações. Nesta abordagem também foi possível perceber o trade-off de um maior número de integrações entre as aplicações que provavelmente não ocorreriam caso fosse utilizado um monolito modular.

Ainda assim, apesar do maior número de integrações se levarmos em consideração os benefícios da arquitetura de microsserviços como escalamento

independente de cada aplicação, facilidade de manutenção e organização é possível evidenciar que esta arquitetura apresenta mais vantagens do que desvantagem, além de ser uma tendência de mercado.

Conclui-se, portanto, que os objetivos foram atingidos, porém alguns detalhes como autenticação e autorização não foram implementados, mas estes detalhes não invalidam o sucesso da prova de conceito.

Como possibilidade de melhoria destaca-se a implementação de autorização e autenticação mencionada anteriormente, desenvolvimento de novas funcionalidades expostas ao usuário como a exibição e controle dos leads capturados e a implantação das aplicações em uma plataforma de nuvem mais robusta e completa.

Referências

ESCOLA NACIONAL DE SEGUROS, Diretoria de Ensino Técnico. **Teoria Geral do Seguro – 6.ed.** Rio de Janeiro: ENS, 2019.

FENACOR. **Mercado de seguros cresceu 16,6% até novembro.** 2023. Disponível em: https://www.fenacor.org.br/noticias/mercado-de-seguros-cresceu-166-ate-novembro. Acesso em: 25 de junho de 2023.

LIPPERT, Dener. **Cientista do Marketing Digital: Como vender para mais pessoas, mais vezes e pelo maior valor - 1.ed.** São Paulo: Editora Gente, 2021.

SANTOS, André. **Seguros: como vender mais e melhor! técnicas e muitas dicas para aumentar seu desempenho – 1.ed**. Rio de Janeiro: FUNENSEG, 2008.

SEBRAE. **6 motivos para consumidores desistirem no momento de concluir a compra**. 2022. Disponível em: https://sebrae.com.br/sites/PortalSebrae/artigos/6-motivos-para-consumidores-desistirem-no-momento-de-concluir-a-compra,ab57ff3754a42810VgnVCM100000d701210aRCRD. Acesso em: 25 de junho de 2023.