**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

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

Projeto Integrado

Relatório Técnico

Sistema de Controle de Infrações de Trânsito

Guilherme Medina Castello

Belo Horizonte

Junho de 2023.

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 2](#_Toc116844011)

[1. Introdução 3](#_Toc116844012)

[2. Especificação Arquitetural da solução 5](#_Toc116844013)

[2.1 Restrições Arquiteturais 5](#_Toc116844014)

[2.2 Requisitos Funcionais 5](#_Toc116844015)

[2.3 Requisitos Não-funcionais 6](#_Toc116844016)

[2.4 Mecanismos Arquiteturais 6](#_Toc116844017)

[3. Modelagem Arquitetural 7](#_Toc116844018)

[3.1 Diagrama de Contexto 7](#_Toc116844019)

[3.2 Diagrama de Container 8](#_Toc116844020)

[3.3 Diagrama de Componentes 8](#_Toc116844021)

[4. Avaliação da Arquitetura (ATAM) 9](#_Toc116844022)

[4.1 Análise das abordagens arquiteturais 10](#_Toc116844023)

[4.2 Cenários 10](#_Toc116844024)

[4.3 Evidências da Avaliação 11](#_Toc116844025)

[5. Avaliação Crítica dos Resultados 12](#_Toc116844026)

[6. Conclusão 13](#_Toc116844027)

[Referências 14](#_Toc116844028)

## Introdução

A cada ano que passa, cresce o número de pessoas habilitadas a conduzir veículos automotores, bem como a frota de veículos em circulação no país. Em um cenário onde o trânsito é cada vez mais complexo, aumentam a possibilidade de acidentes, especialmente com vitimas, e as transgressões as leis estabelecidas no país. Neste cenário se faz necessário a utilização de mecanismos que auxiliem na educação dos condutores, seja por campanhas de conscientização e treinamentos, seja através de multas e penalidades.

O Código de Trânsito Brasileiro prevê a aplicação de multas e processos administrativos como forma educar os condutores. Porém sem um sistema informatizado, esse controle se torna extremamente complexo, suscetível a erros e a possíveis impunidades. Também torna difícil a tomada de decisão para campanhas de conscientizações, pois uma vez que a informação necessária é incorreta ou, muitas vezes, inexistente.

Este cenário, que se apresenta cada vez mais complexo, motivou a elaboração de um sistema de controle de infrações de trânsito que auxilia na fiscalização, na otimização de processos e que fornece as informações necessárias para que o poder público possa tornar a aplicação da lei mais ágil, com menos erros, tendo todas as informações necessárias para a tomada de decisão, e com isso torne o trânsito brasileiro mais seguro.

O objetivo principal deste trabalho é apresentar um projeto arquitetural para um sistema de controle de infrações de trânsito que auxilie a gestão da informação gerada a partir de autos de infração de trânsito aplicados por agentes, bem como a instauração de processos administrativos previstos no Código de Trânsito Brasileiro.

Este trabalho tem os seguintes objetivos específicos:

* Especificar o modelo arquitetural proposto através de requisitos, restrições e mecanismos arquiteturais;
* Apresentar a documentação da arquitetura proposta através de diagramas de contexto, containeres e componentes através do modelo C4;
* Avaliar a arquitetura proposta com o modelo ATAM;

## Especificação Arquitetural da solução

## Restrições Arquiteturais

R1: O sistema deve ser baseado em micros serviços.

R2: As APIs devem seguir o padrão Restful (Exceto com sistema nacional

R3: O sistema deve utilizar mecanismos de filas para processamento assíncrono.

R4: Cada micro serviço deve ter seu próprio banco de dados.

## Requisitos Funcionais

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O usuário deve poder acessar o sistema através de usuário e senha. | B | M |
| RF02 | O usuário deve poder alterar/recuperar a senha para acesso ao sistema. | B | M |
| RF03 | O sistema deve permitir acesso compatível com o perfil do usuário logado. | B | M |
| RF04 | O sistema deve possibilitar a manutenção de um cadastro de artigos conforme Código de Trânsito Brasileiro. | B | B |
| RF05 | O agente de trânsito deve poder cadastrar infrações de trânsito. | M | A |
| RF06 | O agente de trânsito deve poder visualizar/editar as infrações cadastradas por ele. | M | A |
| RF07 | O servidor do Órgão de Trânsito deve poder visualizar/editar as infrações incluídas por seus agentes. | M | A |
| RF08 | O sistema deve instaurar de forma automática processos administrativos por somatório de pontos de acordo com o Código de Trânsito Brasileiro. | A | A |
| RF09 | O sistema deve instaurar de forma automática processos administrativos por infração específica de acordo com o Código de Trânsito Brasileiro. | A | A |
| RF10 | O sistema deve permitir a visualização dos processos administrativos instaurados. | M | A |
| RF11 | O sistema deve ser integrado ao sistema nacional de infrações para envio de infrações cadastradas por agentes de trânsito. | A | A |
| RF12 | O sistema deve ser integrado ao sistema nacional de infrações para recebimento de infrações cadastradas para os condutores do seu estado. | A | A |

## Requisitos Não-funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | O sistema deve ser compatível com nos principais navegadores de mercado (Edge/Chrome/Firefox) | A |
| RNF02 | A aplicação móvel deve ser nativa Android, compatível com a versão 7 ou superior. | A |
| RNF03 | O sistema deve ser protegido contra acessos não autorizados. | A |
| RNF04 | O sistema deve ser resiliente a falhas. | A |
| RNF05 | O sistema deve permitir um fácil escalonamento horizontal e vertical. | M |
| RNF06 | A comunicação na rede deve ser criptografada. | M |

## Mecanismos Arquiteturais

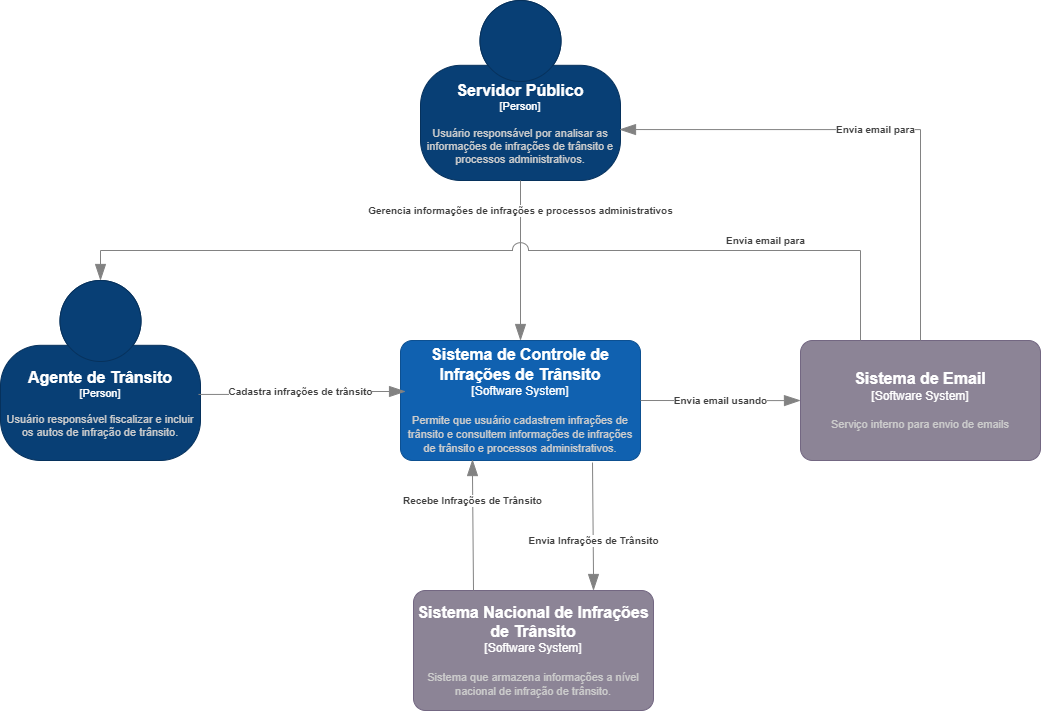
|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | Hibernate/Spring Data |
| Front end | Single Page Application | Angular |
| Back end | Microsserviços | Java/Springboot |
| Mobile | MVVM | Android/Kotlin |
| Integração | Restful Web Services | Spring WebFlux |
| Autorização/Autenticação | OAuth/LDAP | Azure Active Directory/ Spring Security |
| Log do sistema | Logging Framework | Slf4j |
| Observabilidade | Plataforma | Elastic Stack |
| Monitoramento | Plataforma | Prometheus/Grafana |
| Teste de Software | TDD/BDD | JUnit/ Cucumber |
| Resiliência/Tratamento a falhas | Retry/ Circuit Breaker/ Timeout/ Fallback/ Engenharia do Caos | Spring Cloud/ Netflix Chaos Monkey |
| Comunicação | Filas de mensagens | RabbitMQ |
| Versionamento | Git | Github |
| Deploy | CI/CD | Azure DevOps |
| Documentação de API | Especificação OpenAPI | Swagger |

## Modelagem Arquitetural

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à futura implementação.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Dos quatro níveis que compõem o modelo C4, três serão apresentados nesta seção.

## 3.1 Diagrama de Contexto



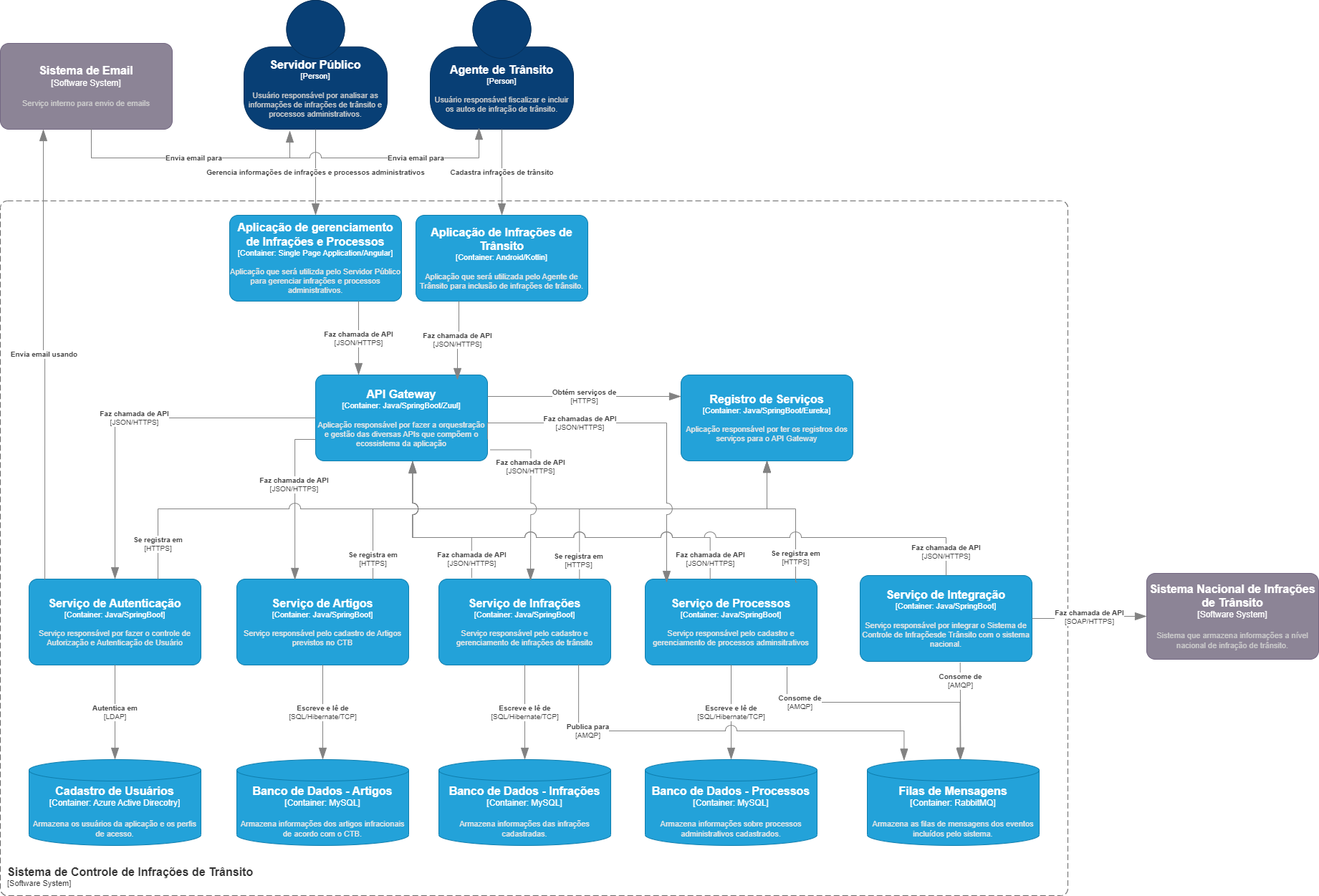
**Figura 1 - Visão Geral do Sistema de Controle de Infrações de Trânsito**

*.*

A figura 1 mostra o contexto no qual o Sistema de Controle de Infração de Trânsito está inserido. De forma geral o sistema terá dois perfis de usuários: Agente de Trânsito e Servidor Público, onde o primeiro será o responsável por realizar a fiscalização do trânsito e quando constatar uma violação as leis de trânsito, cadastrar uma infração uma infração, já o segundo é responsável por fazer o gerenciamento e análise das informações incluídas pelos agentes e que foram geradas a partir das infrações cadastradas pelos agentes.

O sistema também deverá ser integrado com o Sistema Nacional de Infrações, para que seja possível enviar e receber infrações cadastradas em outras unidades da federação.

## 3.2 Diagrama de Container



**Figura 2 –Visão dos containers que compõem o Sistema de Controle de Infrações de Trânsito**

A figura 2 apresenta os containers do Sistema de Controle de Infrações de Trânsito, bem como se dá a comunicação entre eles.

Os containers que os usuários irão interagir diretamente serão as interfaces do sistema e foram divididos em duas aplicações distintas. Enquanto o Servidor Público utiliza a aplicação de gerenciamento infrações e processos, o Agente de Trânsito irá utilizar a aplicação móvel para inclusão de infrações.

Todas as requisições serão feitas através de um API Gateway, com uso de protocolo HTTPS e no padrão Restful/JSON. Esse API Gateway irá rotear as requisições para os micros serviços adequados.

Cada micro serviço terá uma responsabilidade distinta dentro do contexto da aplicação e terá uma base de dados isolada dos demais micros serviços. Todos eles deverão se registrar em um registro de serviços, o qual irá fornecer um ponto único para o API Gateway obter informações dos serviços disponíveis.

O serviço de autenticação será responsável por fornecer mecanismos de autenticação e autorização para os usuários e serviços do sistema. Ele contará com integração ao Active Directory, para unificar o a base de dados dos usuários com demais sistemas utilizados pelos mesmos.

O serviço de infrações é responsável pelo cadastro e gerenciamento das infrações. Sempre que uma infração for incluída no sistema, além de persistir a informação em banco de dados, o serviço irá enviar um evento para a fila de mensagens, desta forma os outros serviços serão notificados deste evento e poderão realizar ações específicas.

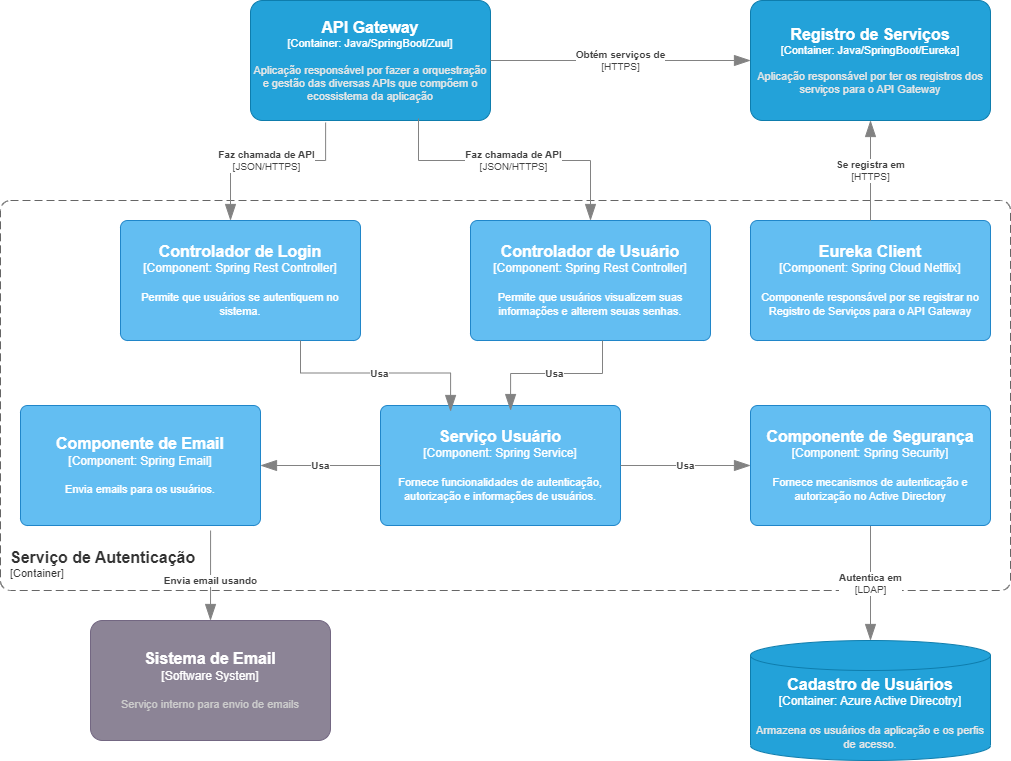
O serviço de processos que tem a responsabilidade de inclusão dos processos administrativos, será acoplado em uma fila de mensagens e a cada evento recebido nesta fila, poderá analisar a informação e então incluir um novo processo administrativo por pontuação ou por infração específica.

A comunicação com o Sistema Nacional de Infrações de Trânsito será responsabilidade do serviço de integração, que também está ligado a fila de mensagens, e poderá enviar a informação para o sistema de destino. Também está previsto nesse serviço o recebimento de infrações vindas do sistema nacional, nesse caso o serviço irá realizar a requisição para incluir uma nova infração utilizando o serviço de infrações.

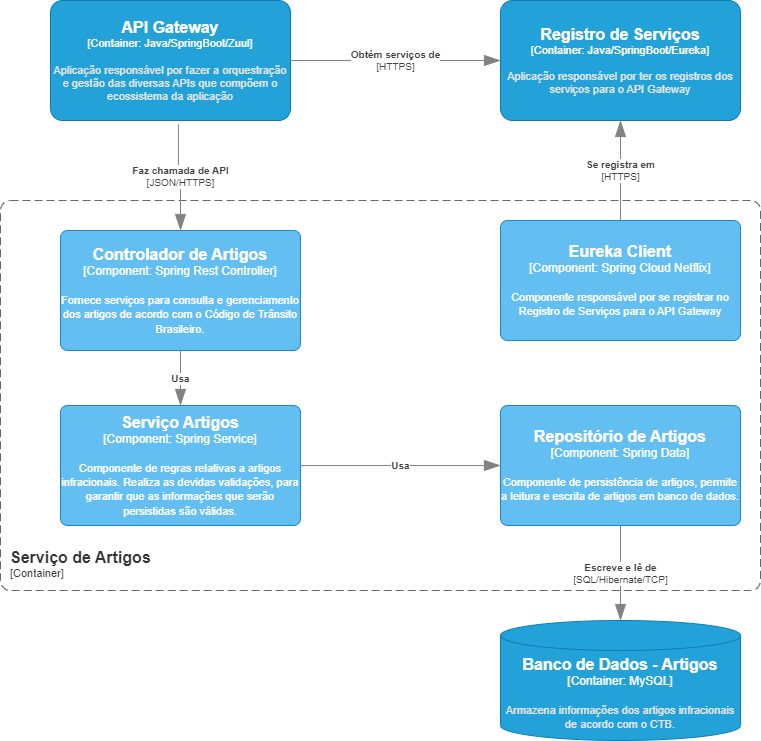
O serviço de artigos será responsável pelo cadastro geral de artigos segundo o Código de Trânsito Brasileiro. Ele também será responsável por prover informações que permitam com que o serviço de processos possa identificar se uma infração poder ser utilizada para compor um processo administrativo, seja por pontuação ou por infração específica.

## 3.3 Diagrama de Componentes

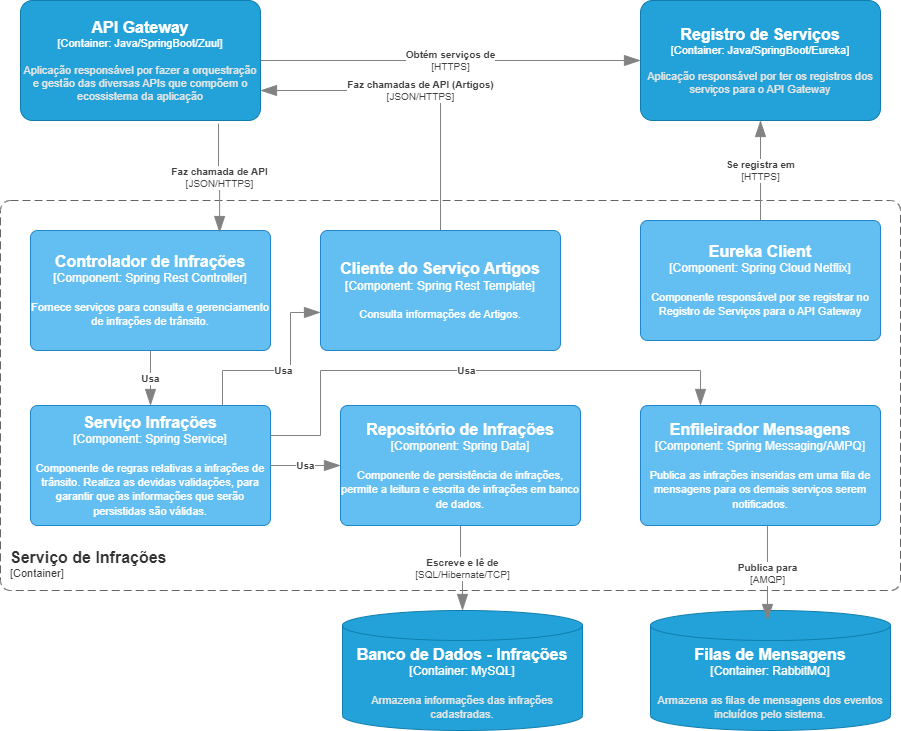
Para uma melhor visualização e entendimento dos componentes do sistema, foi decidido criar um diagrama de componentes para cada um dos containers mais complexos do sistema. Os containers API Gateway, Registro de Serviços e das interfaces de usuários não serão detalhados em forma de diagrama de componentes por terem uma complexidade menor ou por serem compostos apenas por um componente dentro da arquitetura proposta.



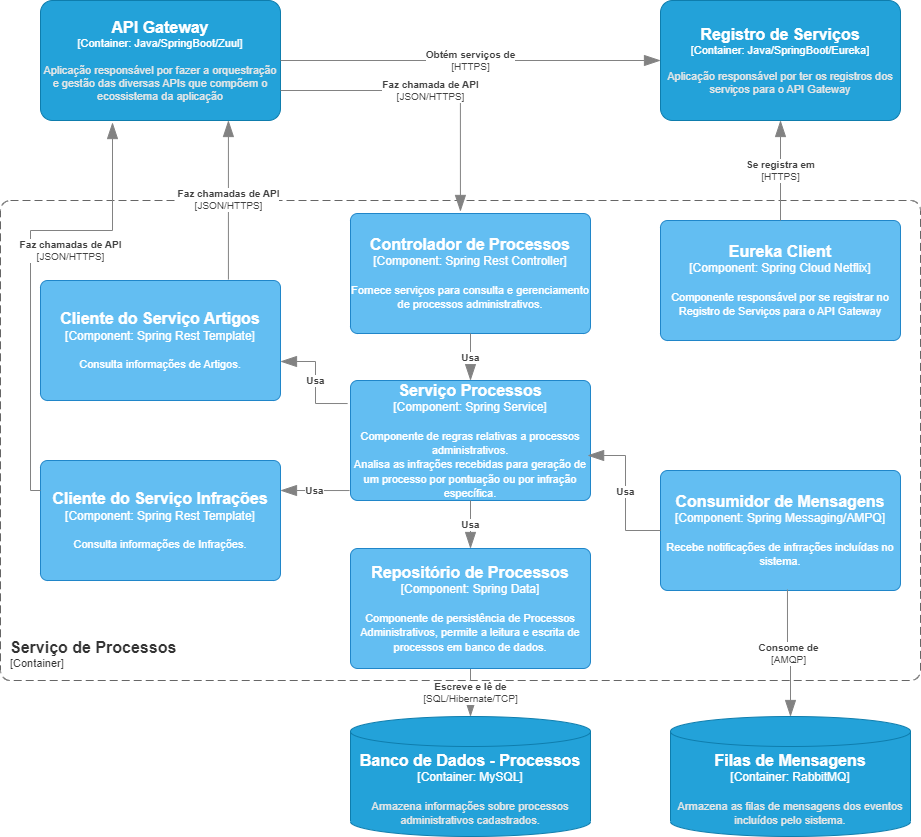
**Figura 3 –Visão dos componentes do container do Serviço de Autenticação**



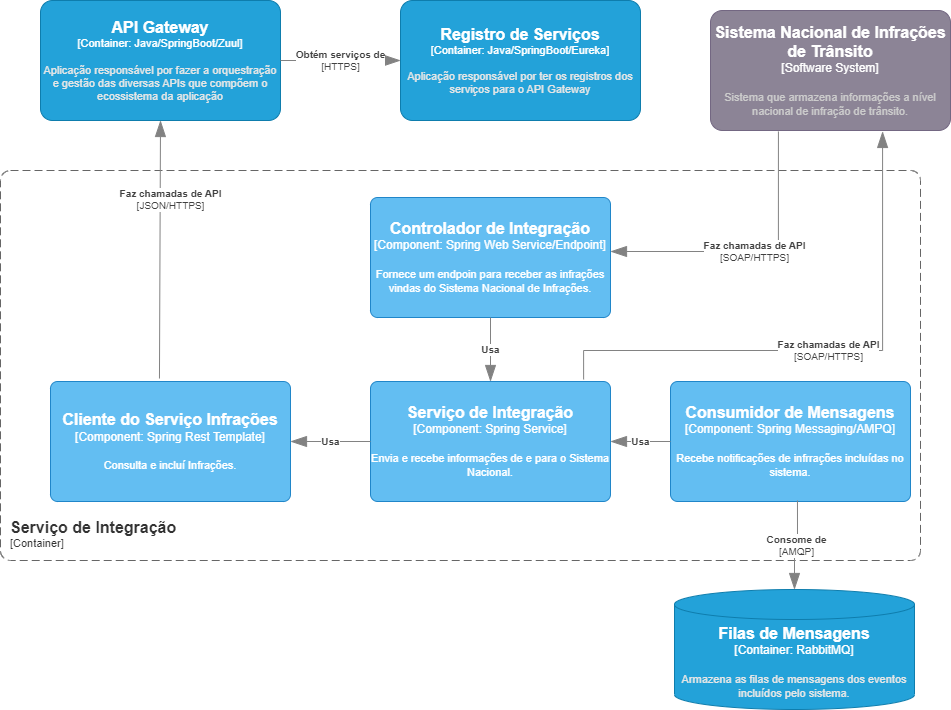
**Figura 4 –Visão dos componentes do container do Serviço de Artigos**



**Figura 5 –Visão dos componentes do container do Serviço de Infrações**



**Figura 6 –Visão dos componentes do container do Serviço de Processos**



**Figura 7 –Visão dos componentes do container do Serviço de Integração**

A arquitetura proposta para o sistema prevê a implementação de um API Gateway , com um registro de serviços, para que todo o tráfego de requisições HTTPS/REST ocorra de forma centralizada e que um micros serviço não precise necessariamente saber o endereço exato dos demais.

Adotar uma arquitetura baseada em micros serviços, com bancos de dados independentes e comunicação via HTTPS e REST, trará benefícios significativos ao sistema. Nessa abordagem, cada micro serviço é responsável por uma função específica do sistema, permitindo uma maior modularidade, escalabilidade e flexibilidade. Com bancos de dados independentes para cada micro serviço, temos a vantagem de isolamento e autonomia, o que evita que uma falha em um serviço afete os outros. Além disso, cada serviço pode escolher o banco de dados mais adequado às suas necessidades, otimizando o desempenho e a eficiência do sistema como um todo. A comunicação entre os micros serviços ocorre por meio de HTTPS e REST, garantindo segurança, confiabilidade e interoperabilidade. O uso do HTTPS (Protocolo de Transferência de Hipertexto Seguro) criptografa as informações transmitidas, protegendo-as de interceptações indesejadas. Enquanto isso, o padrão REST (Representational State Transfer) oferece uma abordagem simples e padronizada para a troca de dados entre os serviços, facilitando a integração e a comunicação eficiente entre eles. Essa combinação de micros serviços com bancos de dados independentes e comunicação via HTTPS e REST promove uma arquitetura robusta, modular e altamente escalável, permitindo um desenvolvimento ágil e uma evolução contínua do sistema.

Cada micro serviço será projetado utilizando arquitetura em camadas, onde a quantidade é definida de acordo com a necessidade específica de cada um. Essa abordagem tem como vantagem a clara separação de responsabilidades e a modularidade, facilitando a manutenção e o desenvolvimento evolutivo do sistema.

A arquitetura proposta também prevê o uso de filas de mensagens juntamente com o padrão de coreografia em micros serviços. Esses aspectos trazem benefícios significativos para o sistema. A combinação de filas de mensagens e padrão de coreografia em micros serviços torna a arquitetura proposta resiliente, escalável e flexível. Ela permite que o sistema seja capaz de lidar com cargas de trabalho variáveis e fornecer uma melhor experiência aos usuários.

Ao adotar filas de mensagens podemos estabelecer uma comunicação assíncrona entre os micros serviços. Isso permite que o serviço de infrações publique mensagens em uma fila, enquanto outros serviços, processos e integração, se inscrevam nessa fila para processar as mensagens de acordo com suas responsabilidades. Essa abordagem desacopla os serviços, tornando-os independentes e flexíveis em relação ao tempo de processamento e disponibilidade. Além disso, as filas de mensagens garantem uma melhor tolerância a falhas, uma vez que as mensagens são armazenadas e podem ser processadas posteriormente, mesmo em situações de indisponibilidade temporária de um serviço.

Em relação ao padrão de coreografia, cada micro serviço é responsável por sua própria lógica de negócios e colabora com outros serviços por meio de eventos e mensagens. Em vez de ter um micro serviço orquestrando o fluxo de trabalho completo, cada serviço reage a eventos e toma ações com base neles. Esse padrão permite uma maior autonomia dos serviços e simplifica a evolução e a manutenção do sistema, uma vez que cada serviço pode ser atualizado ou substituído independentemente dos outros.

Conforme os diagramas apresentados nas figuras 3 a 7, as entidades participantes da solução são:

* **Componente API Gateway:** Presente em todas as figuras, o API Gateway desempenhará um papel fundamental na orquestração e gestão das diversas APIs que compõem o ecossistema da aplicação. Ao centralizar a entrada de todas as solicitações, o API Gateway fornecerá uma interface unificada para os clientes, permitindo o acesso simplificado a funcionalidades e serviços específicos. Além disso, o API Gateway será responsável por gerenciar aspectos cruciais, como autenticação, autorização, segurança e monitoramento de tráfego, garantindo um alto nível de segurança e controle ao sistema como um todo. Essa arquitetura baseada em API Gateway tornará o desenvolvimento, a manutenção e a expansão do sistema mais ágeis e escaláveis, proporcionando uma experiência otimizada e confiável aos usuários finais.
* **Componente Registro de Serviços:** Presente em todas as figuras, o registro de serviços desempenha um papel fundamental no contexto de um API Gateway. Ao implementar um API Gateway, é essencial ter um registro centralizado que permita o gerenciamento e a descoberta de serviços disponíveis na arquitetura. O registro de serviços é responsável por coletar informações sobre os diferentes micros serviços existentes no sistema e fornecer um ponto de entrada único para o API Gateway rotear as solicitações.
* **Componente Eureka Client:** Fará o registro do microsserviço no registro de serviços.
* **Componente de Email:** Presente na figura 3, será responsável por permitir que o sistema se comunique com o provedor de email da organização para envio de emails para os usuários. Ele será utilizado para recuperação e alteração de senha de acesso.
* **Componente de Segurança:** Será o componente responsável por integrar o sistema com Active Directory para autenticação e autorização dos usuários no sistema.
* **Componentes Controladores (Login, Usuário, Artigos, Infrações e Processos):** serão os componentes responsáveis por lidar com as requisições HTTPS/REST recebida em cada micro serviço.
* **Componentes Serviços (Usuário, Artigos, Infrações, Processos e Integração):** são os componentes que irão conter as regras de negócio de cada micro serviço ao qual pertencem.
* **Componentes Repositórios (Artigos, Infrações e Processos):** São os componentes responsáveis por fazer a integração com o banco de dados, fornecendo as interfaces necessárias para que o micro serviço persista as informações em banco de dados.
* **Componente Enfileirador de Mensagens:** irá realizar a publicação das infrações incluídas pelo serviço de infrações. Essa mensagem será enviada para uma fila de infrações para serem consumidas pelos demais serviços.
* **Componente Consumidor de Mensagens:** Presente nas figuras 6 e 7, será responsável por receber as notificações de infrações incluídas e então gerar um processo administrativo, no caso do serviço de processos, e enviar a infração para o sistema nacional, caso do serviço de integração.
* **Componentes Cliente (Rest Template):** Presente nas figuras 5, 6 e 7, serão os componente que irão interagir com o API Gateway para obter dados de domino dos demais micro serviços.
* **Componente Controlador de Integração:** diferentemente dos demais controladores, essa será especificamente para recebimento das infrações do sistema nacional. Ele irá fornecer um endpoint para que o sistema nacional possa enviar as informações a cerca de infrações inseridas em outros estados para a base estadual de cada unidade da federação.

## Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção, visando avaliar se ela atende ao que foi proposto.

## Análise das abordagens arquiteturais

O projeto de proposto busca proporcionar a criação de um sistema robusto e seguro, destacando baixo acoplamento entre os componentes, alta resiliência operacional para enfrentar falhas, facilidade de manutenção por meio de modularização e padrões de codificação, além de medidas abrangentes de segurança. Esses elementos combinados resultam em um sistema confiável, capaz de se adaptar a mudanças, resistir a adversidades e manter a integridade dos dados.

Dentre as características deste projeto, o uso de mensageria com RabbitMQ, em conjunto com uma abordagem de arquitetura orientada a eventos, é um dos principais. Ele nos permite aumentar a resiliência a falhas nos processos adicionais a criação de infrações sem onerar a criação das mesmas. Uma vez que a mensagem seja enviada para uma das filas previstas no sistema, o processamento da mensagem é realizado de forma assíncrona, garantindo uma resposta rápida ao usuário. Em caso de falha no processamento a mensagem é enviada para uma fila secundária (dead letter) e após um período de tempo na fila secundária, essa mensagem é redirecionada para a fila principal para ser processada novamente. Somado a esse mecanismo de retentativas, um mecanismo de “Circuit Breaker” está previsto para caso, muitas mensagens estejam falhando por um serviço estar fora do ar, ao invés de processar e sobrecarregar ainda mais o processamento, redireciona a mensagem automaticamente para a fila secundária.

Na questão de segurança, a adoção de um componente de segurança que utiliza JSON Web Tokens (JWT) em conjunto com o Active Directory oferece várias vantagens notáveis. Ao empregar JWTs, que são tokens compactos e autocontidos, em conjunto com o Active Directory, que é um serviço de diretório robusto, assegura-se uma camada eficaz de proteção e autenticação para o sistema. Isso permite que os usuários se autentiquem de maneira segura e eficiente, reduzindo a complexidade das interações de login.

Além de outras abordagens arquiteturais apresentadas na seção 3.3 juntamente com os diagramas de componentes, a proposta também prevê a adoção do processo de DevOps com a utilização do Azure DevOps por ser uma solução em nuvem e muito versátil.

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de Qualidade** | **Cenários** | **Importância** | **Complexidade** |
| Interoperabilidade | Cenário 1: Compatibilidade com principais navegadores. | A | B |
| Cenário 2: O sistema deve se comunicar com tecnologias diferentes. | A | M |
| Portabilidade | Cenário 3: Compatibilidade com dispositivos Android com versão 7 ou superior. | A | B |
| Segurança | Cenário 4: O sistema deve ser protegido contra acessos não autorizados. | A | M |
| Cenário 5: O sistema deve ser protegido contra acessos não autorizados. | A | M |
| Cenário 6: A comunicação na rede deve ser criptografada. | A | M |
| Confiabilidade | Cenário 7: O sistema deve ser resiliente a falhas. | A | A |
| Escalabilidade | Cenário 8: O sistema deve permitir um fácil escalonamento horizontal e vertical. | M | A |

## 

## 4.2 Cenários

**Cenário 1 – Interoperabilidade**: Ao acessar a aplicação de gerenciamento de infrações e processos com os navegadores Google Chrome, Mozilla Firefox e Microsoft Edge, a aplicação deve funcionar sem qualquer limitação.

**Cenário 2 – Interoperabilidade:** Ao acessar os serviços de artigos, infrações e serviços via HTTP GET, os mesmos devem retornar as informações no formato JSON.

**Cenário 3 – Portabilidade:** Ao instalar a aplicação de infrações de trânsito em um dispositivo Android com a versão 7 ou superior, a aplicação deve funcionar plenamente.

**Cenário 4 – Segurança:** Ao acessar a aplicação de gerenciamento de infrações e processos sem estar autenticado, o sistema deve redirecionar o usuário para a tela de login do sistema, onde será possível inserir as informações de usuário e senha e então se autenticar. Após a devida autenticação, o usuário deve ser redirecionado a tela principal da aplicação onde poderá visualizar as funcionalidades de acordo com seu perfil de acesso.

**Cenário 5 – Segurança:** Ao acessar a aplicação de infrações de trânsito em um dispositivo Android sem estar autenticado, a aplicação deve exibir a tela de login da aplicação e após a devida autenticação, o usuário deve ser redirecionado a tela principal da aplicação onde poderá visualizar as funcionalidades de acordo com seu perfil de acesso.

**Cenário 6 – Segurança:** Ao acessar qualquer serviço do sistema, as requisições devem utilizar HTTPS para garantir que a comunicação entre o navegador/aplicação móvel e os serviços sejam seguros e protegidos através da criptografia utilizada pelo protocolo.

**Cenário 7 – Confiabilidade:** Simular o recebimento de diversas infrações sendo cadastradas, enquanto o serviço de artigos esteja fora de funcionamento. O serviço de processos deve criar uma instrução para processamento e controlar via RabbitMQ o processamento e reprocessamento de criação de processos. No momento em que o serviço de artigos estiver disponível novamente, o serviço de processos poderá recuperar as mensagens impactadas para reprocessá-las normalmente. Esse cenário visa garantir que em caso de mal funcionamento de algum componente, outros componentes que dependem do mesmo, conseguem recuperar-se e garantir o funcionamento do sistema como um todo.

**Cenário 8 – Escalabilidade:** Simular um grande fluxo de infrações sendo criadas pelo serviço de integração, que irá gerar um grande fluxo no serviço de infrações. O serviço de infrações deverá estar inicialmente rodando simultaneamente em duas instâncias, porém nos momentos de sincronia do serviço de integração, através de mecanismos como Kubernetes, o número de instâncias deve ser ajustado para que o fluxo de criação de infrações não afete os agentes de trânsito que estarão utilizando o aplicativo móvel.

## 4.3 Evidências da Avaliação

Apresente as medidas registradas na coleta de dados. Para o que não for possível quantificar apresente uma justificativa baseada em evidências qualitativas que suportem o atendimento ao requisito não-funcional.

|  |  |
| --- | --- |
| 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 1 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| O sistema de monitoramento envia uma requisição para o serviço REST do módulo de informações gerenciais. | |
| Mecanismo: | |
| Criar um serviço REST para atender às requisições do sistema de monitoramento | |
| 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 |

Acrescente imagens e descreva os testes realizados, de tal forma que se comprove a realização da avaliação.

Faça isto para todos os cenários apresentados no tópico 6.1.

## Avaliação Crítica dos Resultados

Apresente aqui, de forma resumida, os principais pontos positivos e negativos da arquitetura proposta. Adote uma postura crítica que permita entender as limitações arquiturais, incluindo os prós e contras das tecnologias. Você pode utilizar o formato textual ou produzir um quadro resumo.

Ex. de quadro resumo:

|  |  |
| --- | --- |
| **Ponto avaliado** | **Descrição** |
| xxxxxxxxxxxxxxx | xxxxxxxxxxxxxxxxxxxx |
|  |  |
|  |  |

## Conclusão

Descreva, de forma sucinta, quais foram as lições aprendidas na execução do seu projeto arquitetural. Procure apresentá-las de tal forma que fiquem configurados os *trade-offs* da arquitetura produzida, como por exemplo, Segurança X Desempenho, Granularidade X Manutenibilidade, etc.

Aqui deve ser apresentado também tudo que se aprendeu com esse projeto, de modo a servir como ajuda para outros profissionais.

Também se faz necessário evidenciar as possibilidades de melhoria do projeto, caso se deseje dar continuidade a ele.Nesse sentido, indique possíveis ajustes ou melhorias arquiteturais, que possam vir a ser realizados futuramente.

Lições aprendidas (ex.):

1. xxxxxxxxxxxxxxxxx
2. xxxxxxxxxxxxxxxxx
3. xxxxxxxxxxxxxxxxx

## Referências

BRASIL. Lei nº 9.503, de 23 de setembro de 1997. Institui o Código de Trânsito Brasileiro. Diário Oficial da União, Brasília, DF, 24 set. 1997. Disponível em: <https://bd.camara.leg.br/bd/bitstream/handle/bdcamara/18141/codigo_transito_8ed.pdf>. Acessado em: 26 jun. 2023

BROWN, Simon, Infoq, “O modelo C4 de documentação para Arquitetura de Software”, publicado em 01/08/2022, disponível em: <https://www.infoq.com.br/articles/C4-architecture-model/>, acessado em: 02/07/2023

TANZU, VmWare, Spring Cloud, Spring.io, publicado em 2022, disponível em: <https://spring.io/projects/spring-cloud>, acessado em: 30/06/2023

Fowler, M. (2002). Padrões de Arquitetura de Aplicações Corporativas. Porto Alegre: Bookman.

RICHARDSON, Chris. Microservices Patterns: With examples in Java. O'Reilly Media, 2018.

Elastic, Elastic Stack, publicado em 2014, disponível em: <https://www.elastic.co/pt/elastic-stack/>, acessedado em 20/06/2023.