

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

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

Projeto Integrado

Relatório Técnico

Sistema de Gestão de Ocorrência - SGO

Thiago Gutenberg Carvalho da Costa

Belo Horizonte
Fevereiro 2022

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

Projeto Integrado – Arquitetura de Software Distribuído	2
1. Introdução	3
2. Cronograma do Trabalho	5
3. Especificação Arquitetural da solução	7
3.1 Restrições Arquiteturais	7
3.2 Requisitos Funcionais	7
3.3 Requisitos Não-funcionais	9
3.4 Mecanismos Arquiteturais	9
4. Modelagem Arquitetural	11
4.1 Diagrama de Contexto	11
4.2 Diagrama de Container	12
4.3 Diagrama de Componentes	14
5. Prova de Conceito (PoC)	19
5.1 Integrações entre Componentes	19
5.2 Código da Aplicação	23
6. Avaliação da Arquitetura (ATAM)	25
6.1. Análise das abordagens arquiteturais	25
6.2. Cenários	26
6.3. Evidências da Avaliação	27
6.4. Resultados Obtidos	53
7. Avaliação Crítica dos Resultados	54
8. Conclusão	57
Referências	58

1. Introdução

Segundo o Dicionário Escolar (2015, p. 284) a infraestrutura nada mais é que um conjunto de serviços básicos (de saneamento, de transporte, de energia e de telecomunicação) tornando-se uma área fundamental para o desenvolvimento socioeconômico de uma região. Nessa perspectiva, à medida em que centros urbanos se expandem, percebe-se a necessidade de manutenção dessas estruturas que compõem uma infraestrutura urbana, se tornando um ponto vital e de extrema importância para a não depreciação das mesmas e também para o bem estar da comunidade.

Neste contexto, o processo de solicitação de manutenção em estruturas dos serviços públicos (poste de luz, canos de distribuição de água, asfalto, etc.) é feito de forma geral unicamente via canal telefônico, onde além da restrição de horário de atendimento sendo somente em horário comercial, limita-se o registro de uma ocorrência distinta por vez para cada intervenção corretiva, mesmo sendo em uma mesma localização. Portanto, mediante este problema, indaga-se: diante de enorme incidência de reclamações em uma região, o processo de registro de ocorrência está sendo feito de forma desburocratizada e eficiente?

Então, afim de simplificar o processo de registro de incidentes pela população e também com o intuito de modernizar, ampliar e prover mais eficiência e transparência, há a necessidade de implementação de um Sistema de Gestão de Ocorrência onde seja possível a abertura de um chamado para resolução de vários problemas relacionados a um tipo de serviço em um determinado endereço.

Sendo assim, o objetivo geral deste trabalho é apresentar a descrição do projeto arquitetural de uma aplicação de gerenciamento de ocorrências de manutenção em estruturas públicas para uma administração regional.

Onde os objetivos específicos são:

- Implementar serviço de ocorrências, parte responsável por expor APIs Web em conformidade com o padrão arquitetural *RESTful* provendo interoperabilidade para realização do processamento negocial e de persistência de uma ocorrência;

Sistema de Gestão de Ocorrência - SGO

- Implementar serviço de localização, que através da comunicação com um sistema externo será responsável por obter informações sobre um endereço a partir de um CEP;
- Implementar serviço de notificações, cuja responsabilidade será o processamento e disparo assíncronos de eventos de notificação relacionados a situação de uma ocorrência;
- Implementar serviço web, responsável por permitir que uma pessoa possa interagir com o sistema, a princípio através de um navegador web com uma interface gráfica responsiva;
- Implementar serviço de relatórios, responsável por disponibilizar relatórios com resumo e/ou estatísticas de atividades realizadas aos interessados.

2. Cronograma do Trabalho

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

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
01/01/2022	07/01/2022	1. Analisar ideias e selecionar uma para ser abordada e servir como a base para elaboração do relatório técnico.	Tema central do projeto integrado.
08/01/2022	15/01/2022	2. Apresentar informações gerais, referentes ao escopo do projeto, mais especificamente quanto ao contexto, a problemática e a motivação do mesmo.	Introdução, objetivo geral e objetivos específicos do projeto.
16/01/2022	22/01/2022	3. Definir os requisitos arquiteturais da solução.	Descrição dos requisitos que podem impactar diretamente ou indiretamente na macro arquitetura da solução.
23/01/2022	01/02/2022	4. Definir os requisitos funcionais da solução.	Descrição do fornecimento de serviços do sistema e como o mesmo deverá se comportar a partir de entradas específicas.
02/02/2022	06/02/2022	5. Definir os requisitos não-funcionais da solução.	Especificar restrições às funcionalidades do sistema.
07/02/2022	07/02/2022	6. Definição dos mecanismos arquiteturais.	Descrição de conceitos técnicos para padronizar aspectos do sistema.
13/02/2022	14/02/2022	7. Criar o diagrama de contexto.	Diagrama de alto nível que mostra de uma forma geral a macroarquitetura da solução no modelo C4.
13/02/2022	14/02/22	8. Vídeo de apresentação inicial.	Apresentação inicial do projeto, ideia completa sobre o projeto.
16/02/2022	19/02/2022	9. Criar o diagrama de container.	Diagrama que detalha o sistema, mostra os bancos de dados, serviços e descreve as tecnologias usadas.
20/02/2022	28/02/2022	10. Criar o diagrama de componentes.	Diagrama com detalhes de um serviço, mostra camadas de integração de componentes na base de código.
16/02/2022	16/05/2022	11. Iniciar o desenvolvimento do projeto.	Protótipo funcional contendo o em seu escopo os requisitos prioritários.
01/03/2022	08/03/2022	12. Criar o diagrama de código.	Diagrama de detalha a composição de classes definidas em um componente, refletindo diretamente o código.
16/05/2022	31/05/2022	13. Avaliar a arquitetura através do modelo ATAM (<i>Architecture Trade-off Analysis Method</i>).	Análise da abordagem da arquitetura com os cenários e apresentação dos resultados.

Sistema de Gestão de Ocorrência - SGO

01/06/2022	08/06/2022	14. Conclusão.	Lições aprendidas e pontos de melhorias na arquitetura.
09/06/2022	11/06/2022	15. Vídeo de apresentação final.	Apresentação completa do projeto realizado, focando a arquitetura.

3. Especificação Arquitetural da solução

3.1 Restrições Arquiteturais

R1: Seguir o padrão arquitetural de microsserviços;

R2: As APIs devem estar no padrão *RESTful* preferivelmente seguindo o princípio HATEOAS (Hypermedia as the Engine of Application State) para prover melhor interoperabilidade;

R3: O back end deve ser implementado utilizando o conjunto de ferramentas para padrões comuns em sistemas distribuídos providos pelo projeto Spring Cloud;

R4: Os componentes que compõem a interface gráfica do sistema devem ter comportamento responsivo para se adequar melhor nas dimensões de tela dos dispositivos dos usuários;

R5: O sistema deve ser complacente à LGPD (Lei Geral de Proteção de Dados).

3.2 Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve permitir que um usuário externo se registre através de um formulário de cadastro indicando nome de usuário, e-mail, primeiro nome, último nome e senha.	M	A
RF02	O sistema deve permitir que o usuário se autentique através de um formulário de acesso (login) onde deverá ser informado um nome de usuário e senha previamente cadastrada.	A	A
RF03	O sistema deve permitir o registro de uma ocorrência por um usuário.	A	A
RF04	O sistema deve permitir a alteração de detalhes de uma ocorrência registrada, caso a mesma tenha sido devolvida pelo atendente para ajustes.	M	A
RF05	O sistema deve permitir que uma ocorrência registrada possa ser cancelada pelo usuário que a criou caso a mesma não esteja com situação já aprovada.	B	B
RF06	O sistema deve permitir que sejam vinculados vários incidentes de um mesmo tipo de serviço em uma ocorrência durante o	M	M

Sistema de Gestão de Ocorrência - SGO

	processo de registro.		
RF07	O sistema deve permitir anexar foto para cada incidente vinculado a uma ocorrência durante o processo de registro.	A	B
RF08	O sistema deve permitir o acompanhamento de uma ocorrência através de uma pesquisa a partir de um número do protocolo.	B	A
RF09	O sistema deve listar os protocolos de ocorrências registradas para o usuário logado.	B	M
RF10	O sistema deve permitir que um atendente analise e classifique priorizando uma ocorrência em uma escala de urgência de: Baixo, Moderado ou Alto.	B	A
RF11	O sistema deve permitir que um atendente altere a situação de uma ocorrência para: Aprovada, Cancelada, Devolvida.	B	A
RF12	O sistema deve permitir a buscar automaticamente um endereço a partir de um CEP informado pelo usuário durante cadastro de uma ocorrência.	M	A
RF13	O sistema deve permitir que o usuário selecione uma localidade a partir de um mapa durante cadastro de uma ocorrência.	A	B
RF14	O sistema deve permitir que o usuário possa desligar o recebimento de notificações via e-mail sobre a situação de uma ocorrência registrada.	M	A
RF15	O sistema deve enviar notificações via componente de interface (<i>badge</i>) sobre a situação da ocorrência para o usuário.	M	M
RF16	O sistema deve enviar notificações via e-mail para o usuário sobre a situação da ocorrência caso o usuário opte por receber esse tipo de notificação.	M	M
RF17	O sistema deve permitir gerar relatórios referente as localizações que mais tiveram ocorrências registradas.	M	A
RF18	O sistema deve permitir gerar relatórios sobre a quantidade de horas de participação de um atendente nas ocorrências - TMA (Tempo Médio de Atendimento)	M	M
RF19	O sistema deve permitir gerar relatórios sobre a quantidade de horas em que uma ocorrência ficou em aberto até ser atendida - TME (Tempo Médio de Espera)	M	M
RF20	O sistema deve notificar o sistema externo de ordem de serviço para gerar uma O.S (Ordem de Serviço) quando uma ocorrência mudar para situação aprovada.	M	A

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

ID	Descrição	Prioridade (B/M/A)*
RNF01	Desempenho — A operação de salvar uma ocorrência tem que ser rápida, não podendo exceder mais do que 3 segundos em média para ser executada.	A
RNF02	Usabilidade — A interface gráfica deve ser responsiva aplicando um design adaptativo para melhorar a disposição do conteúdo na tela do dispositivo melhorando a experiência do usuário.	M
RNF03	Compatibilidade — O sistema deverá funcionar corretamente nos navegadores web modernos mais usados (Google Chrome, Edge e Firefox).	B
RNF04	Segurança — O acesso a APIs privadas precisa ser seguro, exigindo a devida autenticação e autorização do usuário, caso contrário resultará respectivamente nos erros HTTP 401 - (<i>Unauthorized</i>) ou HTTP 403 (<i>Forbidden</i>).	A
RNF05	Padrão — O software deverá ser orientado a camadas possuindo baixo acoplamento e alta coesão seguindo o princípio de Responsabilidade Única para prover melhor manutenibilidade e evolução.	A
RNF06	Confiabilidade — Rastreamento distribuído para coleta e pesquisa de dados de tempo de requisições para detectar possíveis problemas de latência na arquitetura de serviços deverá ter um percentual de amostragem de 100%	A

*B=Baixa, M=Média, A=Alta.

3.4 Mecanismos Arquiteturais

Esta seção descreve os mecanismos que compõem a arquitetura do software fazendo uma ligação entre a definição conceitual da solução (análise) e sua implementação, passando pela visão de desenho.

Análise	Design	Implementação
Segurança - Gestão de Identidade e Acesso	Método de acesso - Single-Sign On (SSO)	Keycloak
Segurança - Autenticação e Autorização	Protocolo de autenticação interoperável	OpenID Connect – OIDC
Persistência – Abstração	Spring Data JPA (Java Persistence API)	Hibernate
Persistência – Relacional	Sistema de gerenciamento de banco	MySQL

Sistema de Gestão de Ocorrência - SGO

	de dados (SGBD)	
Persistência – Não-relacional	Banco de dados distribuído e baseado em documentos	MongoDB
Versionamento – Ferramenta	Sistema de controle de versões distribuído	Git
Versionamento – Sistema online	Sistema online de controle para desenvolvimento de software	GitHub
Front end – Estratégia de implementação web	Single Page Application	Angular
Front end – Componentes	Biblioteca de Componentes de Interface do Usuário	Angular Material
Back end - Framework	Framework de aplicação	Spring
Back end - Ferramentas	Padrões comuns para sistemas distribuídos (Serviço de registro e descoberta, rastreamento distribuído)	Spring Cloud
Integração – Assíncrona	Broker de mensageria	RabbitMQ
Integração – Síncrona	Web Service - REST API	Comunicação de requisição/resposta via protocolo <i>Hypertext Transfer Protocol</i> (HTTP)
Integração - Rastreo de eventos distribuídos	Coletar dados de tempo necessários para solucionar problemas de latência em arquiteturas de serviço	Zipkin
Log do sistema	Simple Logging Facade for Java (SLF4J) API	Logback
Teste de Software	Framework para teste unitário	JUnit 5
Teste de Software – Mock	Framework de mocking para teste unitário	Mockito
Deploy – Distribuição	Unidade padrão de software que empacota o código e todas as suas dependências de forma autocontida.	Docker Container
Deploy – Orquestração	Orquestrar ambientes distribuídos baseados em containers	Kubernetes

4. Modelagem Arquitetural

Esta seção apresenta a modelagem arquitetural através de diagramas hierárquicos de forma a permitir completo entendimento em diferentes níveis de abstração da solução proposta visando à implementação da prova de conceito (seção 5).

4.1 Diagrama de Contexto



Figura 1 – Diagrama de Contexto (Visão Geral da Solução). Fonte: [SGO_C4_Context](#) (Alta resolução).

A figura 1 descreve a macroarquitetura do Sistema de Gestão de Ocorrência através do diagrama de contexto, mostrando como o sistema se encaixa no mundo em termos das pessoas que o utilizam e dos outros sistemas de *software* com os quais ele interage. Em azul está o Sistema de Gestão de Ocorrência que será construído, usuários externos e internos podem interagir com ele afim de registrar ou atender uma ocorrência. Em cinza, são os sistemas de software externos que já existem no qual o Sistema de Gestão de Ocorrência irá interagir com eles para buscar uma localização ou para notificar sobre a situação de uma ocorrência para que seja gerada uma ordem de serviço.

4.2 Diagrama de Container

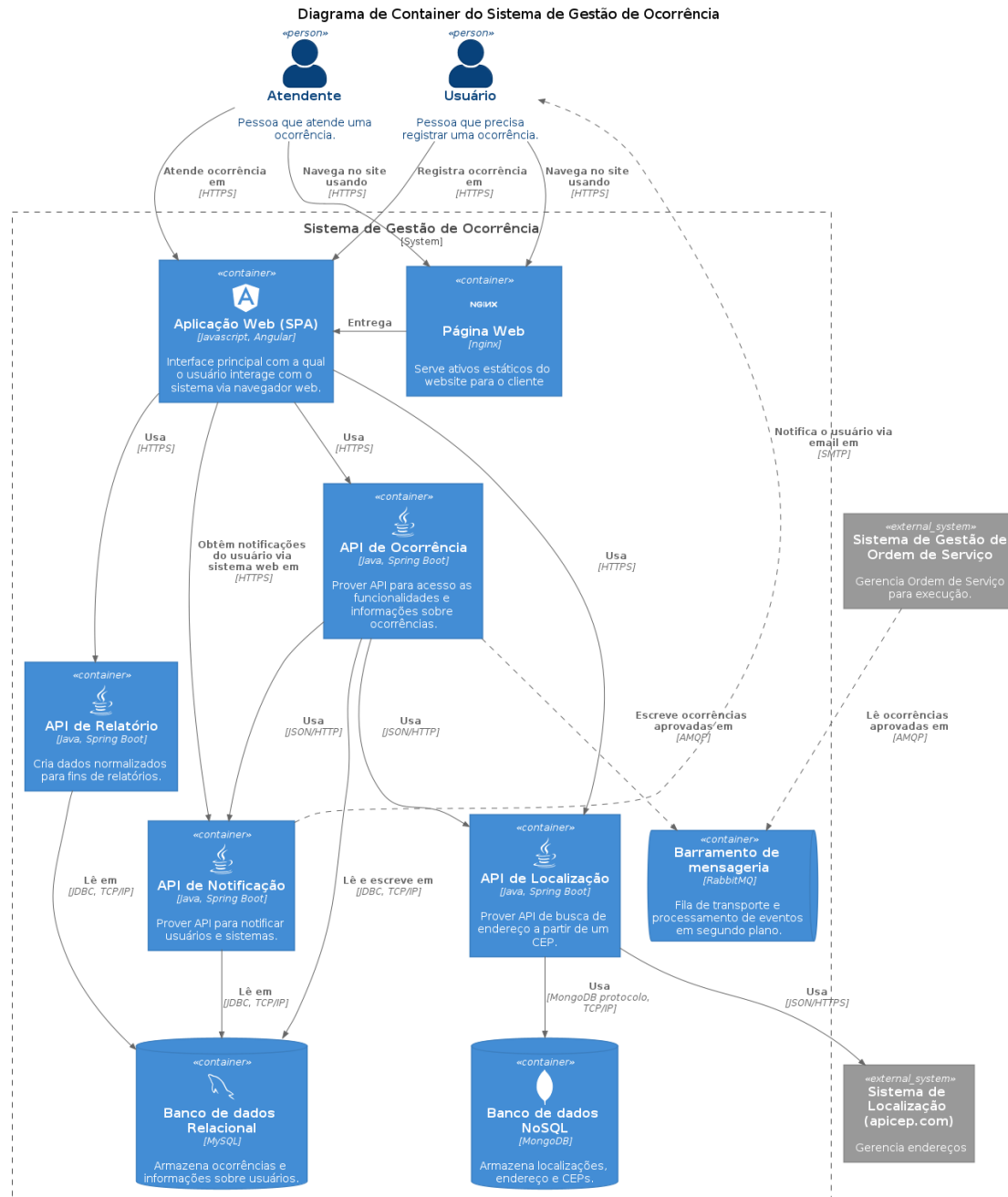


Figura 2 – Diagrama de Container. Fonte: SGO C4 Container (Alta resolução).

A figura 2 detalha através do diagrama de *container* mostrando de forma ampliada os *containers* (em azul) que compõem o Sistema de Gestão de Ocorrência, onde em cada *container* é evidenciado sua natureza, bem como a

tecnologia que é usada e também os protocolos de integração com os quais eles interagem entre si. A página web é onde são servidos os conteúdos estáticos (HTML, CSS e JavaScript) fornecido pelo NGINX que compõem a aplicação web feita em Angular que por sua vez será renderizada aos usuários via navegador web. Os *containers* cujo nome começa com “API de” são microserviços feito em Java/Spring Boot que expõem APIs *RESTful* provendo funcionalidades específicas e coesas para determinados assuntos e podem utilizar o banco de dados relacional para manter dados. O microserviço de localização se comunica com um outro sistema externo, para busca de informações sobre endereços relacionados a um determinado Código de Endereçamento Postal – CEP e utiliza o banco de dados NoSQL para persistir e ler dados. O microserviço de ocorrência mantém ocorrências criadas pelo usuário e se integra com um sistema externo de atendimento e gestão de ordem de serviços via barramento de mensageria enviando para uma fila RabbitMQ as ocorrências em situação aprovada. O microserviço de notificação mantém notificações relacionadas às situações das ocorrências e pode avisar o usuário via e-mail. O microserviço de relatório usa as informações mantidas em banco de dados, agregando-as para geração de relatórios.

4.3 Diagrama de Componentes

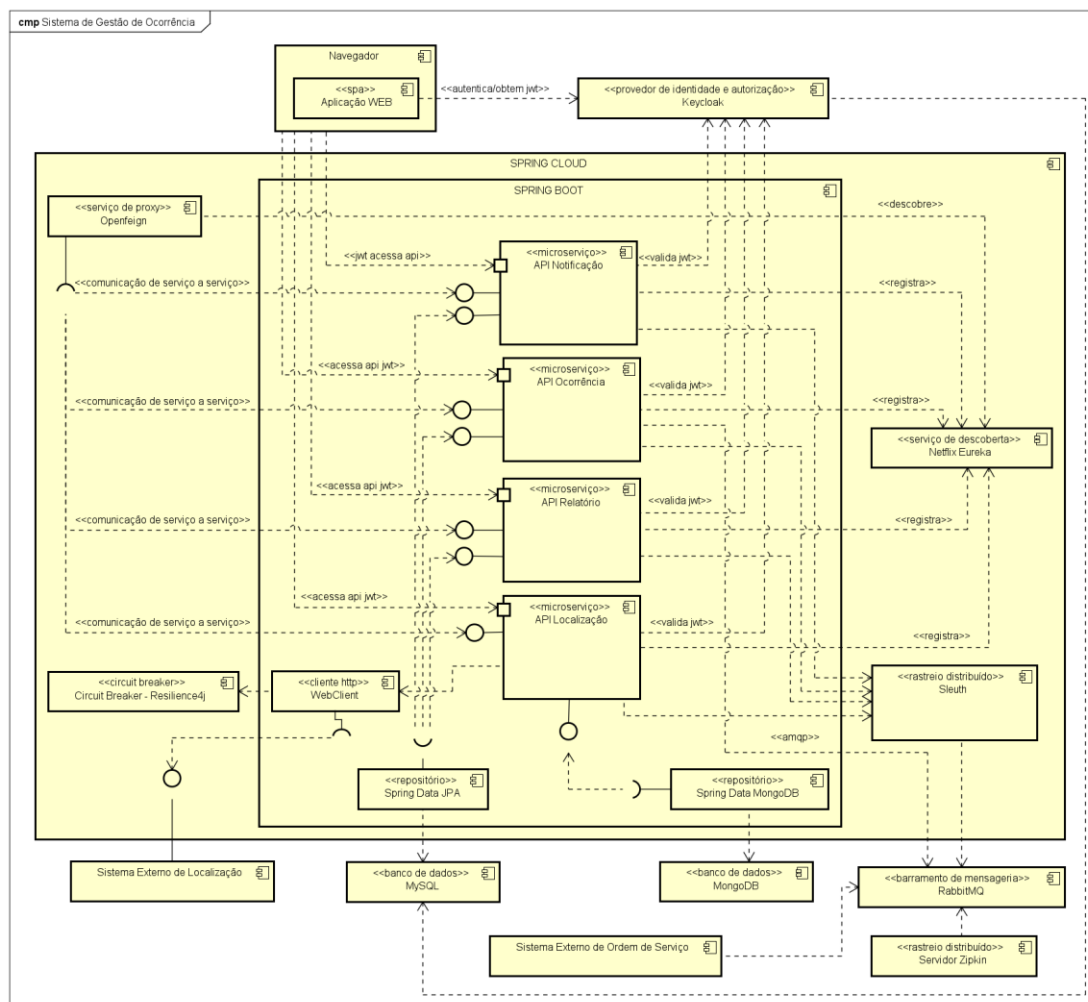


Figura 3 – Diagrama de Componentes. Fonte: [SGO UML Component](#) (Alta resolução).

A figura 3 mostra a estrutura do Sistema de Gestão de Ocorrência, descrevendo suas dependências e suas interfaces reutilizáveis e substituíveis através do diagrama de componentes da aplicação baseado na UML.

Conforme diagrama apresentado na figura 3, o sistema segue padrão/estilo arquitetural orientado a microserviços, respeitada a definição de Restrição Arquitetural, onde a aplicação é dividida em uma coleção de serviços (módulos) separados, possuindo características de baixo acoplamento, independência de implantação, alta manutenibilidade e testabilidade e separação de capacidades negociais, permitindo então uma entrega rápida, frequente e confiável de funcionalidades. Além disso, proporciona também oportunidade de evolução dos

módulos no que tange à utilização de outras tecnologias e/ou linguagens de programação.

Padrões comuns para sistemas distribuídos, definidos nos Mecanismos Arquiteturais, são seguidos nessa arquitetura e estão representados no diagrama da figura 3, sendo eles:

- Padrão de Registro de Serviços, viabilizado através dos componentes Netflix Eureka, mantém um cadastro das instâncias e localidades dos serviços facilitando a descoberta e solicitação sem a necessidade de que um cliente e/ou *API gateway* saiba exatamente o IP e porta de um determinado microserviço para eventuais solicitações. As instâncias de serviço são registradas durante inicialização e desregistradas no encerramento.
- Padrão de Rastreo Distribuído, agrega o conceito de observabilidade, onde através do componente Sleuth são atribuídos identificadores únicos para as requisições. Essa identificação é mantida por toda cadeia de processamento em todos os microserviços acionados e métricas de tempo transcorrido sobre um percentual de amostragem das solicitações e operações realizadas são enviadas à uma fila (Componente RabbitMQ) e consumidas por um serviço centralizado (Componente Zipkin) afim de identificar possíveis problemas de latência em arquiteturas de serviços distribuídos.
- Padrão de *Circuit Breaker*, através do componente Resilience4J proporciona confiabilidade durante integrações com APIs remotas, rejeitando novas invocações por um certo período depois que falhas consecutivas são observadas, evitando assim onerar ainda mais um serviço que já está sobrecarregado, não gerando um efeito dominó, ou seja, derrubando serviços de forma consecutiva. Respostas alternativas (*fallbacks*) podem ser retornadas para o serviço invocador como por exemplo uma resposta cacheada.

Os elementos do modelo que representam partes independentes e intercambiáveis da solução são:

- Componente API de Ocorrência – Microserviço desenvolvido utilizando o framework SpringBoot, expõe *endpoints RESTful* acessíveis a partir de

solicitações HTTP para um IP e porta específico juntamente com um JSON Web Token no cabeçalho de autenticação para possibilitar a manipulação de dados que compõem uma ocorrência e através do processamento negocial manter as informações no banco de dados MySQL. Esse serviço depende de um *Identity Provider* (IdP) representado pelo componente Keycloak para verificar a identidade do usuário que invoca a API.

- Componente API de Localização – Microserviço desenvolvido utilizando o framework SpringBoot, expõe *endpoints RESTful* acessíveis a partir de solicitações HTTP para um IP e porta específico juntamente com um JSON Web Token no cabeçalho de autenticação para possibilitar busca de informações de endereço relacionadas à um CEP. Esse serviço, através de um cliente HTTP, se integra com um sistema externo para realizar solicitações de busca, obtendo como retorno o endereço em formato JSON que por sua vez é cacheado no banco de dados MongoDB para propiciar certa confiabilidade de resposta caso a integração externa tenha algum problema. Esse serviço depende de um *Identity Provider* (IdP) representado pelo componente Keycloak para verificar a identidade do usuário que invoca a API.
- Componente API de Notificação – Microserviço desenvolvido utilizando o framework SpringBoot, expõe *endpoints RESTful* acessíveis a partir de solicitações HTTP para um IP e porta específico juntamente com um JSON Web Token no cabeçalho de autenticação para inserção e busca de mensagens informativas relacionadas a uma ocorrência no banco de dados MySQL. Esse serviço depende de um *Identity Provider* (IdP) representado pelo componente Keycloak para verificar a identidade do usuário que invoca a API.
- Componente API de Relatório – Microserviço desenvolvido utilizando o framework SpringBoot, expõe *endpoints RESTful* acessíveis a partir de solicitações HTTP para um IP e porta específico juntamente com um JSON Web Token no cabeçalho de autenticação para gerar relatórios em PDF ou Excel e/ou obter informações agregadas no banco de dados MySQL para servir de insumo para dashboards para auxiliar em tomadas de decisões. Esse serviço depende de um *Identity Provider* (IdP) representado pelo

componente Keycloak para verificar a identidade do usuário que invoca a API.

- Componente Aplicação Web – Portal Web desenvolvido com Angular e Angular Material, permiti que uma pessoa possa interagir com o sistema, através de um navegador web com uma interface gráfica responsiva e simples, possui integração com um provedor de identidade através do OpenID Connect, camada de identificação baseada no protocolo OAuth 2, para prover acesso às funcionalidades privadas que exigem autenticação.
- Componente Keycloak – Servidor de autenticação para gerenciamento de acesso e identificação de usuários e sistemas. Os serviços dependem desse *Identity Provider* (IdP) para verificação da identidade do usuário que invoca uma API.
- Componente Spring Cloud OpenFeign – Cliente declarativo para serviço web que fornece funcionalidade de chamada HTTP balanceada e integração com serviços de descoberta como o Eureka.
- Componente Spring Cloud Netflix Eureka – Serviço de registro e descoberta de serviços, como um catálogo para proporcionar alta disponibilidade de comunicação entre microserviços em uma arquitetura distribuída.
- Componente Spring Cloud Sleuth – Componente que proporciona rastreamento distribuído através da adição de identificadores ao Slf4J *Mapped Diagnostic Context* (MDC) e integração com serviços agregadores de log como o Zipkin.
- Componente Spring Cloud Circuit Breaker – Componente que tem função semelhante à um disjuntor para prover tolerância a falhas durante integração com outros serviços.
- Componente Spring Data JPA – Componente que facilita e abstrai a complexidade de implementação de repositórios (camada de acesso a dados) baseados na especificação *Java Persistence API* (JPA) para reduzir o esforço de acesso à um banco de dados.
- Componente Spring Data MongoDB – Componente que fornece integração com banco de dados orientados a documentos através de um modelo

centralizado em POJO para interagir com coleções facilitando a implementação de repositórios para acesso à camada de dados. Propiciando acesso aos endereços das ocorrências.

- Componente Spring WebClient – Cliente Web de alta performance para fazer chamadas HTTP utilizando uma abordagem de Reatividade através do projeto Spring Reactor. Proporcionando características de baixa latência e alta produtividade, além de não bloqueio durante solicitações HTTP de integração com outros serviços.
- Componente MySQL – Banco de dados relacional escalável e de alto desempenho que é bastante popular. Usado principalmente pelo sistema para manter as informações oriundas do processo de gestão de ocorrências e de regras de perfil de acesso de usuários.
- Componente MongoDB – Banco de dados orientado a documento. Usado para cadastro e busca de informações relacionadas à localização da ocorrência, como informações de endereço e CEP. E no futuro poderá auxiliar em funcionalidade de Geo Localização.
- Componente Zipkin – Sistema de rastreamento distribuído onde através da coleta de dados resume a porcentagem de tempo gasto em um serviço e se as operações falharam ou não, ajudando a solucionar problemas de latência em arquiteturas orientada a serviços.
- Componente RabbitMQ – Barramento de mensageria extremamente performático, bastante usado e conhecido. Sua principal função no sistema é prover uma comunicação assíncrona através de tipo de mensagens enviadas à uma determinada fila para processamento futuro. Sendo um componente que permite uma resiliência de integração entre serviços.
- Componente Sistema externo de Localização – Sistema externo que auxilia na busca de informações sobre endereços a partir de um CEP através de um catálogo de endereços atualizados.
- Componente Sistema externo de Ordem de Serviço – Representa um sistema fictício que é responsável por processar ocorrências que após um atendimento inicial tiveram a situação aprovada. Iniciando então o ciclo de

conserto e resolução do problema que foi reportado através do registro de ocorrência.

5. Prova de Conceito (PoC)

Esta seção apresenta a prova de conceito arquitetural, detalhando e avaliando importantes aspectos das funcionalidades em relação ao requisito funcionais e não-funcionais da arquitetura.

5.1 Integrações entre Componentes

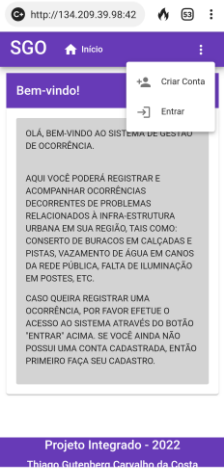
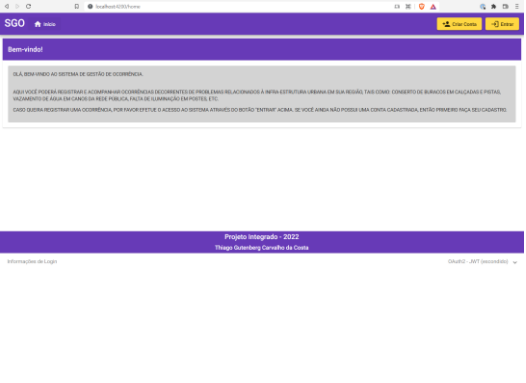
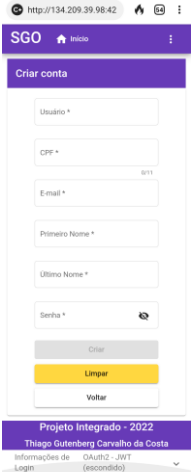
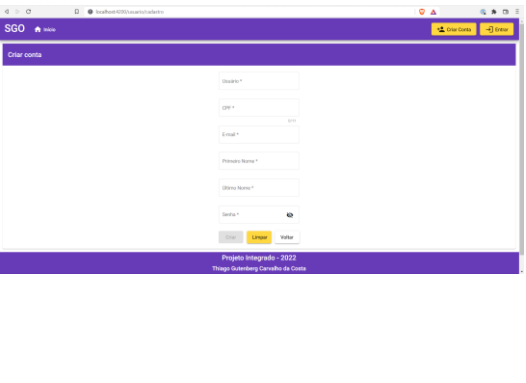
A PoC foi desenvolvida a partir de requisitos prioritários coerentes a especificação proposta e implementada de maneira eficaz afim de permitir a validação das integrações entre os componentes.

Requisitos Funcionais (prioritários)	Componentes e <i>middlewares</i> envolvidos	Integrações e protocolos de comunicação	Requisitos Não-Funcionais
Cadastro de usuários.	Portal Web	Comunicação com a API <i>RESTful</i> de usuários (HAL+JSON).	Segurança, Usabilidade, Compatibilidade
	API Ocorrência	Comunicação TCP/JDBC com MySQL e via REST (JSON) com a API do Keycloak.	Segurança, Padrão, Desempenho, Confiabilidade.
	Keycloak	Comunicação com MySQL via TCP/JDBC.	Segurança
Acesso (login) de usuários.	Portal Web	Comunicação via OAuth2/ <i>OpenID Connect</i> com o Keycloak.	Segurança, Usabilidade, Compatibilidade.
	Keycloak	Comunicação com MySQL via TCP/JDBC.	Segurança
Registro de ocorrência.	Portal Web	Comunicação com a API de ocorrências e endereços via <i>RESTful</i> (HAL+JSON).	Segurança, Usabilidade, Compatibilidade

Sistema de Gestão de Ocorrência - SGO

	API Ocorrência	Comunicação com banco de dados MySQL via TCP/JDBC.	Segurança, Padrão, Desempenho, Confiabilidade.
	API Localização	Comunicação com API de externa de CEP via REST (JSON).	Padrão.
Lista de ocorrências.	Portal Web	Comunicação com a API de ocorrências (HAL+JSON).	Segurança, Padrão, Desempenho, Compatibilidade.
	API Ocorrência	Comunicação com banco de dados MySQL via TCP/JDBC.	Padrão.

As telas do sistema foram desenhadas para proporcionar uma boa usabilidade tanto em ambiente *mobile* quanto *desktop*, utilizando componentes Angular Material e Angular Flexbox. São elas:

Mobile	Desktop
Tela Inicial	
	
Tela Cadastro	
	
Tela Login	

Sistema de Gestão de Ocorrência - SGO



Figuras das telas do sistema – Fonte: <https://bit.ly/37QpUCY> (Alta resolução).

5.2 Código da Aplicação

Nesta seção é apresentado o diagrama de classes de componentes que foram pensados para o Sistema de Gestão de Ocorrência - SGO.

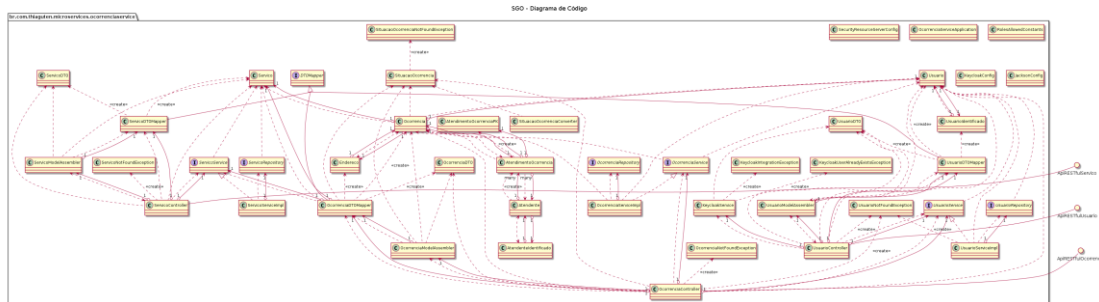


Figura 4 – Estrutura de código do módulo de ocorrências. Fonte: [Módulo-Ocorrência C4-Code](#) (Alta resolução).

A estrutura da aplicação mostrada na figura 4 apresenta os componentes de código e suas funções no software implementado:

- Conjunto de classes para configurações das características da aplicação e de integração, tais como configurações de segurança e interoperabilidade, além de classes de exceções, de integração com serviços de gestão de identidade, classes de suporte, etc.
- Conjunto de classes que compõem API *RESTful* de usuários, classes de modelo de dados relacional, classes da camada de validação negocial e classes para persistência e busca de informações no banco de dados.
- Conjunto de classes que compõem API *RESTful* de serviços, classes de modelo de dados relacional, classes da camada de validação negocial e classes para persistência e busca de informações no banco de dados.
- Conjunto de classes que compõem API *RESTful* de ocorrências, classes de modelo de dados relacional, classes da camada de validação negocial e classes para persistência e busca de informações no banco de dados.

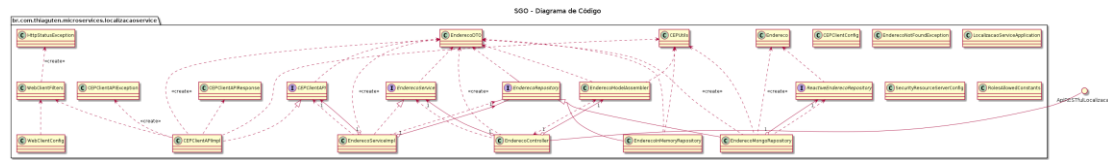


Figura 5 – Estrutura de código do módulo de localização. Fonte: [Módulo-Localização C4-Code](#) (Alta resolução).

A estrutura da aplicação mostrada na figura 5 apresenta os componentes de código e suas funções no software implementado:

- Conjunto de classes para configurações das características da aplicação e de integração, tais como configurações de segurança e interoperabilidade, além de classes de exceções, de integração com serviços de consulta de CEP, etc.
- Conjunto de classes que compõem API *RESTful* de endereços, classes de modelo de dados relacional, classes da camada de validação negocial e classes para persistência e busca de informações no banco de dados.

O **link** do vídeo de apresentação da PoC e do repositório onde encontra-se o código-fonte, diagramas, roteiros, etc., sobre o protótipo funcional está descrito no apêndice deste documento.

6. Avaliação da Arquitetura (ATAM)

Nesta seção é apresentada a avaliação da arquitetura desenvolvida para o Sistema de Gestão de Ocorrência (SGO), segundo o método *Architecture Trade-off Analysis Method* (ATAM).

6.1. Análise das abordagens arquiteturais

A proposta arquitetural tem como características componentes modulares cujos atributos de qualidade são apresentados através dos cenários para análise referentes aos requisitos informados na seção 3.3.

Atributos de Qualidade	Cenários	Importância (B/M/A)*	Complexidade (B/M/A)*
Segurança	Cenário 1: O sistema deve prover segurança no acesso à APIs privadas, exigir devida autenticação e autorização do usuário para acesso a funcionalidades que não são públicas.	A	A
Desempenho	Cenário 2: O sistema deve prove boa performance nas operações funcionais, seguir padrão temporal satisfatório e agradável, não transparecer travamentos e sempre proporcionar feedback ao usuário em relação a consultas.	A	M
Padrão	Cenário 3: O sistema deve seguir padrão modular, possuir baixo acoplamento e alta coesão em suas camadas.	A	A
Confiabilidade	Cenário 4: O sistema deve ser resiliente e possuir rastreo de eventos distribuídos para detecção de gargalos ou indisponibilidade em determinadas integrações.	A	B
Usabilidade	Cenário 5: O sistema deve prover interface responsiva, aplicar um design adaptativo para melhorar a disposição do conteúdo na tela do dispositivo melhorando a experiência	M	A

	do usuário.		
Compatibilidade	Cenário 6: O sistema deve funcionar corretamente nos navegadores web modernos mais usados (Google Chrome, Microsoft Edge e Firefox).	B	B
Interoperabilidade	Cenário 7: O sistema deve seguir a restrição <i>Hypermedia As the Engine Of Application State</i> (HATEOAS) nas comunicações baseadas em REST e comunicar com sistemas de outras tecnologias.	B	A

*B=Baixa, M=Média, A=Alta.

6.2. Cenários

Cenário 1 – Segurança: Ao acessar a página pública inicial do sistema o usuário não precisa estar autenticado, porém para acessar as páginas privadas e interagir com outras funcionalidades do sistema, o usuário obrigatoriamente deve possuir uma conta registrada e estar devidamente autenticado. As APIs privadas devem ser seguras e usar o padrão *JSON Web Token* (JWT), onde um acesso direto as URL's das mesmas exigirão que um token do tipo *Bearer*, válido, temporário e com escopo bem definido seja informado no cabeçalho *Authorization* da requisição HTTP. Caso o token não seja informado ou tenha expirado, a resposta da requisição HTTP deve retornar erro 401 – *Unauthorized*. Caso o token seja válido, mas o escopo do mesmo não possua o papel de acesso permitido para acesso a uma funcionalidade em específico, a resposta da requisição HTTP deve retornar erro 403 – *Forbidden*.

Cenário 2 – Desempenho: Ao acessar a funcionalidade para registro de ocorrência e após o preenchimento do formulário de cadastro de uma nova ocorrência, o processo de registro da mesma no sistema deve estar dentro de um padrão temporal satisfatório obedecendo um tempo médio de 3 segundos. Além disso, funcionalidades que exigem consulta devem sempre informar ao usuário um feedback visual de que o sistema está realizando algum processamento, exibindo uma imagem ou texto de carregando.

Cenário 3 – Padrão: Ao haver necessidade de implantar ou escalar um módulo deve ser possível que o procedimento seja feito de forma individual e ao necessitar que ocorra uma manutenção em um módulo somente parte do sistema deve ficar indisponível temporariamente enquanto o restante do mesmo deve estar

operacional. Portanto, o padrão arquitetural do sistema deve ser o de microserviços, onde o sistema como um todo deve ser composto por uma coleção de serviços/módulos dividido de forma organizada e coesa em relação às suas capacidades funcionais, sendo independentes e com baixo acoplamento para ser altamente gerenciável e testável.

Cenário 4 – Confiabilidade: Ao acessar a URL do serviço de rastreamento de eventos distribuídos, o mesmo deve retornar métricas com dados de tempo de requisições entre outras informações pertinentes para que seja possível detectar problemas de latência ou indisponibilidade de comunicação.

Cenário 5 – Usabilidade: Ao acessar o sistema em diferentes dispositivos com diferentes dimensões, as telas devem possuir comportamento responsivo se adaptando e melhorando a disposição do conteúdo na tela sem perda de funcionalidades.

Cenário 6 – Compatibilidade: Ao acessar o sistema em diferentes navegadores web como Firefox e Google Chrome, o mesmo deve funcionar corretamente sem nenhum problema grave que comprometa suas funções principais.

Cenário 7 – Interoperabilidade: Ao interagir com qualquer API *RESTful* de do sistema, ao retorno da mesma deve seguir a restrição *Hypermedia As the Engine Of Application State* (HATEOAS) facilitando e instruindo o cliente a consumir outras funcionalidades atreladas ao contexto sem a necessidade de um conhecimento mais amplo da API.

6.3. Evidências da Avaliação

Nesta seção é apresentada as medidas registradas na coleta de dados e justificativas através de evidências para comprovar atendimento aos requisitos não-funcionais e cenários descritos na seção 6.2.

Evidência da Avaliação do Cenário 1:

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	O sistema deve possuir um padrão de segurança elevado.
Preocupação:	
O sistema deve impedir que recursos privados sejam acessados sem uma devida autenticação.	

Sistema de Gestão de Ocorrência - SGO

Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O usuário tenta acessar uma página privada e/ou API, mas sem estar autenticado no sistema.	
Mecanismo:	
Criar um serviço para gerir identidades e acessos e prover autorização através de protocolo de autenticação interoperável.	
Medida de resposta:	
Negar acesso, retornar para a página inicial para novo cadastro ou autenticação.	
Considerações sobre a arquitetura:	
Riscos:	A falta ou indisponibilidade do serviço de segurança pode prejudicar a usabilidade e confiabilidade do sistema. Comprometendo a validação de tokens de acesso e expondo funcionalidade e API privadas, além de informações sensíveis.
Pontos de Sensibilidade:	Serviço de autorização e autenticação, e pontos de extremidades com TLS (HTTPS) ativado.
Trade-off:	Não há

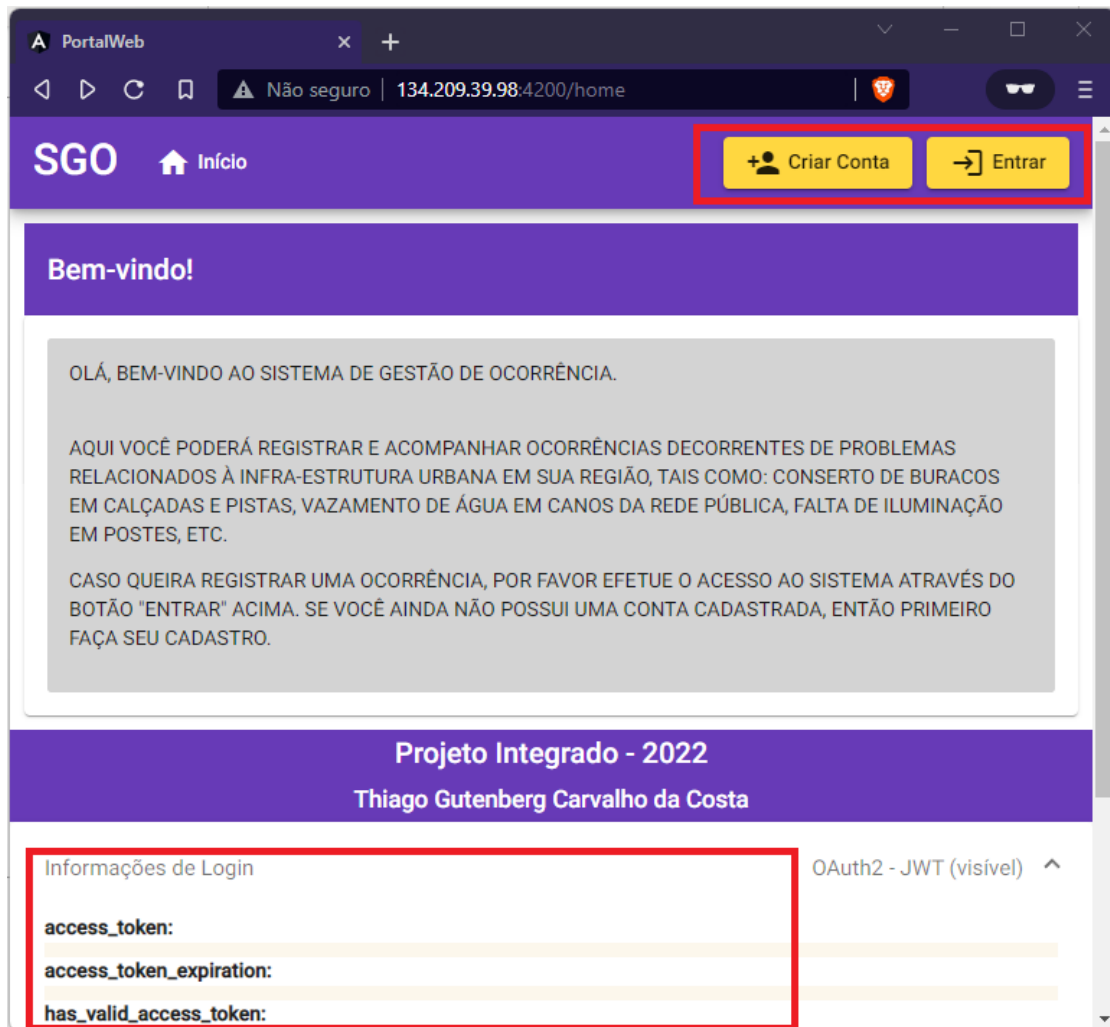


Figura 6 – Evidência 1 (Cenário 1) – Acesso a página inicial do sistema sem estar autenticado, evidenciando a exigência de autenticação inicial para acesso a funcionalidades privadas.

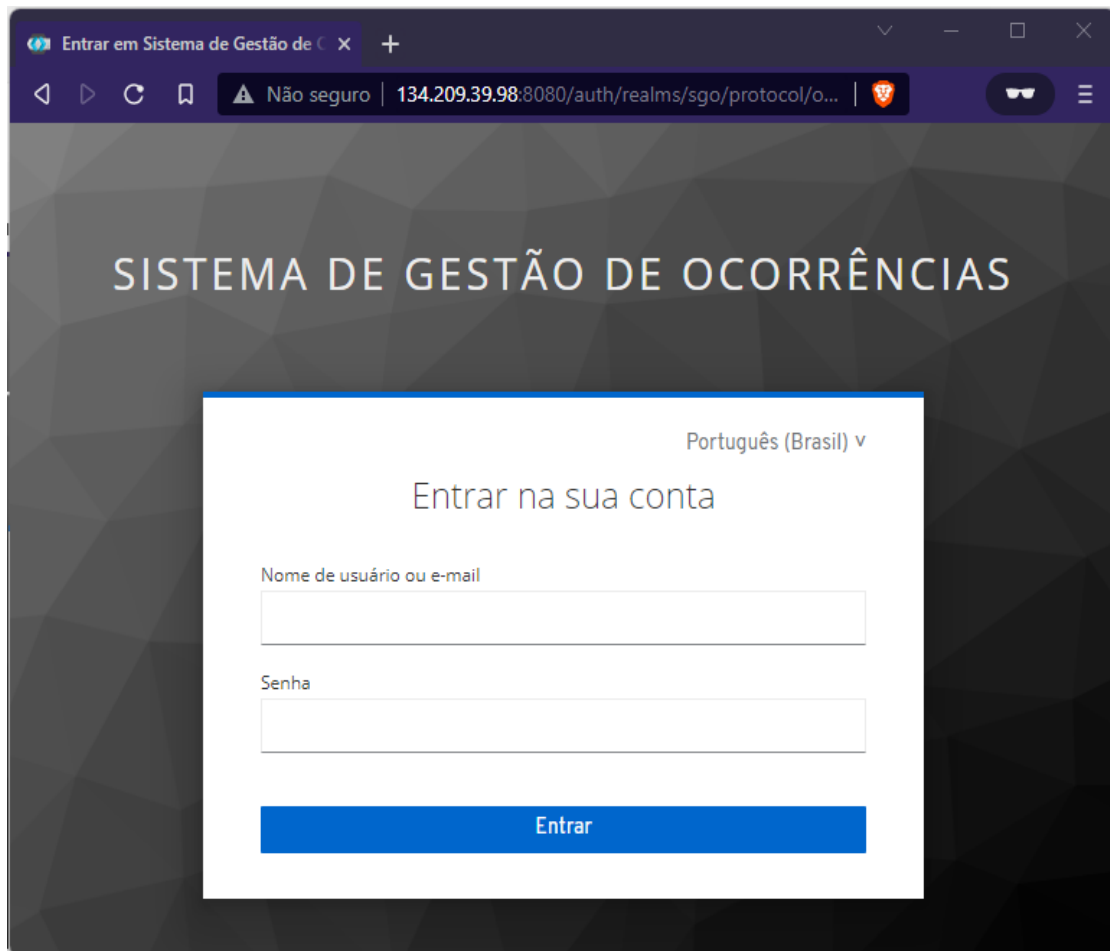


Figura 7 – Evidência 2 (Cenário 1) – Acesso a página de autenticação do sistema após clicar no botão Entrar.

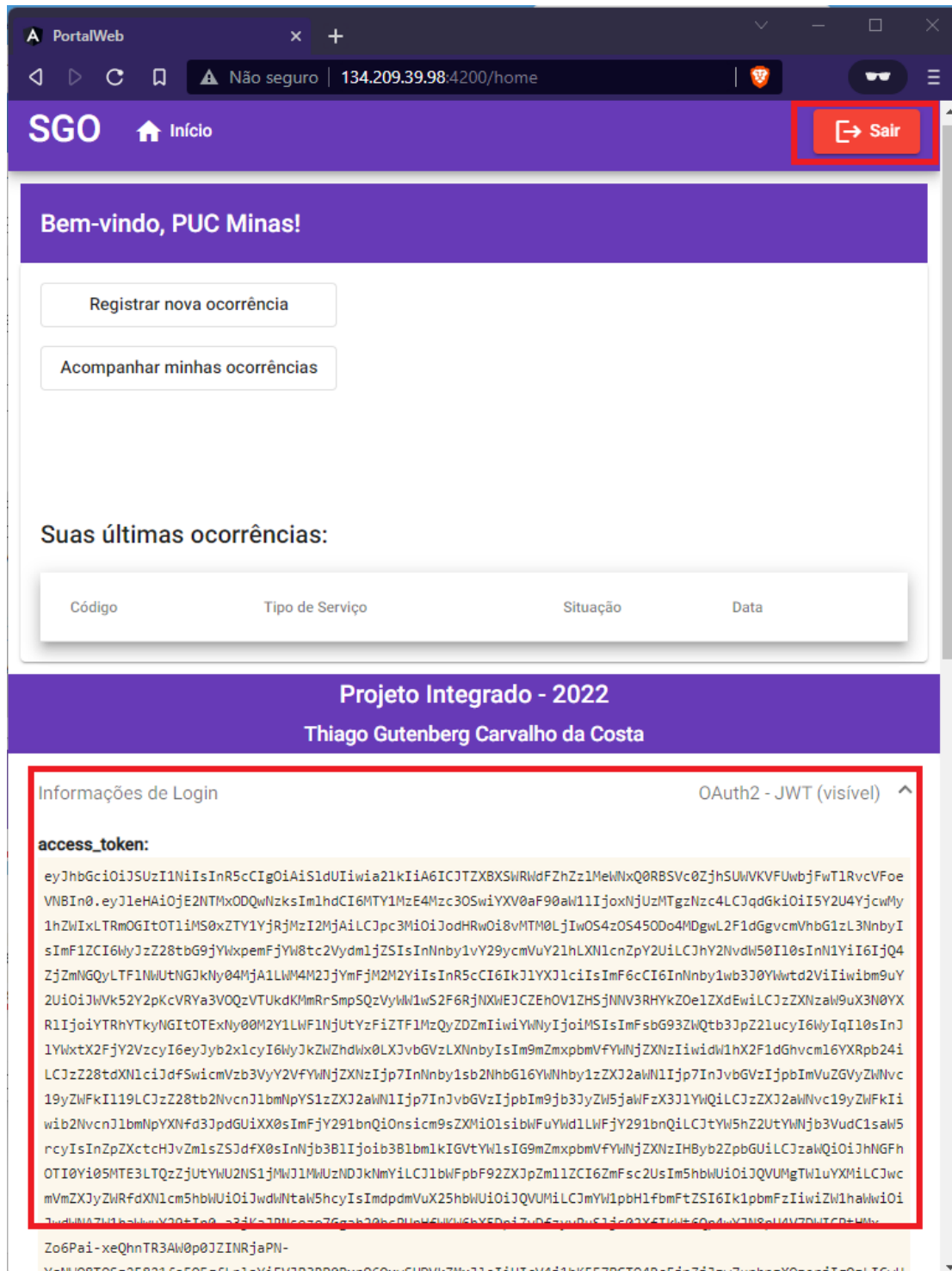


Figura 8 – Evidência 3 (Cenário 1) – Acesso a página inicial do sistema após autenticação.

Evidenciando acesso a funcionalidades privadas.

Sistema de Gestão de Ocorrência - SGO

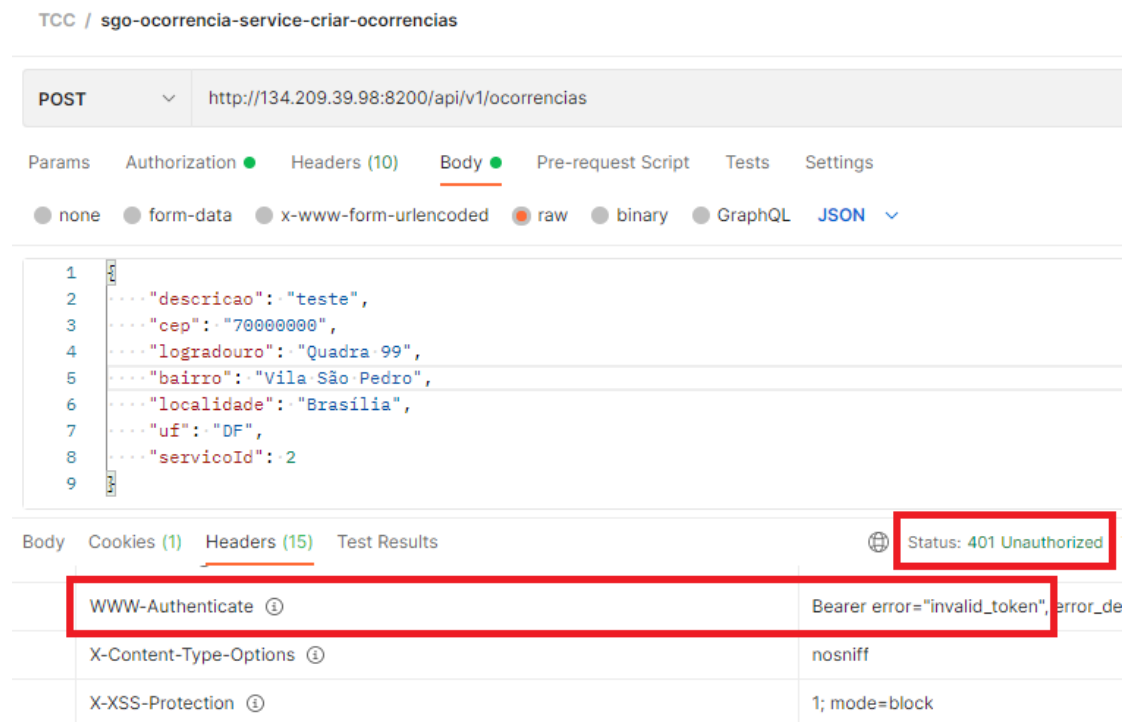


Figura 9 – Evidência 4 (Cenário 1) – Acesso não autorizado à API privada para registro de ocorrência.

Evidência da Avaliação do Cenário 2:

Atributo de Qualidade:	Desempenho
Requisito de Qualidade:	O sistema deve possuir performance de processamento satisfatório.
Preocupação:	
O sistema deve processar uma ação de forma rápida, desejável que o tempo médio seja menor ou igual a 3 segundos.	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O usuário tenta registrar uma ocorrência no sistema.	
Mecanismo:	
Criar um serviço <i>RESTful</i> para atender às requisições de cadastro de ocorrências no sistema.	
Medida de resposta:	
Retornar dados que identifique o sucesso no cadastro da ocorrência.	
Considerações sobre a arquitetura:	
Riscos:	Instabilidade relacionadas a rede podem impactar

	no tempo de processamento das requisições.
Pontos de Sensibilidade:	Alta disponibilidade através de <i>Replica-Set</i> e balanceamento de carga
Trade-off:	Não há

PortalWeb

Não seguro | 134.209.39.98:4200/ocorrencia/registrar

SGO Início Sair

Registrar Ocorrência

Serviço *
Tapa Buraco

CEP *
70297060

Estado *
DF

Cidade *
Brasília

Bairro *
Asa Sul

Endereço *
SQS 414 Bloco F

Descrição *
Buraco enorme no asfalto em frente ao bloco F que está prejudicando o transito de veículos na região.

Criar Limpar Voltar

Projeto Integrado - 2022
Thiago Gutenberg Carvalho da Costa

Informações de Login OAuth2 - JWT (escondido)

Elementos Console Fontes Rede Desempenho Memória

Preservar registro Desativar cache Sem limitação

Filtro Inverter Ocultar URLs de dados

Tudo Fetch/XHR JS CSS Img Mídia Fonte Doc WS Wasm Manifesto Outro Bloqueou os cookies

Solicitações bloqueadas Solicitações de terceiros

10ms 20ms 30ms 40ms 50ms 60ms 70ms 80ms 90ms 100ms 110ms

Gravando atividade de rede...

Faça uma solicitação ou pressione **Ctrl + R** para gravar a atualização.

[Saiba mais](#)

Figura 10 – Evidência 1 (Cenário 2) – Acesso a página de registro de ocorrência para novo cadastro de ocorrência.

The screenshot displays the SGO web application interface. At the top, a green notification box states: "Ocorrência registrada com sucesso! - Código: 46efd861-0001-bc16-0008-0180e98cc87c". Below this, the "Registrar Ocorrência" form is visible, with fields for "Serviço", "CEP", "Estado", "Cidade", "Bairro", "Endereço", and "Descrição". The form includes buttons for "Criar", "Limpar", and "Voltar".

Below the form, a purple banner reads "Projeto Integrado - 2022" and "Thiago Gutenberg Carvalho da Costa". Underneath, the "Informações de Login" section shows "OAuth2 - JWT (escondido)".

The bottom portion of the image shows a browser's developer tools network tab. A red box highlights a network request for "ocorrencias" with a status of 201, a type of xhr, and a size of 1.1 kB. A timeline chart above the table shows the request duration, with a red box highlighting the time taken, which is just under 800ms.

Figura 11 – Evidência 2 (Cenário 2) – Após clicar no botão Criar, uma nova ocorrência foi registrada com sucesso e em pouco menos que 800 milissegundos, evidenciando o bom desempenho e feedback visual do sucesso da operação através de mensagem específica.

Evidência da Avaliação do Cenário 3:

Atributo de Qualidade:	Padrão
Requisito de Qualidade:	O sistema deve possuir módulos independentes e

	autocontidos.
Preocupação:	
O sistema deve ser composto por um conjunto de serviços coesos e com baixo acoplamento entre si.	
Cenário(s):	
Cenário 3	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O mantenedor realizar a implantação dos módulos separados do sistema em <i>containers</i> Docker.	
Mecanismo:	
Fragmentar de forma organizada e coesa as funcionalidades do sistema e dividi-las em micros serviços separados e com baixo acoplamento expondo suas funções, por exemplo, através de APIs <i>RESTful</i> .	
Medida de resposta:	
Indisponibilizar apenas uma parte do sistema quando ocorrer necessidade de manutenção corretiva e/ou evolutiva de algum módulo.	
Considerações sobre a arquitetura:	
Riscos:	Apesar do baixo acoplamento entre os módulos, a indisponibilidade de algum deles pode gerar comportamentos inesperados durante processamento de fluxos complexos com várias integrações e comunicações.
Pontos de Sensibilidade:	Evolução/Versionamento de API e suporte a API legada até migração completa por todos clientes.
Trade-off:	Não há

```

Windows PowerShell
root@tcc-pucminas-arq-s

105 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

*** System restart required ***
*****

Welcome to DigitalOcean's 1-Click Docker Droplet.
To keep this Droplet secure, the UFW firewall is enabled.
All ports are BLOCKED except 22 (SSH), 2375 (Docker) and 2376 (Docker).

* The Docker 1-Click Quickstart guide is available at:
  https://do.co/3j6j3po#start

* You can SSH to this Droplet in a terminal as root: ssh root@134.209.39.98

* Docker is installed and configured per Docker's recommendations:
  https://docs.docker.com/install/linux/docker-ce/ubuntu/

* Docker Compose is installed and configured per Docker's recommendations:
  https://docs.docker.com/compose/install/#install-compose

For help and more information, visit https://do.co/3j6j3po

*****

To delete this message of the day: rm -rf /etc/update-motd.d/99-one-click
Last login: Sun May 22 03:10:53 2022 from 134.209.39.98
root@tcc-pucminas-arq-soft-dist2021-sgo:~# cd sistema-gestao-ocorrencia/
root@tcc-pucminas-arq-soft-dist2021-sgo:~/sistema-gestao-ocorrencia# docker-compose ps

```

Name	Command	State	Ports
keycloak-server	/opt/jboss/tools/docker-en	Up (healthy)	0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 8443/tcp
rabbitmq	docker-entrypoint.sh rabbitmq	Up (healthy)	15671/tcp, 0.0.0.0:15672->15672/tcp, :::15672->15672/tcp, 15691/tcp, 15692/tcp, 25672/tcp, 4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp
sgo-localizacao-service	/cnb/process/web	Up	0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
sgo-mongo	docker-entrypoint.sh mongo	Up	0.0.0.0:27017->27017/tcp, :::27017->27017/tcp
sgo-mysql	docker-entrypoint.sh mysql	Up (healthy)	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
sgo-naming-server	/cnb/process/web	Up	0.0.0.0:8761->8761/tcp, :::8761->8761/tcp
sgo-ocorrencia-service	/cnb/process/web	Up	0.0.0.0:8200->8200/tcp, :::8200->8200/tcp
sgo-portal-web	/docker-entrypoint.sh nginx	Up	0.0.0.0:4200->80/tcp, :::4200->80/tcp
zipkin-server	start-zipkin	Up (healthy)	9410/tcp, 0.0.0.0:9411->9411/tcp, :::9411->9411/tcp

```

root@tcc-pucminas-arq-soft-dist2021-sgo:~/sistema-gestao-ocorrencia#

```

Figura 12 – Evidência 1 (Cenário 3) – Separação e implantação dos módulos específicos que compõem o sistema como um todo.

Evidência da Avaliação do Cenário 4:

Atributo de Qualidade:	Confiabilidade
Requisito de Qualidade:	O sistema deve possuir métricas de comunicações distribuídas.
Preocupação:	

O sistema deve possuir um serviço que facilite a análise dos eventos de comunicação entre os serviços distribuídos.	
Cenário(s):	
Cenário 4	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Os módulos do sistema se integram e enviam amostras de métricas das requisições para um serviço específico Zipkin que centraliza as informações.	
Mecanismo:	
Criar um serviço para atender às requisições de eventos distribuídos relacionados a métricas de processamento, capturando e armazenando tais eventos.	
Medida de resposta:	
Disponibilizar de forma visual acesso aos dados através de telas de consulta.	
Considerações sobre a arquitetura:	
Riscos:	A indisponibilidade do serviço pode dificultar na investigação de problemas de integração e latência que possam ocorrer entre os serviços distribuídos.
Pontos de Sensibilidade:	Fila de mensageria como ponto de resiliência.
Trade-off:	Não há

The screenshot shows the Zipkin web interface in a browser. The address bar indicates the URL is `134.209.39.98:9411/zipkin/?lookback=2h&endTs=165...`. The interface includes a search bar with the text "Find a trace", a "RUN QUERY" button, and a "Service filters" dropdown. Below the search bar, it displays "1 Result". The result is a table with columns: Root, Start Time, Spans, and Duration. The table contains one entry for the service "ocorrencia-service" with a duration of 75.028ms. Below the table, the Trace ID is shown as "e058096d39d414d6" and a button labeled "ocorrencia-service (1)" is visible.

Root	Start Time	Spans	Duration
ocorrencia- get service: /api/v1/ocorrencias	a minute (05/22 ago 00:50:21:706)	1	75.028ms

Trace ID: e058096d39d414d6

ocorrencia-service (1)

Figura 13 – Evidência 1 (Cenário 4) – Página inicial para busca de eventos distribuídos enviados pelos módulos do sistema.

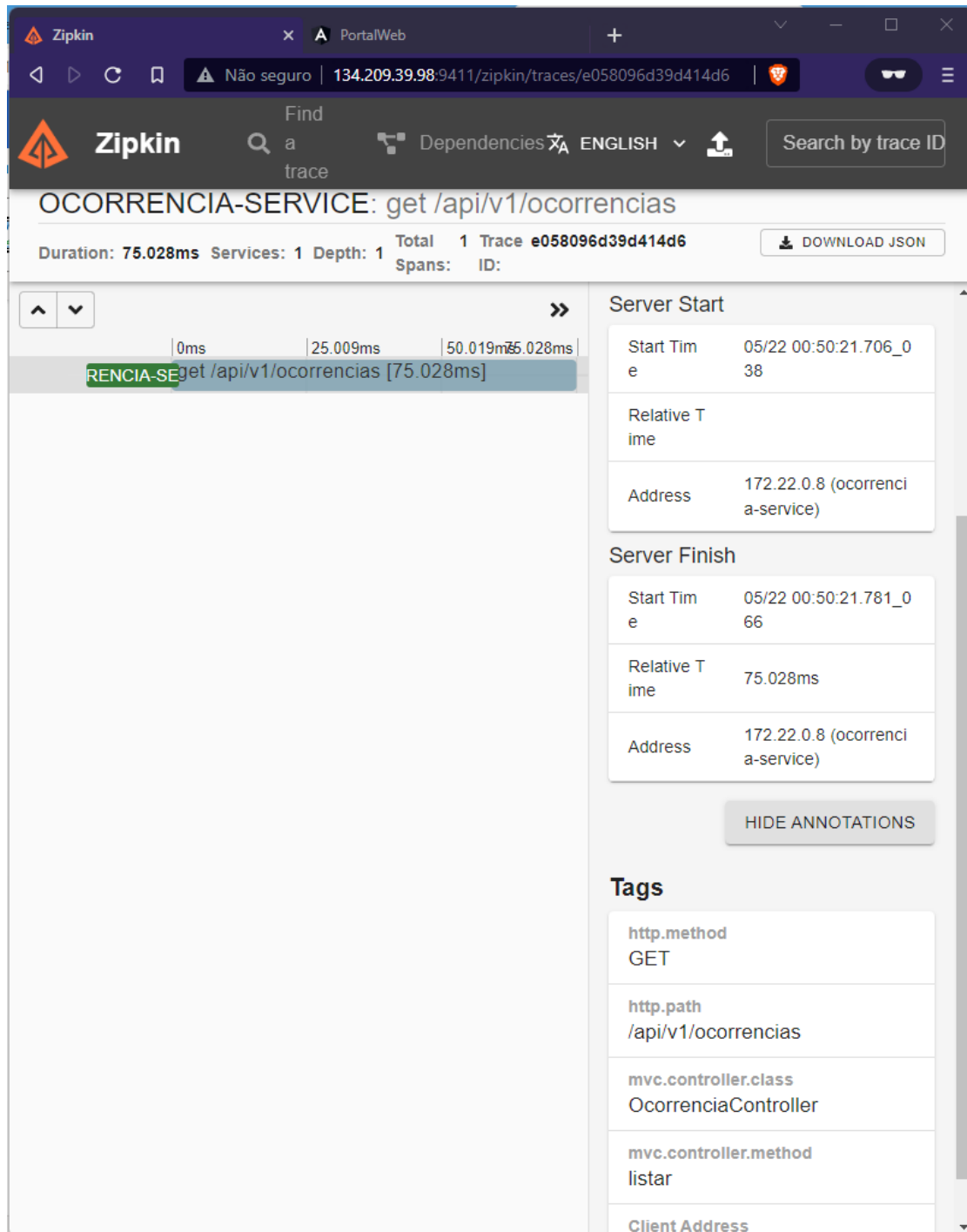


Figura 14 – Evidência 2 (Cenário 4) – Página que detalha um evento específico mostrando mais informações.

Evidência da Avaliação do Cenário 5:

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	O sistema deve se adequar ao ambiente <i>mobile</i> e <i>desktop</i> .
Preocupação:	
O sistema deve possuir design adaptativo (responsivo) quando acessado via dispositivos móveis	

Sistema de Gestão de Ocorrência - SGO

(Celular/Tablets) com resoluções menores e em dispositivos como (PC/Notebook) com resoluções maiores sem perda de funcionalidades.	
Cenário(s):	
Cenário 5	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Usuário acessando o sistema e registrando uma ocorrência.	
Mecanismo:	
Criar telas que tenham suporte ajustável e adaptativo de seus componentes visuais melhorando a disposição do conteúdo na tela.	
Medida de resposta:	
Ajustar de forma dinâmica os componentes visuais da tela em diversos dispositivos mantendo todas as funcionalidades.	
Considerações sobre a arquitetura:	
Riscos:	Podem ocorrer imprevistos na melhor disposição e adaptação do conteúdo caso o dispositivo tenha tela extremamente pequena.
Pontos de Sensibilidade:	Não há
Trade-off:	Não há

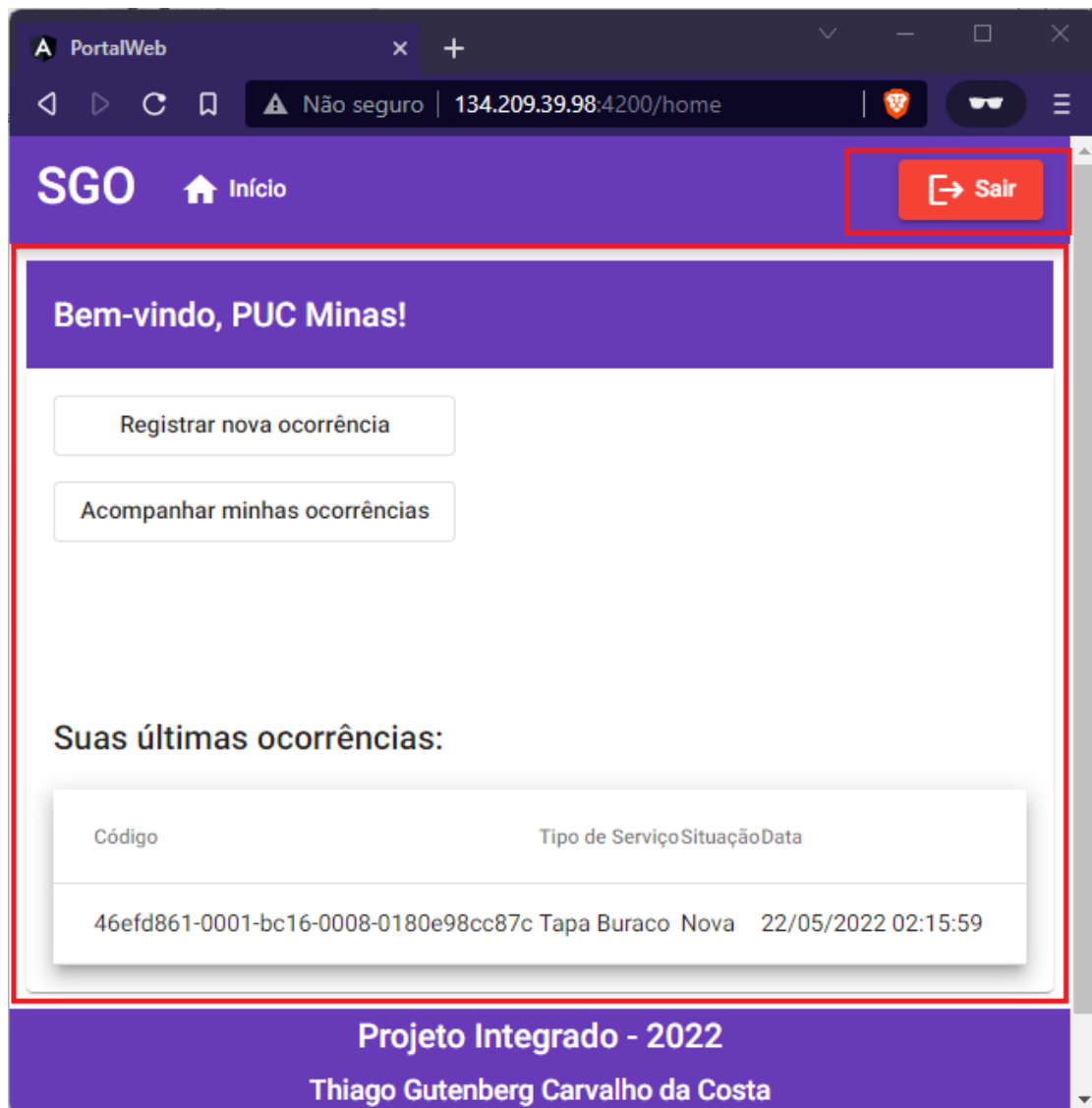


Figura 15 – Evidência 1 (Cenário 5) – Página inicial autenticada acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes padrão.

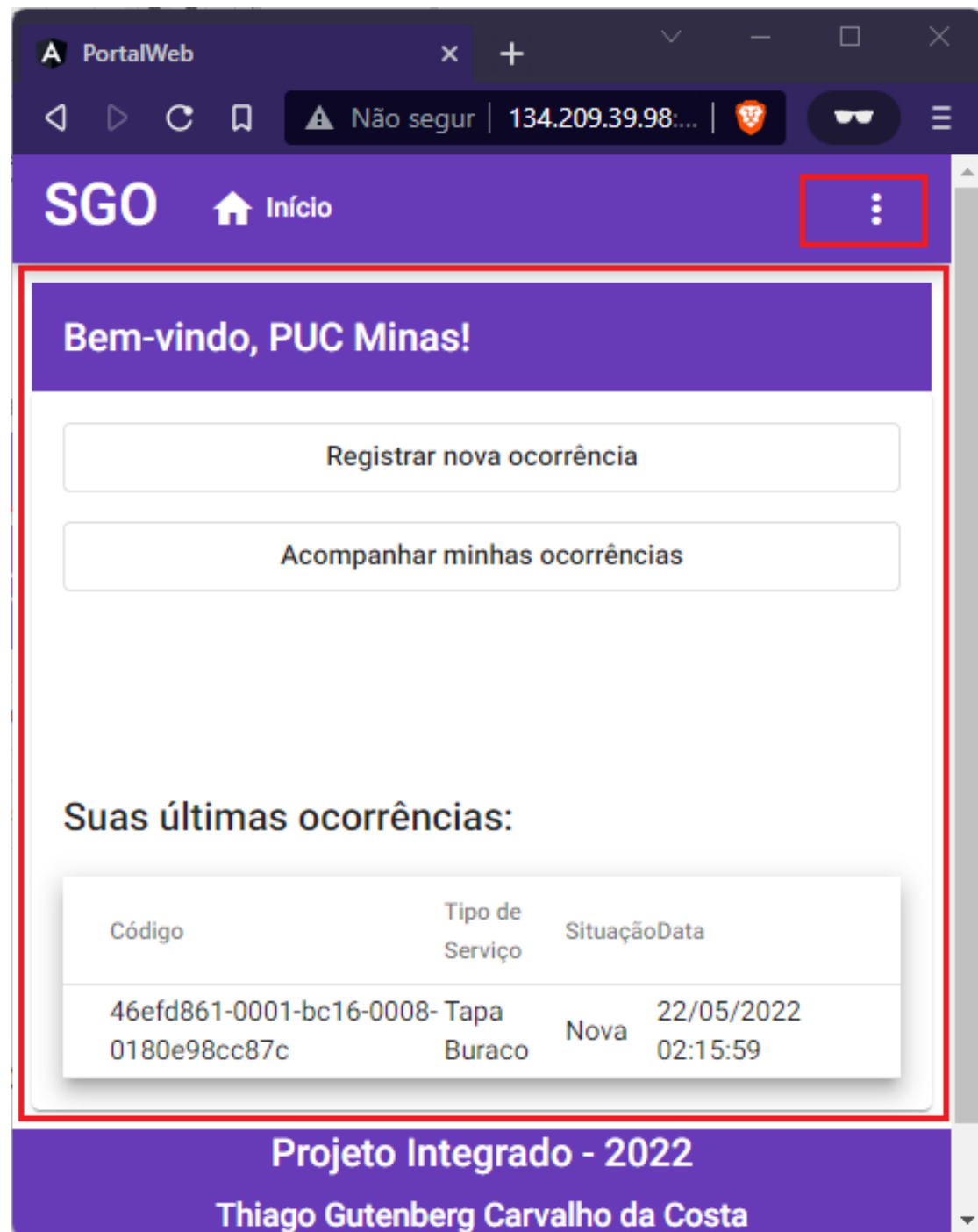


Figura 16 – Evidência 2 (Cenário 5) – Página inicial autenticada acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes, se colapsando e se readequando.

PortalWeb

Não seguro | 134.209.39.98:4200/ocorrenc... |

SGO Início Sair

Registrar Ocorrência

Serviço * CEP * Estado * Cidade *

0/8

Bairro * Endereço *

Descrição *

Criar Limpar Voltar

Projeto Integrado - 2022

Thiago Gutenberg Carvalho da Costa

Figura 17 – Evidência 3 (Cenário 5) – Página de registro de ocorrência acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes padrão.

The screenshot displays a web browser window with the address bar showing 'PortalWeb' and the URL '134.209.39.98:4200/ocorren...'. The page title is 'SGO' with a home icon and the text 'Início'. A red box highlights a menu icon (three vertical dots) in the top right corner. Below the header, a red-bordered box contains the 'Registrar Ocorrência' form. The form includes the following fields: 'Serviço *' (a dropdown menu), 'CEP *' (a text input with a '0/8' character count), 'Estado *', 'Cidade *', 'Bairro *', 'Endereço *', and 'Descrição *' (a text area with a double-slash icon). At the bottom of the form are three buttons: 'Criar' (grey), 'Limpar' (yellow), and 'Voltar' (white). The footer of the page is purple and contains the text 'Projeto Integrado - 2022' and 'Thiago Gutenberg Carvalho da Costa'.

Figura 18 – Evidência 4 (Cenário 5) – Página de registro de ocorrência acessada via navegador web em ambiente Desktop com janela redimensionada para mostrar os componentes visuais com ajustes, se colapsando e se readequando.



Figura 19 – Evidência 4 (Cenário 5) – Página inicial autenticada acessada via navegador web em ambiente Mobile mostrando componentes visuais readequados.

The image shows a mobile application interface for 'SGO' (Sistema de Gestão de Ocorrência). The top status bar shows the time as 10:43 and various connectivity icons. Below the status bar, there's a purple header with 'SGO' and a home icon. The main content area is titled 'Registrar Ocorrência' and contains several form fields: 'Serviço *', 'CEP *', 'Estado *', 'Cidade *', 'Bairro *', 'Endereço *', and 'Descrição *'. Below these fields are three buttons: 'Criar' (grey), 'Limpar' (yellow), and 'Voltar' (white). At the bottom, there's a purple footer with the text 'Projeto Integrado - 2022' and 'Thiago Gutenberg Carvalho da Costa'. Below this, there's a bottom bar with 'Login' and 'OAuth2 - JWT (escondido)'.

Figura 20 – Evidência 6 (Cenário 5) – Página de registro de ocorrência acessada via navegador web em ambiente Mobile mostrando componentes visuais readequados.

Evidência da Avaliação do Cenário 6:

Atributo de Qualidade:	Compatibilidade
Requisito de Qualidade:	O sistema deve funcionar em qualquer navegador moderno.
Preocupação:	
O sistema deve funcionar corretamente em diferentes navegadores web modernos sem comprometer suas funcionalidades.	
Cenário(s):	
Cenário 6	
Ambiente:	

Sistema em operação normal	
Estímulo:	
Usuário acessando o sistema e registrando uma ocorrência em diferentes navegadores web.	
Mecanismo:	
Criar portal web usando o framework Angular/TypeScript onde após transpilação para código nativo JavaScript/CSS o mesmo seja compatível com a maioria dos navegadores web modernos.	
Medida de resposta:	
Disponibilizar site com suas capacidades funcionais e visuais funcionando corretamente independentemente do navegador web utilizado para acesso.	
Considerações sobre a arquitetura:	
Riscos:	Podem ocorrer imprevistos funcionais do site caso o mesmo seja acessado por navegadores muito antigos, legados e descontinuados. Por exemplo: Internet Explorer.
Pontos de Sensibilidade:	Não há
Trade-off:	Não há

The screenshot displays a web application interface for registering an occurrence. The browser window shows the URL '134.209.39.98:4200/ocorrencia/...'. The page features a purple header with the logo 'SGO' and navigation links 'Início' and 'Sair'. The main section, titled 'Registrar Ocorrência', contains a form with the following fields: 'Serviço *' (a dropdown menu), 'CEP *' (with a '0/8' character count), 'Estado *', 'Cidade *', 'Bairro *', 'Endereço *', and 'Descrição *' (a text area). Below the form are three buttons: 'Criar' (disabled), 'Limpar' (highlighted in yellow), and 'Voltar'. The footer is purple and includes the text 'Projeto Integrado - 2022' and 'Thiago Gutenberg Carvalho da Costa'. At the very bottom, a status bar shows 'Informações de Login' and 'OAuth2 - JWT (escondido)'.

Figura 21 – Evidência 1 (Cenário 6) – Página de registro de ocorrência acessado via navegador Microsoft Edge.

The screenshot displays the SGO web application interface. At the top, a purple header bar contains the 'SGO' logo, a home icon labeled 'Início', and a red 'Sair' button. Below this is a section titled 'Registrar Ocorrência' with a form for recording incidents. The form includes fields for 'Serviço *' (a dropdown menu), 'CEP *', 'Estado *', 'Cidade *', 'Bairro *', 'Endereço *', and 'Descrição *' (a text area). A '0/8' character count is visible below the CEP field. At the bottom of the form are three buttons: 'Criar' (disabled), 'Limpar' (highlighted in yellow), and 'Voltar'. Below the form is a purple footer bar with the text 'Projeto Integrado - 2022' and 'Thiago Gutenberg Carvalho da Costa'. At the very bottom, a status bar shows 'Informações de Login' on the left and 'OAuth2 - JWT (escondido)' with a dropdown arrow on the right.

Figura 22 – Evidência 2 (Cenário 6) – Página de registro de ocorrência acessado via navegador Mozilla Firefox.

Figura 23 – Evidência 3 (Cenário 6) – Página de registro de ocorrência acessado via navegador Brave.

Evidência da Avaliação do Cenário 7:

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve se comunicar com outros sistemas.
Preocupação:	
O sistema deve utilizar padrão de comunicação <i>RESTful</i> seguindo restrição <i>HATEOAS</i> para que possa interagir com outros sistemas independentemente da tecnologia usada na codificação.	
Cenário(s):	
Cenário 7	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Cliente interage com a API enviando uma solicitação HTTP para listagem de ocorrências.	
Mecanismo:	
Criar um serviço <i>RESTful</i> para atender às requisições do cliente para o contexto de ocorrências.	
Medida de resposta:	
Retornar os dados requisitados no formato HAL+JSON.	

Considerações sobre a arquitetura:	
Riscos:	Podem ocorrer falha de comunicação ou lentidão de resposta caso exista alguma instabilidade de rede.
Pontos de Sensibilidade:	Não há
Trade-off:	Não há

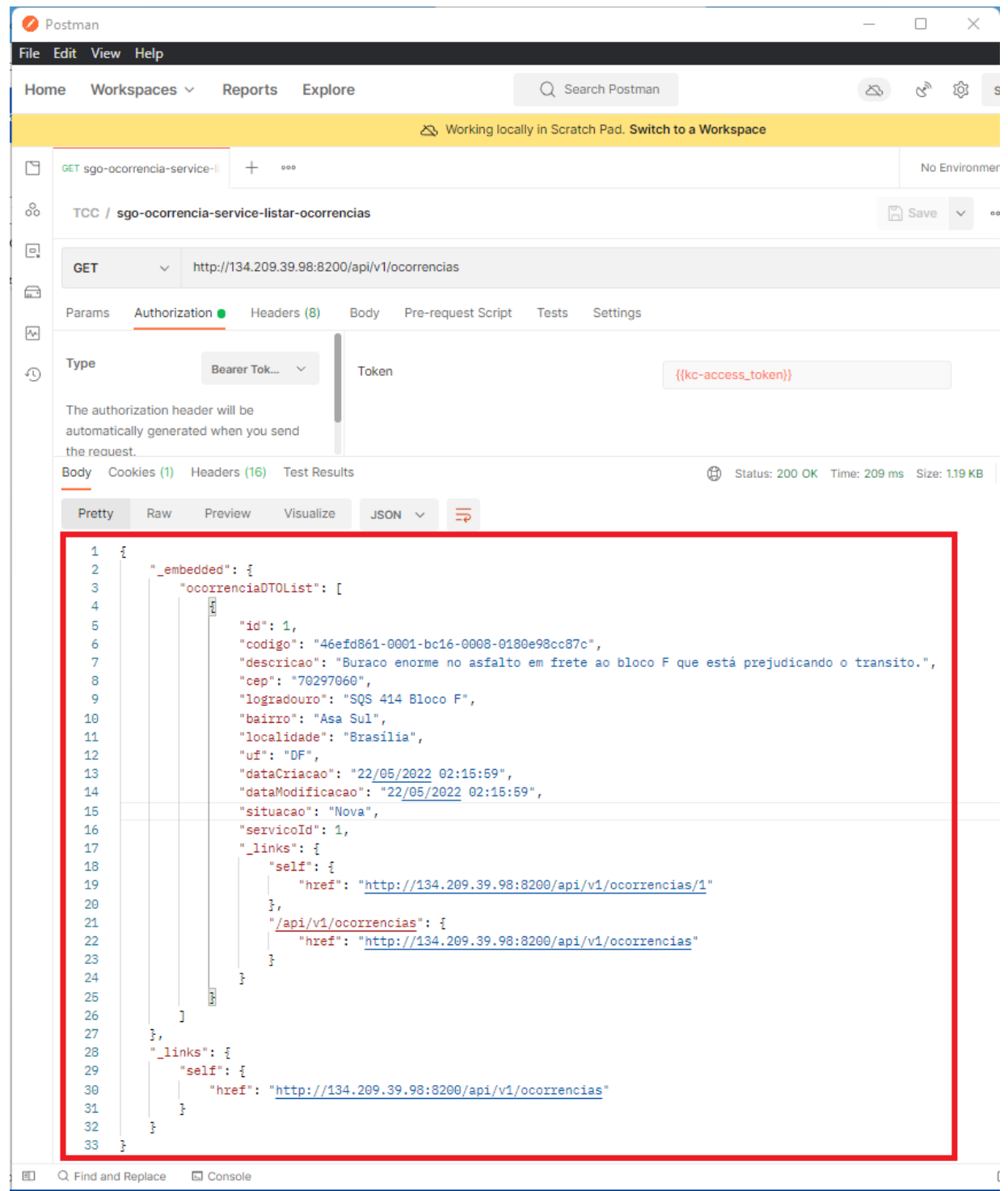


Figura 24 – Evidência 1 (Cenário 7) – Retorno da API de listagem de ocorrências com dados estruturados seguindo restrição HATEOAS.

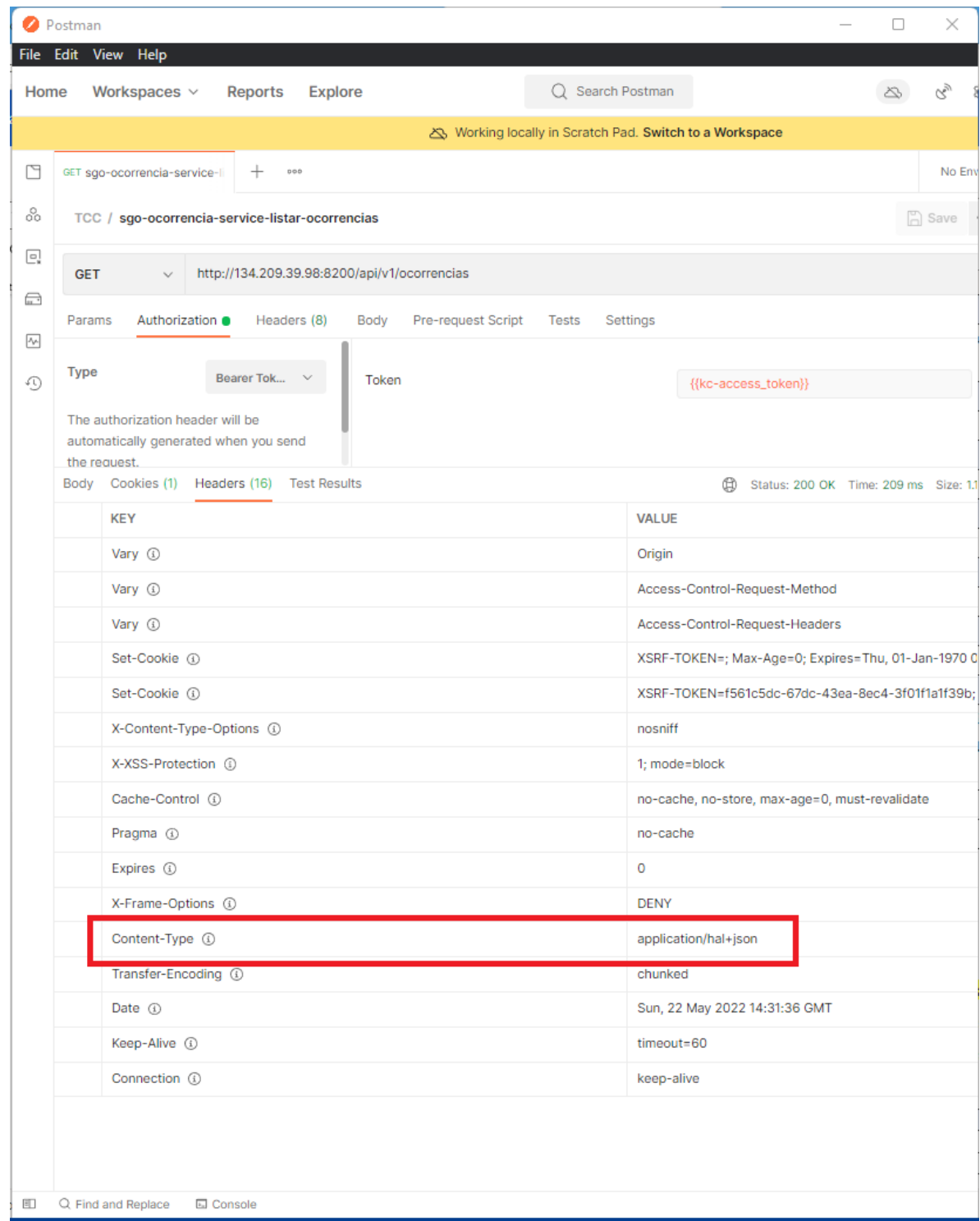


Figura 25 – Evidência 2 (Cenário 7) – Retorno da API de listagem de ocorrências com cabeçalho HTTP (Content-Type) de resposta informando o valor `application/hal+json` seguindo restrição HATEOAS.

6.4. Resultados Obtidos

Nesta seção são apresentados através de uma tabela, os resultados da arquitetura produzida para o sistema, onde os requisitos não-funcionais planejados inicialmente foram todos testados e homologados, considerando também os atributos propostos de qualidade verificados após a PoC.

Requisitos Não Funcionais	Teste	Homologação
RNF01: Desempenho — A operação de salvar uma ocorrência tem que ser rápida, não podendo exceder mais do que 3 segundos em média para ser executada.	OK	OK
RNF02: Usabilidade — A interface gráfica deve ser responsiva aplicando um design adaptativo para melhorar a disposição do conteúdo na tela do dispositivo melhorando a experiência do usuário.	OK	OK
RNF03: Compatibilidade — O sistema deverá funcionar corretamente nos navegadores web modernos mais usados (Google Chrome, Edge e Firefox).	OK	OK
RNF04: Segurança — O acesso a APIs privadas precisa ser seguro, exigindo a devida autenticação e autorização do usuário, caso contrário resultará respectivamente nos erros HTTP 401 - (<i>Unauthorized</i>) ou HTTP 403 (<i>Forbidden</i>).	OK	OK
RNF05: Padrão — O software deverá ser orientado a camadas possuindo baixo acoplamento e alta coesão seguindo o princípio de Responsabilidade Única para prover melhor manutenibilidade e evolução.	OK	OK
RNF06: Confiabilidade — Rastreamento distribuído para coleta e pesquisa de dados de tempo de requisições para detectar possíveis problemas de latência na arquitetura de serviços deverá ter um percentual de amostragem de 100%.	OK	OK

7. Avaliação Crítica dos Resultados

Nesta seção são apresentados através de um quadro resumo, os principais pontos positivos e negativos da arquitetura proposta, esclarecendo também pontos de melhorias para entendimento das limitações arquiteturais, e os prós e contras das tecnologias utilizadas.

Ponto avaliado	Descrição
Padrão arquitetural	<p>A arquitetura de um sistema deve ser pensada para um propósito, onde não existe uma que seja perfeita o suficiente para todas as situações, ou na qual resolverá todos os desafios. Dito isso e, analisando os requisitos arquiteturais propostos e os requisitos não-funcionais idealizados inicialmente para o Sistema de Gestão de Ocorrência, o padrão arquitetural orientado à microserviços foi o que melhor se adequou as necessidades, ciente e levando em consideração alguns pontos abaixo:</p> <p>Prós:</p> <ul style="list-style-type: none"> - Baixo acoplamento a outros serviços, permitindo que a equipe trabalhe independentemente utilizando a tecnologia que mais conhece e sem ser impactado na maior parte do tempo por mudanças em outros serviços; - Maior independência e organização de implantação e de manutenção corretiva/evolutiva pontual; - Eliminação de dependência tecnológica a longo prazo, tendo mais liberdade para desenvolvimento de serviços novos usando tecnologia mais atuais. <p>Contras:</p> <ul style="list-style-type: none"> - Manter a consistência dos dados pode ser mais complexa em uma arquitetura distribuída; - Maior complexidade na gestão e organização das comunicações entre os serviços através de versionamento de APIs; - Identificar e lidar com falhas parciais ou indisponibilidades totais de alguns serviços que pode atrapalhar um fluxo/processamento negocial.
API Gateway	Um ponto de melhoria na arquitetura atual seria o uso de um API

	<p><i>Gateway</i>. Levando em consideração a arquitetura distribuída de serviços que compõem o sistema como um todo, naturalmente existe uma granularidade das APIs com endereços e portas diferentes para cada API, o que pode dificultar o mapeamento, obtenção e consolidação de informações sobre determinado assunto. A implementação de um API <i>Gateway</i> seria interessante para resolver esse problema, criando um ponto centralizado e único, como um proxy reverso agregando informações para os clientes.</p>
Soluções em nuvem	<p>Nenhuma solução pronta em nuvem foi utilizada na arquitetura proposta digamos que por falta de conhecimento ou não querer ficar acoplado à um fornecedor e também porque não houve necessidade, como por exemplo para armazenamento de dados e serviço de gerenciamento de acesso e identidade, onde foram usadas soluções open-source, mais conhecidas e mais flexíveis, como MySQL e Keycloak respectivamente, mas o fato de não utilizar soluções prontas em nuvem não é necessariamente um ponto negativo, nesse caso sendo neutro, pois computação/soluções em nuvem também têm seus prós e contras.</p>
Containerização dos serviços	<p>Um ponto positivo e interessante na arquitetura proposta foi o uso da abordagem de empacotamento dos serviços como uma imagem (Docker) onde os serviços podem ser implantados como container, e de uma vez só facilmente através do Docker Compose. Uma das vantagens disso é que cada serviço é autocontido podendo ser composto de outros serviços exclusivamente necessários para um correto funcionamento operacional do mesmo, melhorando a disponibilidade e <i>throughput</i>, além da escalabilidade, manutenibilidade, etc. E uma pequena desvantagem talvez seria a falta de compatibilidade da imagem entre plataformas no lado servidor (Windows/Linux).</p>
<i>Frontend</i>	<p>A camada <i>frontend</i> feita em Angular 13, foi criada totalmente desacoplada e independente do <i>backend</i> (<i>server-side web application</i>) onde toda comunicação com o servidor é feita através de APIs REST. Isso é interessante e positivo pois flexibilizou o uso de um framework web e mais popular que possui sua própria organização e modularização das camadas através de componentes. Além disso, essa abordagem facilita mudanças caso necessário codificar um outro <i>frontend</i> usando outra tecnologia.</p>

<i>Backend</i>	Os serviços de <i>backend</i> essencialmente nessa arquitetura foram planejados e implementados para expor suas funcionalidades através de APIs <i>RESTful</i> sendo um ponto positivo pois facilita a integração com clientes web ou mobile. O Spring framework foi muito utilizado pelo fato de ser bem conhecido, com funcionalidades prontas para produção e propiciar fácil desenvolvimento de microserviços, mas essa arquitetura proposta não impede que outras tecnologias sejam utilizadas.
----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

8. Conclusão

Este trabalho apresentou um projeto arquitetural e implementação de uma prova de conceito de um Sistema de Gestão de Ocorrência (SGO) para manutenção em estruturas públicas. Através deste trabalho compreende-se com mais clareza as etapas que envolvem a idealização, definição documental, implementação arquitetural e implantação de um software.

Todo o processo foi extremamente importante para o aprendizado, possibilitando desenvolver um senso crítico e ter opinião sólida sobre propostas de soluções arquiteturais. A oportunidade de experimentar e aprender novas tecnologias também foi de grande valia, ajudando a compreender melhor os desafios que envolvem o trabalho com *frontend*, *backend* e DevOps.

Avaliando o protótipo elaborado se faz necessário evidenciar possíveis melhorias arquiteturais, tais como a implementação de um API Gateway e a implantação do sistema distribuído em um cluster Kubernetes para um controle mais adequando dos serviços. Além disso, em relação às melhorias funcionais já listadas na seção de requisitos funcionais, vale destacar a implementação de novas funções de gestão do perfil do usuário autenticado, permitindo alteração de senha, e-mail, etc.

Portanto, pelo apresentado conclui-se que os objetivos pretendidos com a elaboração deste trabalho foram alcançados e para que haja aplicabilidade em um contexto organizacional, aconselha-se alterações para adequação ao real contexto e conjuntura em que a companhia esteja inserida.

Referências

BROWN, Simon. **O modelo C4 de documentação para Arquitetura de Software**. Tradução de: Marcelo Costa. InfoQ, São Paulo, 01, ago. de 2018. Arquitetura. Disponível em: <https://www.infoq.com/br/articles/C4-architecture-model>. Acesso em: 13, fev. 2022.

PLANTUML. **Drawing UML with PlantUML**: PlantUML Language Reference Guide. Brest: PlantUML, 2021 p. 390. Disponível em: <https://plantuml.com/guide>. Acesso em: 14, fev. 2022.

MIHALCEA, Vlad. **High-Performance Java Persistence**. Victoria: Leanpub, 2021.

RAIBLE, Matt. **The Angular Mini-Book**. Toronto, Canadá: C4Media, 2022. *E-book*. Disponível em: <https://www.infoq.com/minibooks/angular-mini-book-v2/>. Acesso em: 10, abr. 2022.

SPRING, Guides. **Building REST services with Spring**. Disponível em: <https://spring.io/guides/tutorials/rest/>. Acesso em: 20 fev. 2022.

BOUCHERON, Brian. **Initial Server Setup with Ubuntu 20.04**. New York, NY: DigitalOcean, 23, abr. de 2020. Disponível em: <https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04>. Acesso em: 13, abr. 2022.

Apêndice

URL do vídeo de apresentação da Etapa 1: <https://youtu.be/GZdCEQZyvSQ>.

URL do repositório do protótipo funcional: [Github - Sistema de Gestão de Ocorrência](#).

URL do vídeo de apresentação do trabalho de forma completa (PoC): <https://youtu.be/yafyIPz1Nc4>.

URL do protótipo funcional disponível na Nuvem: <http://134.209.39.98:4200/>.
Usuário: pucminas. Senha: 123456