

Levantamento de Requisitos do Software

Referências:

IEEE Std. 830 – 1993. IEEE Recommended Practice for Software Requirements Specifications.

Filho, W.P.P. Engenharia de Software: Fundamentos, Métodos e Padrões. LTC: Rio de Janeiro, 2001.

INTEGRANTES DO GRUPO

Arthur Souza Lima (RA: 12723112788)

Aurea Reis (RA: 1272313156)

Alfredo Ruas Neto (RA: 1272326487)

Daniel Santos Lopes (RA: 12723130024)

Enrique Silva dos Reis (RA: 12724149018)

Elvis Oliveira dos Reis (RA: 1272318921)

Edioelson Júnior A. B. Teixeira (RA: 1272318423)

Gabriel Silva Magalhães (RA: 1272313274)

Gabriel Moreira de Oliveira (RA: 1272311981)

Hanspeter Dietiker (RA: 1272313332)

Natan Oliveira da Silva (RA: 12723211400)

Introdução

➤ Objetivo

O objetivo deste documento é fornecer uma descrição detalhada dos requisitos do sistema de gerenciamento de restaurantes, incluindo suas funcionalidades, restrições e requisitos de autenticação. Este documento destina-se ao grupo de desenvolvedores do trabalho da A3, das disciplinas Gestão e qualidade de software e Modelos, métodos e técnicas da engenharia de software. Este documento também tem como objetivo descrever as principais decisões de projeto tomadas pela equipe de desenvolvimento e os critérios considerados durante a tomada destas decisões. Suas informações incluem a parte de *hardware* e *software* do sistema.

➤ Escopo do produto

- O sistema de gerenciamento de restaurantes é uma aplicação web que permite aos usuários, dependendo da sua autenticação, realizar operações de CRUD (Create, Read, Update, Delete) em restaurantes, bem como avaliá-los e deixar comentários. ➤

Materiais de referência

- Para o desenvolvimento desse sistema utilizamos a metodologia ágil Kanban: <https://www.atlassian.com/br/agile/kanban> ;
- Para a implementação de commits usamos o conventional commits: <https://www.conventionalcommits.org/en/v1.0.0/>
- Para boas práticas seguimos uma linha de raciocínio parecida com o livro Código Limpo de Robert Cecil Martin: <https://www.amazon.com.br/Código-Limpo-RobertCMartinebook/dp/B085Q2K632>

➤ Visão geral deste documento

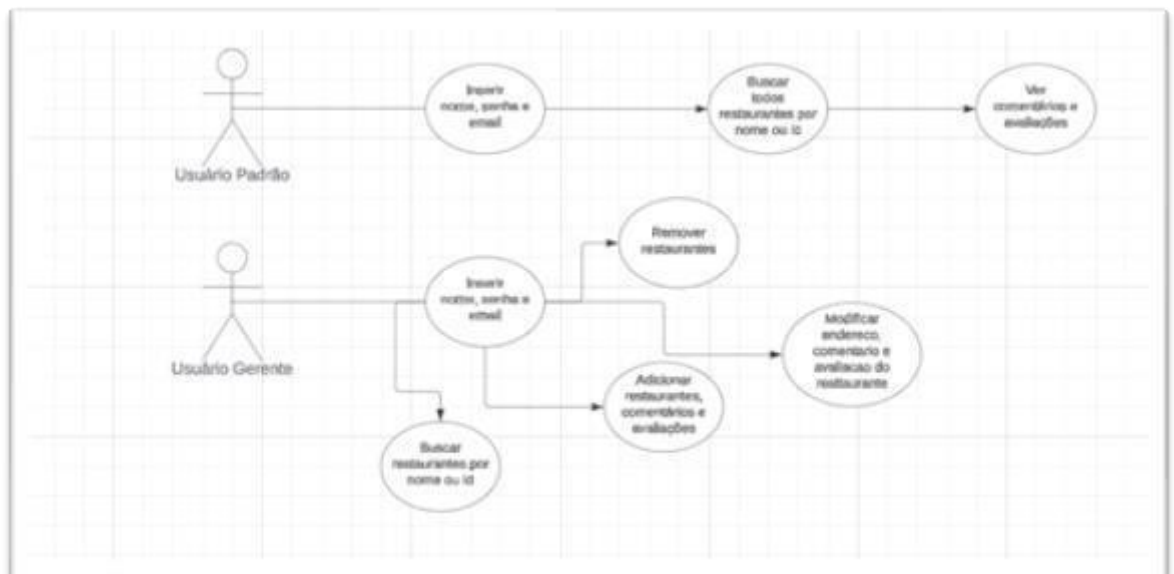
- Este documento está estruturado para fornecer uma visão detalhada dos requisitos do sistema de gerenciamento de restaurantes. Ele inclui seções que descrevem os objetivos, escopo, materiais de referência, definições e siglas, bem como uma visão geral das funcionalidades do sistema. As seções subsequentes

detalham os requisitos específicos do sistema em termos de funcionalidades, requisitos de autenticação, fluxos de trabalho e interfaces de usuário.

Descrição Geral

Perspectiva do Produto

➤ Interfaces de Usuário



➤ Interfaces de Software

- Funcionará no Windows 10;
- Preparação do ambiente com JDK Java 17, IntelliJ IDEA, Git e PostgreSQL.

➤ Requisitos de Adaptação ao Ambiente

- Preparação do ambiente com JDK Java 17, IntelliJ IDEA, Git e PostgreSQL;
- Configuração do Back End com Spring Boot, Lombok, Mockito, Hibernate, e Dev Tools;
- Utilização do Postman para testar e validar os endpoints da aplicação;
- Configuração do Front End com HTML5, CSS e VS Code;

- Automação do CI/CD com GitActions;
- Gerenciamento de projeto e colaboração com Jira Software;
- Manutenção atualizada das dependências durante o desenvolvimento do projeto.

➤ StakeHolders do Projeto

- Desenvolvedores e Equipe de Desenvolvimento: São responsáveis pela criação e implementação do sistema.
- Usuários Finais: São as pessoas que utilizarão o sistema de gerenciamento de restaurantes, incluindo proprietários de restaurantes, funcionários, e clientes
- Administradores de Restaurantes: Responsáveis pela administração e gestão dos restaurantes cadastrados no sistema.
- Proprietários do Projeto: Grupo acadêmico formado para a realização da A3.
- Clientes Potenciais: Pessoas ou organizações que podem se tornar clientes do sistema após o seu lançamento, que estejam interessados no projeto.
- Professore Acadêmico: Responsável pela avaliação do trabalho do Grupo e pelo sucesso do projeto no contexto educacional.

➤ Funções do produto

Cadastro de Restaurantes:

- Permite aos usuários adicionar novos restaurantes com detalhes como nome, endereço, avaliações e comentários;
- Requer autenticação para garantir acesso autorizado.

Consulta de Restaurante:

- Permite aos usuários buscar restaurantes específicos pelo nome ou ID, retornando detalhes completos incluindo avaliações e comentários.

Atualização de Restaurante:

- Usuários autorizados podem atualizar os detalhes de restaurantes existentes, como endereço, comentários e avaliações;
- Restrito aos administradores do sistema.

Exclusão de Restaurante:

- Capacidade para administradores excluïrem restaurantes da plataforma, exigindo autenticação e sendo uma ação irreversível.

Avaliação de Restaurantes:

- Usuários podem avaliar restaurantes em uma escala de 1 a 5 estrelas, considerando categorias como comida, ambiente e funcionários;
- Comentários opcionais podem ser deixados durante a avaliação.

Consulta de Avaliações e Comentários:

- Usuários podem visualizar avaliações e comentários deixados por outros usuários para um restaurante específico;
- Detalhes individuais de cada categoria de avaliação são disponibilizados.

Autenticação e Autorização:

- Requer autenticação para todas as ações relacionadas a adicionar, atualizar ou excluir restaurantes;
- Administradores têm acesso para gerenciar todos os restaurantes e usuários da plataforma.

➤ Características dos Usuários

Administradores:

- Têm permissões amplas para gerenciar restaurantes e usuários;
- Acesso completo a todas as funcionalidades do sistema;
- Provavelmente terão um nível mais elevado de proficiência no processo de negócio e em informática.

Usuários Padrão:

- Podem adicionar avaliações e consultar informações de restaurantes;
- Acesso restrito a certas funções de atualização e exclusão;
- Podem variar em termos de proficiência em informática, mas devem ser capazes de navegar e interagir com o sistema com facilidade.

➤ Representação Arquitetural

Aqui está uma representação arquitetural do projeto de Sistema de Gerenciamento de Restaurantes, baseada no modelo "4+1", conforme especificado:

➤ Visão de Caso de Uso (Caso de Uso)

- Público: Todos os envolvidos no projeto, incluindo desenvolvedores, analistas, integradores e gerentes.
- Área: Requisitos funcionais.
- Artefato da MDS: Realização dos Casos de Uso.

➤ Visão Lógica (Realização dos Casos de Uso)

- Público: Analistas e desenvolvedores.
- Área: Implementação da lógica de negócios.
- Artefato da MDS: Realização dos Casos de Uso.
- Descrição: Esta visão detalha a estrutura lógica do sistema, mostrando como os casos de uso são realizados por meio de componentes de software. Inclui a representação dos principais fluxos de dados e funcionalidades do sistema, como cadastro, busca, atualização e exclusão de restaurantes, além da avaliação e consulta de avaliações.

➤ Visão de Processo (Performance, Escalabilidade, Concorrência)

- Público: Integradores e desenvolvedores.
- Área: Aspectos de desempenho, escalabilidade e concorrência.
- Artefato da MDS: Não especificado.
- Descrição: Esta visão aborda aspectos relacionados ao desempenho e à eficiência do sistema, incluindo sua capacidade de lidar com um grande volume de usuários, transações e dados. Também considera questões de escalabilidade e concorrência para garantir que o sistema possa crescer e funcionar de forma eficaz em diferentes cenários.

➤ Visão de Implementação (Componentes de Software):

- Público: Programadores e desenvolvedores.
- Área: Implementação do sistema em termos de componentes de software.
- Artefato da MDS: Componentes de Software.
- Descrição: Nesta visão, são identificados e detalhados os componentes de software que compõem o sistema, incluindo módulos para a interface do usuário, lógica de negócios, acesso a dados, segurança.

➤ Visão de Implementação (Nodos físicos)

- Público: Gerência de Configuração, Administradores de Sistemas.
- Área: Implantação física e distribuição do sistema.
- Artefato da MDS: Nodos físicos.
- Descrição: Esta visão descreve a infraestrutura física necessária para hospedar e implantar o sistema, incluindo servidores, redes, bancos de dados e outros recursos de hardware e software. Também pode abordar considerações relacionadas à escalabilidade, disponibilidade e segurança da infraestrutura de implantação.

Essa representação arquitetural fornece uma visão abrangente do sistema de gerenciamento de restaurantes, abordando diferentes aspectos desde os requisitos funcionais até a implementação e implementação física do sistema.

➤ Padrões do Projeto (Desing Patterns)

Para o projeto de Sistema de Gerenciamento de Restaurantes, podem ser aplicados diversos padrões de projeto (design patterns) para garantir uma arquitetura robusta, flexível e de fácil manutenção. Aqui estão alguns padrões que utilizamos no projeto:

- **Padrão MVC (Model-View-Controller):** Este padrão é amplamente utilizado em aplicações web e separa a aplicação em três componentes principais: o Modelo (responsável pela lógica de negócios e acesso aos dados), a Visão (responsável pela apresentação dos dados ao usuário) e o Controlador (responsável por receber e processar as requisições do usuário, atualizando o Modelo e a Visão conforme necessário).
- **Padrão DTO (Data Transfer Object):** Este padrão é útil para transferir dados entre componentes da aplicação de forma eficiente. Ele encapsula os dados em objetos simples, que podem ser facilmente serializados e transportados entre diferentes camadas da aplicação ou entre sistemas distribuídos.

➤ Cronograma Projetado do Produto

O cronograma do projeto será dividido em etapas principais, incluindo análise de requisitos, design, implementação, testes e lançamento. O tempo estimado para cada etapa pode variar, mas uma projeção inicial sugere o seguinte:

- Análise de Requisitos: 1 semana
- Design e Arquitetura: 1 semana
- Implementação: 4 semanas
- Testes e Depuração: 4 semanas
- Lançamento e Entrega: 2 semanas

➤ Riscos e Considerações

Alguns dos principais riscos e considerações a serem levados em conta durante o desenvolvimento do projeto incluem:

- Complexidade dos Requisitos: Requisitos adicionais podem surgir durante o desenvolvimento, o que pode impactar o cronograma e os recursos necessários.
- Dependências Externas: O projeto pode depender de tecnologias ou serviços externos, cuja disponibilidade ou confiabilidade pode afetar o progresso do projeto.
- Mudanças de Escopo: Mudanças no escopo do projeto podem ocorrer durante o desenvolvimento, exigindo ajustes no cronograma e nos recursos.

➤ Conclusão da Estimativa do Produto

Essas estimativas fornecem uma visão preliminar do escopo, recursos necessários e cronograma projetado para o desenvolvimento do sistema de gerenciamento de restaurantes. À medida que o projeto avança, essas estimativas serão refinadas e atualizadas para refletir com mais precisão os requisitos e o progresso real do projeto.

➤ Plano de Testes

• Introdução

Este documento descreve os requisitos a testar, os tipos de testes definidos para cada iteração, os recursos de hardware e software a serem empregados e o cronograma dos testes ao longo do projeto. As seções referentes aos requisitos, recursos e cronograma servem para permitir ao gerente do projeto acompanhar a evolução dos testes.

• Requisitos a Testar

Nesta seção, são listados os requisitos a serem testados, subdivididos em casos de uso e requisitos não funcionais.

- Cadastro de Restaurantes:

Adição de Restaurantes:

Descrição: Permite aos usuários adicionar novos restaurantes com detalhes como nome, endereço, avaliações e comentários.

- **Critérios de Teste:**

- Verificar se os usuários podem adicionar restaurantes com sucesso.
- Validar se todos os detalhes obrigatórios são fornecidos durante a adição do restaurante.
- Testar se os restaurantes adicionados são salvos corretamente no sistema.

1. Autenticação de Acesso:

Descrição: Requer autenticação para garantir acesso autorizado durante a adição de restaurantes.

- **Critérios de Teste:**

- Verificar se a autenticação é necessária para adicionar um restaurante.
- Testar se apenas usuários autenticados podem adicionar novos restaurantes.
- Validar se usuários não autenticados são impedidos de adicionar restaurantes.

- Consulta de Restaurante:

1. Busca de Restaurantes:

Descrição: Permite aos usuários buscar restaurantes específicos pelo nome ou ID, retornando detalhes completos incluindo avaliações e comentários.

- **Critérios de Teste:**

- Verificar se os usuários podem buscar restaurantes pelo nome com sucesso.
- Testar se os resultados da busca incluem os detalhes completos do restaurante, incluindo avaliações e comentários.

- Atualização de Restaurante:

1. Atualização de Detalhes do Restaurante:

Descrição: Usuários autorizados podem atualizar os detalhes de restaurantes existentes, como endereço, comentários e avaliações.

- **Critérios de Teste:**

- Verificar se usuários autorizados podem atualizar os detalhes do restaurante com sucesso.
- Testar se apenas usuários autorizados têm permissão para atualizar os detalhes do restaurante.

- **Exclusão de Restaurante:**

Descrição: Capacidade para administradores excluirmos restaurantes da plataforma, exigindo autenticação e sendo uma ação irreversível.

- **Critérios de Teste:**

- Verificar se apenas administradores têm permissão para excluir restaurantes.
- Testar se a exclusão de restaurantes é uma ação irreversível e requer autenticação adequada.

- **Avaliação de Restaurantes:**

Descrição: Usuários podem avaliar restaurantes em uma escala de 1 a 5 estrelas, considerando categorias como comida, ambiente e funcionários.

- **Critérios de Teste:**

- Verificar se os usuários podem avaliar restaurantes com sucesso.
- Testar se as avaliações dos usuários são registradas corretamente no sistema.

- **Consulta de Avaliações e Comentários:**

Descrição: Usuários podem visualizar avaliações e comentários deixados por outros usuários para um restaurante específico.

- **Critérios de Teste:**

- Verificar se os usuários podem visualizar as avaliações e comentários de outros usuários para um restaurante específico.

- Testar se os detalhes individuais de cada categoria de avaliação são disponibilizados corretamente.

- **Autenticação e Autorização:**

Descrição: Requer autenticação para todas as ações relacionadas a adicionar, atualizar ou excluir restaurantes.

- **Critérios de Teste:**

- Verificar se todas as ações relacionadas a adicionar, atualizar ou excluir restaurantes exigem autenticação.
- Testar se apenas administradores têm acesso para gerenciar todos os restaurantes e usuários da plataforma.

Esta lista de requisitos a testar abrange os casos de uso identificados e os requisitos não funcionais relacionados ao sistema de gerenciamento de restaurantes.

- **Tipos de teste**

Nesta seção, descrevemos os tipos de testes escolhidos para cada iteração do projeto. Esta seleção baseia-se nos requisitos do sistema, tipo de aplicação e recursos disponíveis. Cada tipo de teste é detalhado quanto à sua técnica e critérios de completude.

Iteração 1: Testes Unitários e de Integração

- **Testes Unitários**

Objetivo: Verificar individualmente as menores partes testáveis do software, como funções e métodos.

Técnica: Utilizar mocks e stubs para isolar as unidades de código.

Ferramentas: JUnit, Mockito.

Critérios de Completude: Cobertura de pelo menos 80% do código.

Todos os métodos críticos testados com múltiplos casos de uso.

- **Testes de Integração**

Objetivo: Testar a integração entre diferentes módulos e componentes do sistema.

Técnica: Executar testes que envolvem múltiplas unidades de código em conjunto.

Ferramentas: JUnit, Spring Test.

Critérios de Completude: Testar todos os fluxos de dados principais entre módulos.

Verificar a comunicação correta entre os componentes integrados.

Iteração 2: Testes de Caixa Preta e Caixa Branca

- **Testes de Caixa Preta**

Objetivo: Validar a funcionalidade do software sem considerar a estrutura interna do código.

Técnica: Criar casos de teste baseados nos requisitos funcionais.

Ferramentas: Postman para entrada e saída de dados.

Critérios de Completude: Cobrir todos os requisitos funcionais do sistema.

Verificar a correção das saídas para um conjunto de entradas especificado.

- **Criando Usuário :**

-Entrada de dados:

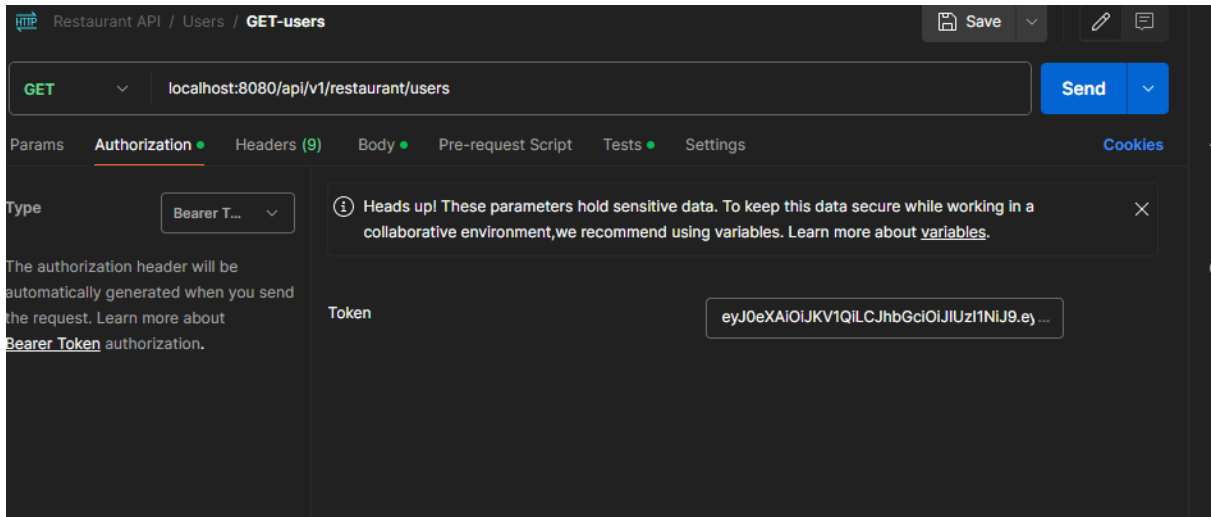
```
1 {  
2   "name": "user",  
3   "username": "user@email.com",  
4   "password": "123456"  
5 }
```

-Saída de dados:

```
Body  Cookies  Headers (14)  Test Results (1/1)  Status: 201 Created  Time: 1253 ms  Size: 496 B  Save as example  ...  
Pretty  Raw  Preview  Visualize  JSON  ...  
1 {  
2   "id": 14,  
3   "name": "user",  
4   "username": "user@email.com",  
5   "role": "DEFAULT"  
6 }
```

- **Buscando usuários:**

-Entrada de dados:

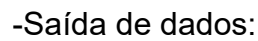


-Saída de dados:



- **Buscando usuários por id:**

-Entrada de dados:



PUT

localhost:8080/api/v1/restaurant/users/7

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

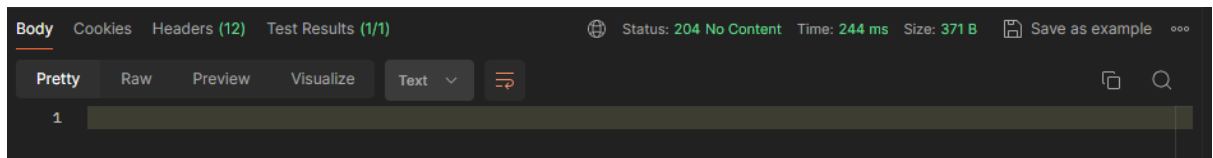
☐ GraphQL

☒ JSON

Beautify

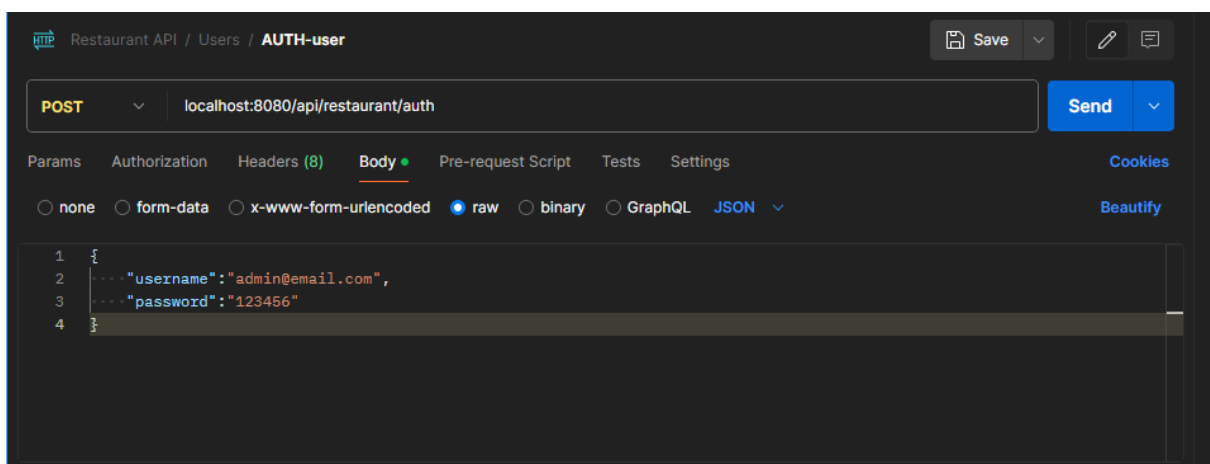
```
1 {
2   "password": "123456",
3   "newPassword": "654321",
4   "confirmPassword": "654321"
5 }
```

-Saída de dados:

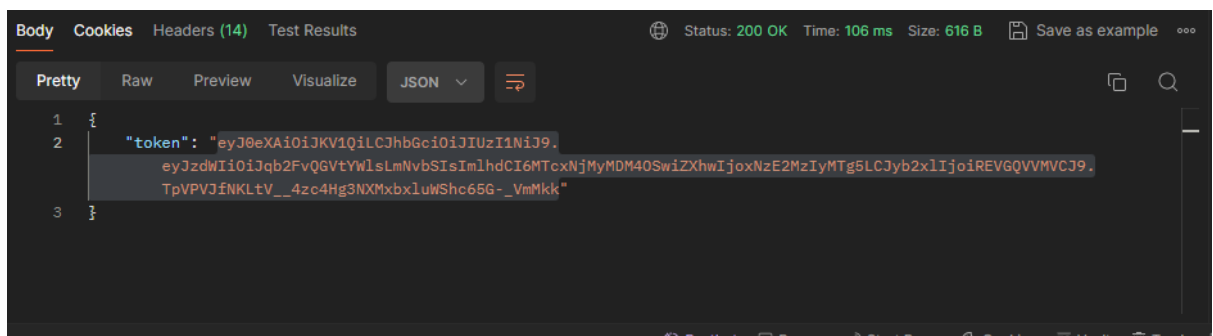


- **Autenticação de um usuário:**

-Entrada de dados:

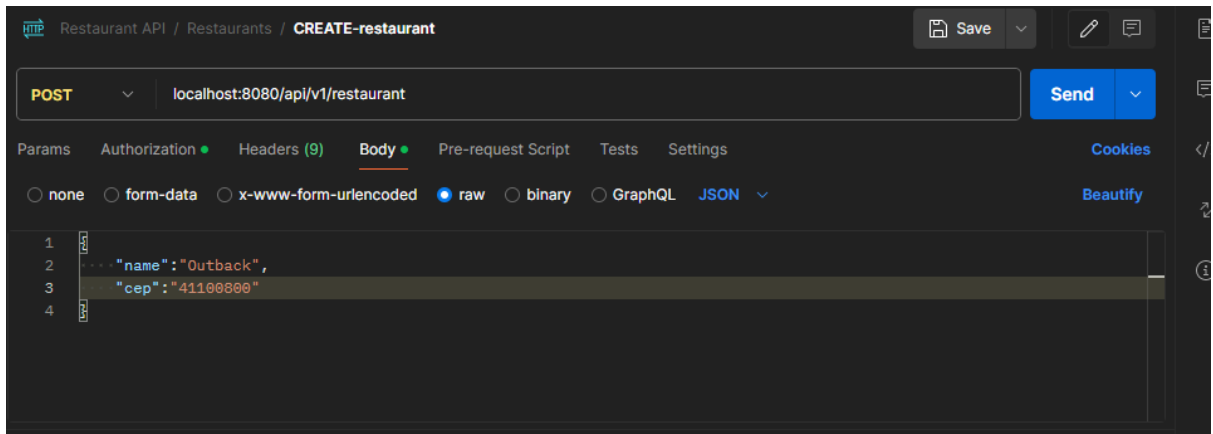


-Saída de dados:

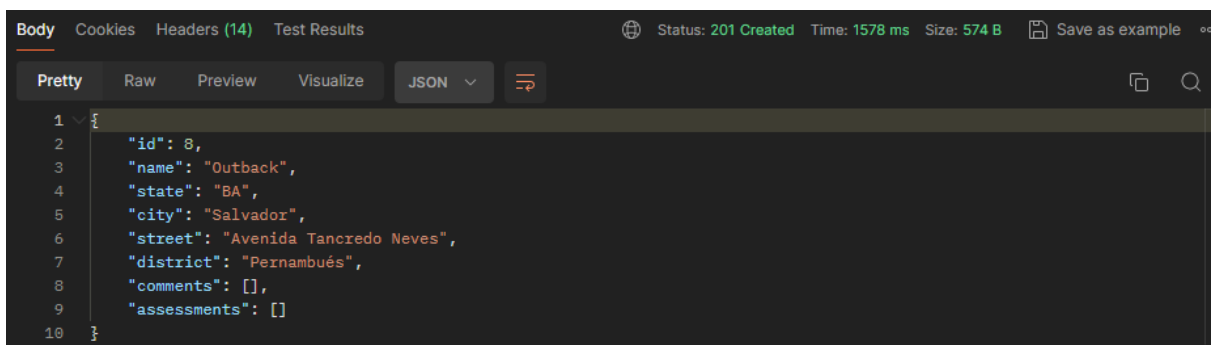


- **Criação de um restaurante**

-Entrada de dados:

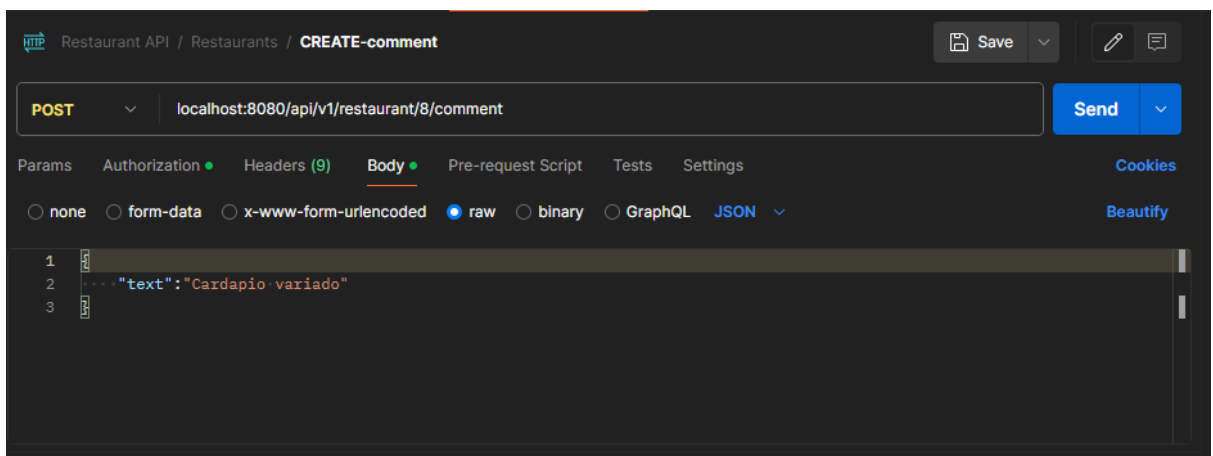


-Saída de dados:

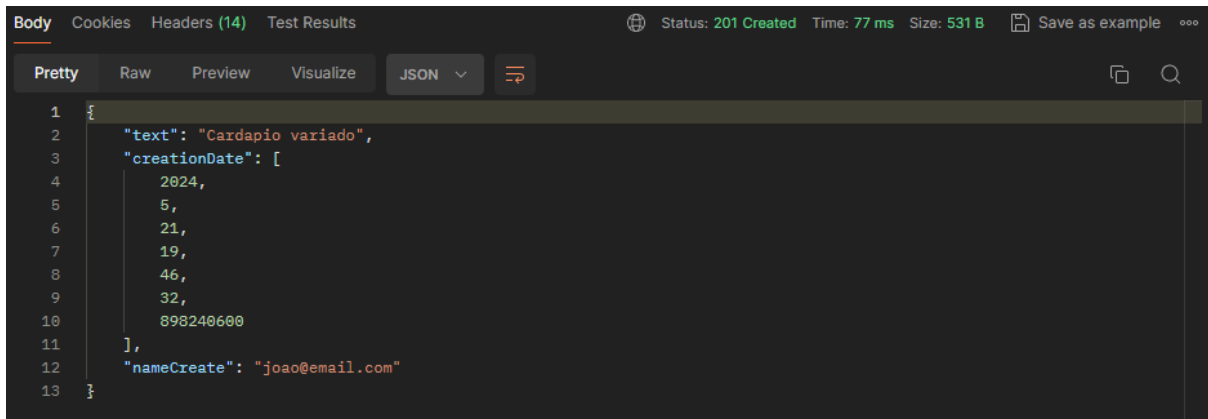


- **Comentando em um restaurante**

-Entrada de dados:



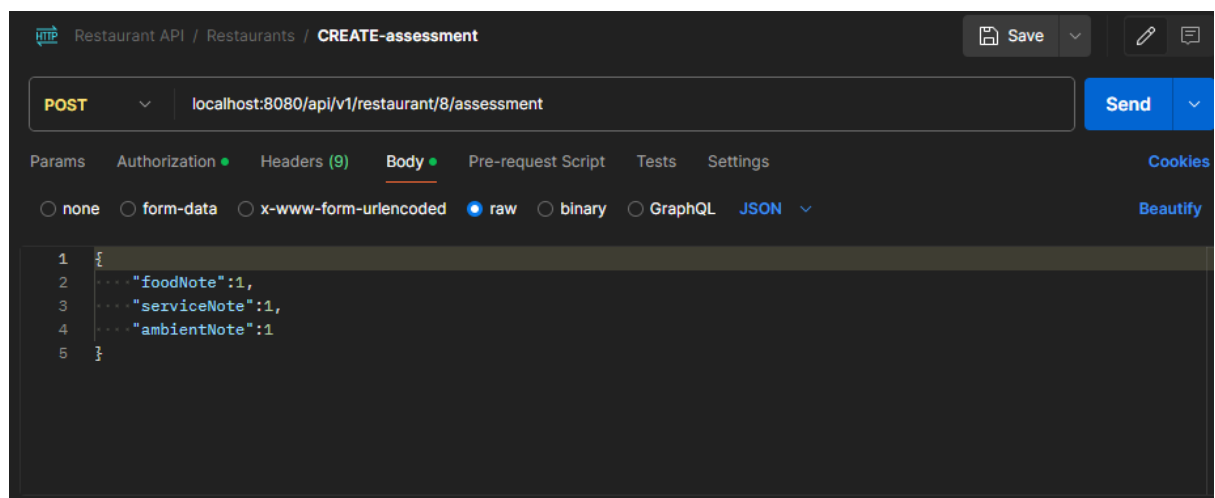
-Saída de dados:



```
1 {
2   "text": "Cardapio variado",
3   "creationDate": [
4     2024,
5     5,
6     21,
7     19,
8     46,
9     32,
10    898240600
11  ],
12   "nameCreate": "joao@email.com"
13 }
```

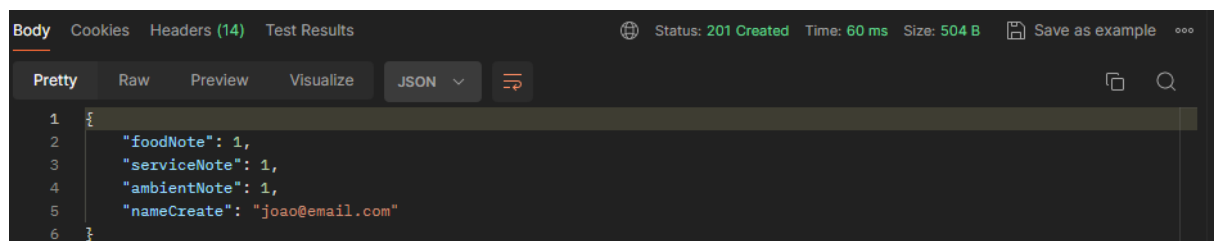
- Avaliando restaurante

-Entrada de dados:



```
1 {
2   ... "foodNote":1,
3   ... "serviceNote":1,
4   ... "ambientNote":1
5 }
```

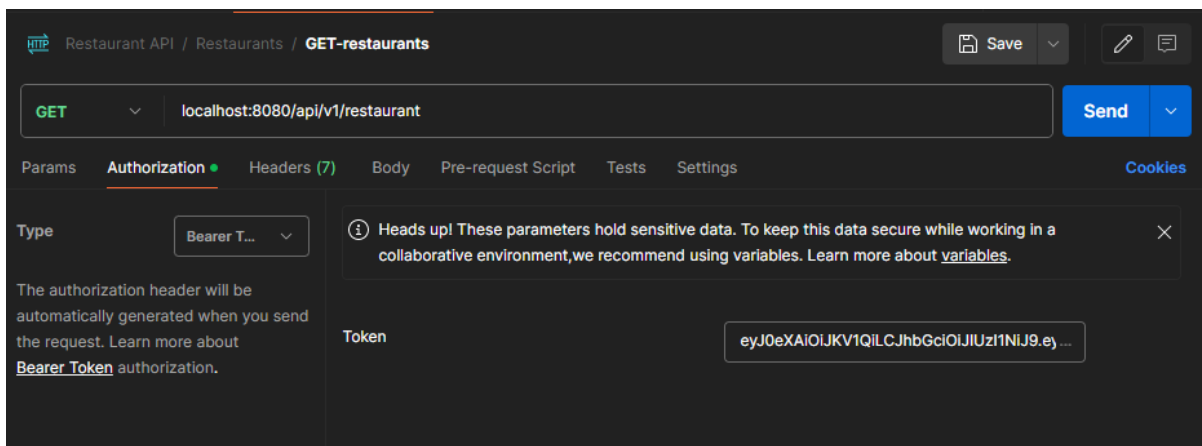
-Saída de dados:



```
1 {
2   "foodNote": 1,
3   "serviceNote": 1,
4   "ambientNote": 1,
5   "nameCreate": "joao@email.com"
6 }
```

- Buscando restaurantes

- Entrada de datos:



-Saída de dados:

```
{
  "id": 7,
  "name": "Lacerda BVhia",
  "state": "BA",
  "city": "Salvador",
  "street": "Rua Vital Soares",
  "district": "Brotas",
  "comments": [
    {
      "text": "Cardapio variado",
      "creationDate": [
        2024,
        4,
        23,
        13,
        6,
        18,
        660783000
      ],
      "nameCreate": "joao@email.com"
    }
  ]
}
```

```

{
  "id": 8,
  "name": "Outback",
  "state": "BA",
  "city": "Salvador",
  "street": "Avenida Tancredo Neves",
  "district": "Pernambués",
  "comments": [
    {
      "text": "Cardapio variado",
      "creationDate": [
        2024,
        5,
        21,
        19,
        46,
        32,
        898241000
      ],
      "nameCreate": "joao@email.com"
    }
  ]
}

```

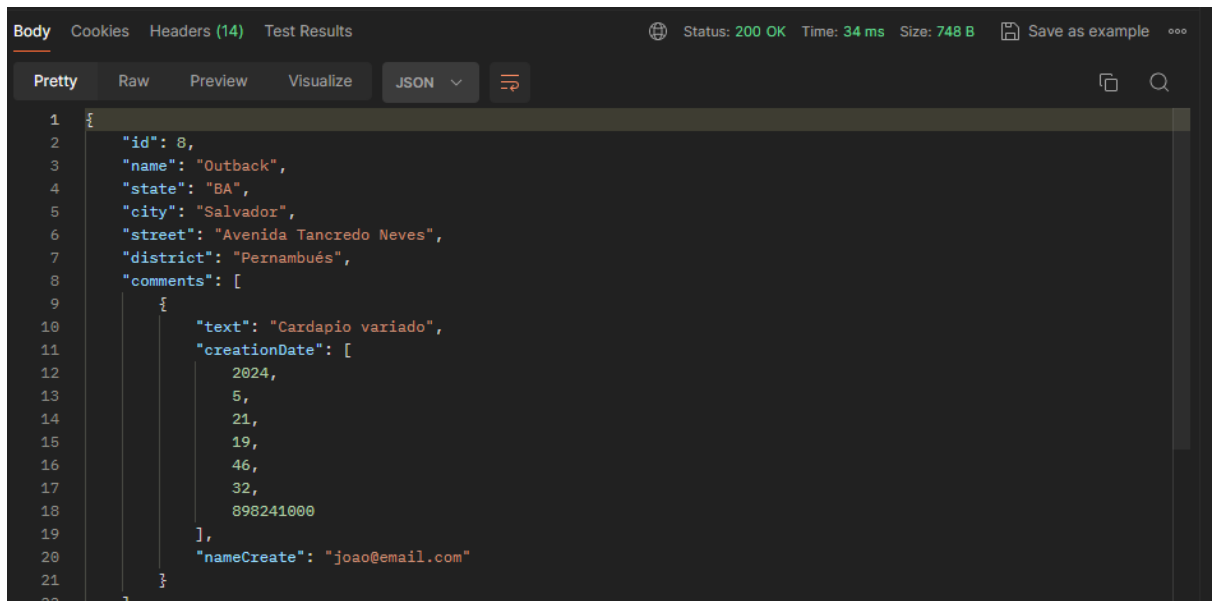
- Buscando restaurante por id

-Entrada de dados:

The screenshot shows a REST client interface with the following details:

- URL:** `localhost:8080/api/v1/restaurant/8`
- Method:** `GET`
- Authorization:** Bearer Token. The token value is `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ...`.
- Warning:** Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Saida de Dados:



```
1 {
2   "id": 8,
3   "name": "Outback",
4   "state": "BA",
5   "city": "Salvador",
6   "street": "Avenida Tancredo Neves",
7   "district": "Pernambúes",
8   "comments": [
9     {
10      "text": "Cardapio variado",
11      "creationDate": [
12        2024,
13        5,
14        21,
15        19,
16        46,
17        32,
18        898241000
19      ],
20      "nameCreate": "joao@email.com"
21    }
22  ]
23 }
```

- **Testes de Caixa Branca**

Objetivo: Examinar a estrutura interna do código e garantir que todos os caminhos possíveis sejam testados.

Técnica: Analisar o código-fonte e criar casos de teste que cobrem os diferentes caminhos de execução.

Crítérios de Completude: Cobertura de caminho e cobertura de condições e decisões.

Assegurar que todas as instruções e ramos do código são executados pelo menos uma vez.

Requisitos Específicos

Interfaces Externas

➤ Visão Geral

O sistema foi desenvolvido para facilitar a gestão de restaurantes, oferecendo várias funcionalidades importantes. Os usuários podem adicionar, buscar, atualizar e excluir restaurantes, com a exigência de autenticação para garantir a segurança dos dados. Avaliações dos restaurantes são permitidas em uma escala de 1 a 5 estrelas, considerando aspectos como comida, ambiente e funcionários, e os usuários podem deixar comentários opcionalmente. Além disso, é possível visualizar as avaliações e comentários deixados por outros usuários para restaurantes específicos. Os administradores têm permissões amplas, incluindo o gerenciamento completo dos restaurantes e usuários,

enquanto os usuários padrão têm acesso restrito, podendo apenas adicionar avaliações e consultar informações dos restaurantes.

Requisitos Funcionais

➤ CRUD de Restaurantes:

- Adicionar restaurantes;
- Buscar um restaurante específico;
- Atualizar detalhes de um restaurante;
- Deletar um restaurante (com autenticação).

○ Avaliação de Restaurantes:

- Avaliar restaurantes com estrelas (1 a 5);
- Avaliar comida, ambiente e funcionários (individualmente);
- Deixar comentários opcionais ao realizar uma avaliação.

➤ Consulta de Avaliações e Comentários:

- Visualizar avaliações e comentários de outros usuários para um restaurante específico.

➤ Autenticação e Autorização:

- Exigir autenticação para todas as ações relacionadas a adicionar, atualizar ou excluir restaurantes;
- Administradores têm acesso para gerenciar todos os restaurantes e usuários da plataforma.

Requisitos não-funcionais

➤ Tecnologias utilizadas:

- Java 17;
- Spring Boot;
- IntelliJ IDEA;
- PostgreSQL;

- Git;
- Postman;
- HTML5;
- CSS3;
- Visual Studio Code;
- GitActions;
- Jira Software;
- Lombok;
- Mockito;
- Hibernate;
- Dev Tools;

Estudo de Viabilidade

○ O que acontece se o sistema não for implementado?

- Quando os clientes não avaliam os restaurantes, a comunicação falha, prejudicando a confiança dos clientes e a capacidade do restaurante de melhorar. Isso desencoraja novos clientes e impede que o restaurante adapte os seus serviços para agradar mais pessoas.

○ Quais são os problemas com os processos atuais?

- As pessoas necessitam ir até um restaurante e consumir ao menos uma vez para saber a qualidade do restaurante.

○ Como o sistema proposto pode ajudar?

- Tomada de decisão dos consumidores: Usuários podem consultar avaliações e dar opiniões sobre um restaurante.
- Feedback para os restaurantes: Restaurantes recebem avaliações

de usuários e a partir disso podem adaptar seus serviços e construir um boa reputação.

○ Quais serão os possíveis problemas de integração?

- Usuários podem fazer falsas avaliações

○ É necessário a adoção de nova tecnologia ou o desenvolvimento de novas habilidades?

- Não é necessário a adoção de novas tecnologias ou habilidades, uma vez que a utilização de sites é muito conhecida pelo público geral.

○ Quais facilidades devem ser fornecidas pelo sistema?

- Uma seção para “Perguntas frequentes”;
- Alternar Idiomas (PT-BR / EN);
- Alternar tema (Dark / Light);
- Portabilidade, adaptabilidade a diferentes tamanhos de tela.

Informações de Suporte

Documentação Spring Boot: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

Repositório do Projeto: <https://github.com/hanspeterdietiker/A3-Project-Unifacs>

KRU41: The “4+1” view model of software architecture, Philippe Kruchten, November 1995,

<http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf>