INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
GOIANO - CAMPUS CERES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MARCOS ANTONIO DOURADO FILHO

BUSINESS EVAL: BACK-END DE SISTEMA DE AVALIAÇÃO DE DESEMPENHO EMPRESARIAL

CERES - GO
2023

MARCOS ANTONIO DOURADO FILHO

BUSINESS EVAL: BACK-END DE SISTEMA DE AVALIAÇÃO DE DESEMPENHO EMPRESARIAL

Trabalho de conclusão de curso, vinculado à disciplina de Trabalho de Curso II, do curso de Bacharelado em Sistemas de Informação no Instituto Federal Goiano – Campus Ceres.
Orientador(a): Prof. Dr. Rafael Divino Ferreira Feitosa
Coorientador(a): Prof. Dra. Jaqueline Alves Ribeiro

CERES - GO
2023

Sistema desenvolvido pelo ICMC/USP
Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas - Instituto Federal Goiano

DD739b

Dourado Filho, Marcos Antonio
BUSINESS EVAL: BACK-END DE SISTEMA DE AVALIAÇÃO DE DESEMPENHO EMPRESARIAL / Marcos Antonio Dourado Filho; orientador Rafael Divino Ferreira Feitosa; co-orientadora Jaqueline Alves Ribeiro. -- Ceres, 2023.
77 p.

TCC (Graduação em Bacharel em Sistemas de Informação) -- Instituto Federal Goiano, Campus Ceres, 2023.

1. API Rest. 2. Java. 3. Diagnosticar. 4. Desempenho. I. Feitosa, Rafael Divino Ferreira, orient. II. Ribeiro, Jaqueline Alves, co-orient. III. Título.

Responsável: Johnathan Pereira Alves Diniz - Bibliotecário-Documentalista CRB-1 n°2376

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor: Marcos Antonio Dourado Filho

Matrícula: 2018103202030185

Título do trabalho: BUSINESS EVAL: BACK-END DE SISTEMA DE AVALIAÇÃO DE DESEMPENHO EMPRESARIAL

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 07 / 12 / 2023

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Ceres (Local) — 07 / 12 / 2023 (Data)

Marcos Antonio Dourado Filho
Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

Rafael D. F. Feitosa
Assinatura do(a) orientador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

Ao(s) 23 dia(s) do mês de novembro do ano de dois mil e vinte e três, realizou-se a defesa de Trabalho de Curso do(a) acadêmico(a) Marcos Antonio Dourado Filho, do Curso de Bacharelado em Sistemas de Informação, matrícula 2018103202030185, cujo título é "BUSINESS EVAL: BACK-END de sistema de avaliação de desempenho empresarial". A defesa iniciou-se às 21 horas e 15 minutos, finalizando-se às 22 horas e 15 minutos. A banca examinadora considerou o trabalho **APROVADO** com média 8,9 no trabalho escrito, média 9,7 no trabalho oral, apresentando assim média aritmética final de **9,3** pontos, estando o(a) estudante **APTO** para fins de conclusão do Trabalho de Curso.

Após atender às considerações da banca e respeitando o prazo disposto em calendário acadêmico, o(a) estudante deverá fazer a submissão da versão corrigida em formato digital (.pdf) no Repositório Institucional do IF Goiano – RIIF, acompanhado do Termo Ciência e Autorização Eletrônico (TCAE), devidamente assinado pelo autor e orientador.

Os integrantes da banca examinadora assinam a presente.

(Assinado Eletronicamente)
Prof. Dr. Rafael Divino Ferreira Feitosa

Orientador

(Assinado Eletronicamente)
Profª. Dra. Jaqueline Alves Ribeiro

Coorientadora

(Assinado Eletronicamente)
Prof. Me. Roitier Campos Gonçalves

Membro interno

(Assinado Eletronicamente)
Prof. Esp. Paulo Henrique Rodrigues Araujo

Membro externo

Documento assinado eletronicamente por:

- Rafael Divino Ferreira Feitosa, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 23/11/2023 22:17:25.
- Jaqueline Alves Ribeiro, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 23/11/2023 22:19:22.
- Roitier Campos Goncalves, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 24/11/2023 13:41:39.
- Paulo Henrique Rodrigues Araújo, Paulo Henrique Rodrigues Araújo - Outros - Instituto Federal Goiano - Campus Ceres (10651417000410), em 27/11/2023 15:17:50.

Este documento foi emitido pelo SUAP em 23/11/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 550032
Código de Autenticação: 2dcdad9279

INSTITUTO FEDERAL GOIANO
Campus Ceres
Rodovia GO-154, Km.03, Zona Rural, 03, Zona Rural, CERES / GO, CEP 76300-000
(62) 3307-7100

AGRADECIMENTOS

Agradeço a todos os colegas e professores que contribuíram positivamente para a minha trajetória acadêmica, em especial ao meu colega, Thiago Nascimento, que trabalhou junto a mim esta ideia, ao meu orientador, Rafael Feitosa, por todo o empenho, apoio e colaboração na elaboração deste trabalho, assim como a minha coorientadora Jaqueline Ribeiro, que de pronto abraçou este trabalho, dando suporte, direcionamento e incentivo. Por demais, agradeço ao Instituto Federal Goiano - Campus Ceres, como instituição que sempre incentivou o crescimento e desenvolvimento acadêmico.

“O sucesso consiste em ir de fracasso em fracasso sem perder o entusiasmo."

Winston Churchill

RESUMO

Este trabalho aborda o desenvolvimento de um sistema que possibilite a avaliação prática e intuitiva do desempenho empresarial. O projeto, denominado Business Eval, propõe a criação de um sistema que utiliza questionários virtuais para avaliar o desempenho de diferentes setores dentro de uma empresa. O objetivo é simplificar os processos de avaliação e aumentar a eficiência, permitindo a identificação de áreas que requerem atenção. O sistema é composto por uma API Rest, responsável pelo processamento dos dados, e um sistema web, que serve como interface de interação do usuário. O sistema discutido aqui é a API Rest, empregando tecnologias como Java, Spring e PostgreSQL. Os principais públicos de interesse incluem os colaboradores, que respondem ao questionário de diagnóstico, os gestores das empresas, responsáveis por cadastrar as empresas e enviar convites para que os colaboradores respondam aos questionários, e os gestores do sistema, que criam e mantêm as perguntas a serem respondidas e têm acesso a todos os dados. O projeto visa fornecer uma ferramenta eficaz para melhorar o desempenho da empresa, especialmente na produção e no ambiente de trabalho.

Palavras-chave: API Rest; Java; Diagnosticar; Desempenho;

ABSTRACT

This work addresses the development of a system that enables the practical and intuitive evaluation of business performance. The project, called Business Eval, proposes the creation of a system that utilizes virtual questionnaires to assess the performance of different sectors within a company. The objective is to simplify evaluation processes and enhance efficiency, allowing for the identification of areas that require attention. The system consists of a Rest API, responsible for data processing, and a web system, which serves as the user interaction interface. The system discussed here is the Rest API, employing technologies such as Java, Spring, and PostgreSQL. The main stakeholders include employees, who respond to the diagnostic questionnaire, company managers, responsible for registering companies and sending invitations for employees to answer questionnaires, and system managers, who create and maintain the questions to be answered and have access to all data. The project aims to provide an effective tool for improving company performance, especially in production and the work environment.

Keywords: Rest API; Java; Diagnose; Performance;

LISTA DE ABREVIATURAS E SIGLAS

WEB: World Wide Web.
API: Application Programming Interface
IDE: Integrated Development Environment
MVC: Model View Controller
JSON: JavaScript Object Notation
JWT: JSON Web Token
JPA: Java Persistence API
STS: Spring Tools Suite
JRE: Java Runtime Environment
SQL: Structured Query Language
ACID: Atomicity, Consistency, Isolation and Durability
SGBD: Sistema de Gerenciamento de Bancos de Dados
REST: Representational State Transfer RESTful - RESTful Web Services
URIs: Uniform Resource Locators
HTTP: Hypertext Transfer Protocol
POO: Programação Orientada a Objetos

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

O mundo está passando por mudanças sociais, tecnológicas que ameaçam a sobrevivência das empresas em todo o globo. Mudanças no mercado, novos concorrentes com tecnologia mais avançada e exigências normativas são alguns dos motivos que podem afetar a competitividade das empresas, menciona Mascarenhas (2008). Pela necessidade de aprimoramento contínuo do ambiente empresarial, faz-se necessária a utilização de ferramentas que permitam diagnosticar, de maneira eficaz, o desempenho que a empresa possui dentro de seus diversos setores.

O diagnóstico organizacional, de acordo com Cantos (2017), é não apenas concebido como uma ferramenta de análise situacional para embasar decisões, mas também encarado como um processo abrangente, estruturado em fases. Inicialmente, busca-se identificar deficiências, levantar oportunidades de aprimoramento e descrever os problemas existentes; posteriormente, são propostas ações de melhoria, decorre Asencio (2017).

Segundo Porto (2006), a utilização de sistemas de informações gerenciais por meio da Internet permite o acompanhamento das ações organizacionais, a integração entre os vários níveis de tomadas de decisões e o compartilhamento de informações, o que pode otimizar o desempenho operacional das empresas.

O intuito deste trabalho é o desenvolvimento de uma sistema, que tenha como finalidade facilitar e agilizar o processo de avaliação de uma empresa, permitindo observar os setores que possuem uma melhor performance dentro de uma organização, assim como os que necessitam de melhorias, auxiliando na tomada de estratégias, para assim obter melhores resultados no ambiente empresarial.

A ideia da elaboração do sistema Business Eval, foi gerada com o propósito de auxiliar a avaliação de desempenho que as empresas possuem em seus diversos setores através de um questionário dinâmico e prático, que poderá ser respondido por qualquer colaborador daquela organização, e que possui alta disponibilidade de acesso por meio da Internet, tratando-se da plataforma web.

O sistema Business Eval se trata de um programa que permite diagnosticar o desempenho que os setores diversos de uma organização possuem com relação aos métodos de atuação e ambiente de trabalho dos colaboradores, possibilitando uma visão ampla do estado individual de cada setor em contraste com os demais setores. Possui, como objetivo, possibilitar ao usuário basear-se nos dados obtidos através de questionário que é apresentado

dinamicamente, onde cada questão deve ser respondida por meio de uma escala de 0 a 10, onde 0 representa “discordo totalmente” e 10 representa “concordo totalmente”.

Para a elaboração deste projeto, foi idealizado desenvolvimento de dois sistemas, sendo o primeiro uma **API Rest**, que é responsável por todo tratamento dos dados, desde a persistência dos mesmos em uma base de dados remota até todo o processamento das informações, regras de negócio, validação e entrega dos resultados, por meio de requisições utilizando o protocolo **HTTP.** O segundo, uma Aplicação Web, que tem como função introduzir o funcionamento do sistema ao usuário, mostrando de forma gráfica e dinâmica os questionários e apresentar os resultados de modo esquematizado (gráfico de radar). Toda interação intersistêmica acontece por meio das requisições, delimitando as funções de cada subsistema. No presente documento, será abordado o desenvolvimento apenas da **API Rest**, que pode ser nominalmente identificada com **back-end.**

Conforme ressaltado por Araújo (2005), aprimorar a eficiência de uma organização está intrinsecamente ligado à habilidade de realizar um diagnóstico preciso de suas próprias forças e fraquezas. Essa perspectiva destaca a aplicação da ferramenta de diagnóstico como uma etapa inicial essencial no processo de intervenção organizacional.

Devido a importância que uma ferramenta específica, como aqui apresentada, surgiu o projeto, Business Eval. Este, tendo como tema central o desenvolvimento de um back-end de sistema de avaliação de desempenho empresarial, que utiliza de um questionário virtual como ferramenta para diagnóstico de performance dos setores de uma empresa.

2. METODOLOGIA

Na elaboração deste projeto, foi optado por utilizar a arquitetura de software em camadas, sendo estas divididas em **back-end** e **front-end**. Essa separação se dá para que haja mais clareza na elaboração das funcionalidades diversas do projeto, onde cada camada pode ser desenvolvida separadamente. Desta forma elas podem interagir entre si, mas sem que uma interfira diretamente no funcionamento da outra. Este modelo é muito utilizado para lidar com o desenvolvimento de aplicações que possuem interações com múltiplas plataformas, e também para permitir que seja desenvolvido, de forma independente, por vários desenvolvedores, o mesmo projeto.

O desenvolvimento do Business-Eval se dividiu em duas etapas, sendo a primeira responsável pelo processo de modelagem do sistema, onde foi realizado levantamento de requisitos, definição de regras de negócio e levantamento de ferramentas mais apropriadas para o projeto.

Para o desenvolvimento da **API**, foram definidas as seguintes ferramentas, que foram agrupadas segundo suas características:

Linguagem de Programação: **Java;**

Ecossistema: **Spring;**

Banco de dados: **PostgreSQL;**

Posteriormente foi iniciada a etapa de implementação, onde ocorreu a codificação da **API**, utilizando as ferramentas anteriormente citadas. Na primeira fase da implementação, foi realizada a codificação do banco de dados PostgreSQL, e criação do servidor. A seguir, é definido o padrão **MVC**, sendo usual na utilização do ecossistema **Spring,** para codificação do **Java**. Após realizada a implementação, foram realizados testes para validação do funcionamento, assim como correções e aprimoramentos nas funcionalidades.

Importante ressaltar que concomitantemente está sendo elaborado o sistema que irá possuir a interface, que será utilizada para que o usuário possa realizar suas interações, assim como a visualização dos dados, se conectando com o sistema, aqui apresentado. O mesmo está sendo desenvolvido para a plataforma **WEB**, sendo feito uso do ecossistema **Node.JS**, onde a aplicação é elaborada utilizando o framework **Angular JS.**

3. DESCRIÇÃO DO SISTEMA

O sistema é uma **API Rest** desenvolvida para atender a necessidade de diagnósticos rápidos e fáceis, expostos por meio de gráficos de radar em uma interface terceira, em que se pode avaliar, de forma clara, os setores de uma empresa onde exista maior necessidade de intervenção e melhorias.

Para o desenvolvimento da **API** foi definida a utilização da linguagem de programação **JAVA**, utilizando os frameworks oferecidos pelo ecossistema **Spring**, que permitem o aproveitamento de códigos e a utilização de diversos recursos já existentes. Usando da integração do modelo **POO** (Paradigma de orientação a objetos), paradigma que melhor se adequa ao projeto, juntamente à base de dados, por meio **Spring Data JPA**, que comunica os serviços do **SGBD (**sistema de gerenciamento de banco de dados), **PostgreSQL**, que foi escolhido pelas vantagens de ser de fácil manuseio e não necessitar da compra de licença para uso, seja qual for sua finalidade.

3.1 Descrição do Problema

Segundo Chiusoli (2017), muitas empresas e organizações sofrem com problemas de gestão de desempenho, onde o ambiente de trabalho prejudica a produção devido a falta de organização estratégica. É evidente que, quanto maior a empresa, mais complexa se torna a identificação dos setores que estão desalinhados com o papel considerado ideal.

Considerando esses fatores, foi compreendido como necessária a criação de uma ferramenta que permite avaliar de forma rápida e dinâmica, não só do ponto de vista do gerente, mas também dos colaboradores. Esse sistema deverá permitir alcançar grandes resultados em diversos aspectos na empresa, principalmente na produção e na qualidade do ambiente de trabalho.

3.2 Principais Envolvidos e suas Características

O sistema BUSINESS-EVAL é uma ferramenta que possui diversos indivíduos que participam do seu funcionamento, possuindo, cada um, uma atribuição específica.

3.2.1 Usuários do Sistema

1. **Gestor da empresa:** É o usuário que realiza o cadastro da empresa. Ele pode preencher o formulário e visualizar os resultados obtidos com todas as respostas.
2. **Colaboradores:** O colaborador pode preencher o formulário pré-cadastrado da sua empresa. Ele não tem acesso aos resultados dessa empresa da qual é colaborador, mas pode fazer uso do mesmo usuário para assumir o papel de gestor de empresa, caso opte por cadastrar uma empresa.
3. **Gerente do Sistema:** Ele tem acesso aos resultados de todas as empresas e pode realizar filtragem personalizada de dados, e manter o cadastro de questões e categorias.

3.2.2 Desenvolvedores do Sistema

As atribuições de manutenção e integração do sistema com a interface web é entregue a um *programador*. Ele deve realizar melhorias e correções de bugs no projeto, garantindo a disponibilidade do sistema e sua integridade.

3.3 Regras de Negócio

Os componentes essenciais de um sistema de informação organizacional são as Regras do Negócio, cuja relevância tem sido cada vez mais reconhecida nos últimos anos. Conforme enfatizado por LEITE & LEONARDI (1998), essas regras desempenham um papel crucial no processo de definição de requisitos para sistemas de informação, sendo consideradas como declarações genéricas fundamentais sobre a organização.

A seguir, são listadas as regras de negócios do Business-Eval:

1. **Realizar login**: para realizar login é necessário informar um email válido. Após este ato, será enviado ao email um código de acesso.
2. **Manter usuário**: As informações do usuário só poderão ser alteradas ou excluídas pelo próprio usuário ou pelo *gerente do sistema*.
3. **Cadastro de empresa**: todo usuário autenticado pode cadastrar uma empresa.
4. **Identificação de empresa**: cada empresa possui o CNPJ como chave única de identificação, não podendo ser cadastrada mais de uma empresa com o mesmo CNPJ.

5. **Manter empresa**: As informações da empresa só poderão ser alteradas ou excluídas pelo usuário que a criou ou pelo *gerente do sistema*.
6. **Responder formulário de desempenho**: o formulário de desempenho só poderá ser respondido pelo criador da empresa e colaboradores autorizados por ele e que façam a autenticação pelo email pré cadastrado.
7. **Visualização dos resultados da empresa**: A visualização dos resultados poderá ser realizada pelo usuário que a criou ou pelo *gerente do sistema.*
8. **Manter formulário**: o formulário é dividido por categorias e cada categoria possui questões cadastradas. Cada questão e cada categoria possui uma ordem de apresentação. Somente o *gerente do sistema* pode cadastrar e excluir as questões, categorias e alterar a ordem de apresentação de ambas.

3.4 Ferramentas Utilizadas

As ferramentas escolhidas para o desenvolvimento do sistema foram caracterizadas pelo conhecimento prévio e suas vantagens tecnológicas para a construção do software.

1. **GitHub**: Plataforma para hospedagem de código-fonte e arquivos com controle de versão usando Git.
2. **Java com Spring**: Java é uma linguagem de programação para construir aplicações em rede e o Spring é um framework open source para Java que usa inversão de controle e injeção de dependência.
3. **PostgreSQL**: Sistema de gerenciamento de banco de dados relacional de código aberto com excelente documentação, baixo consumo de recursos e suporte completo a **SQL** e transações **ACID**.
4. **Docker**: O Docker é uma plataforma de código aberto que automatiza a implantação, o dimensionamento e a gestão de aplicações em contêineres.

3.5 Escopo

Esse sistema tem como objetivo dispor um software que possibilite o diagnóstico rápido e preciso do desempenho de empresa:

- Cadastro de empresa.
- Cadastro de colaboradores.

- Verificação de desempenho.
- Obtenção de dados para futuras pesquisas

Premissas:

- Delegar a responsabilidade de desenvolvimento a um programador capacitado;
- Obtenção dos recursos e ferramentas necessárias.

Restrições:

- Não ultrapassar o tempo pré-estabelecido para a conclusão do sistema.

4. REQUISITOS DO SISTEMA

Este capítulo tem como objetivo descrever os requisitos do sistema. Está subdividido em requisitos funcionais e requisitos não funcionais, para melhor entendimento das funcionalidades do sistema.

4.1 Requisitos Funcionais (casos de uso)

Abaixo estão indicados os requisitos funcionais da aplicação que orienta o desenvolvimento da **API**.

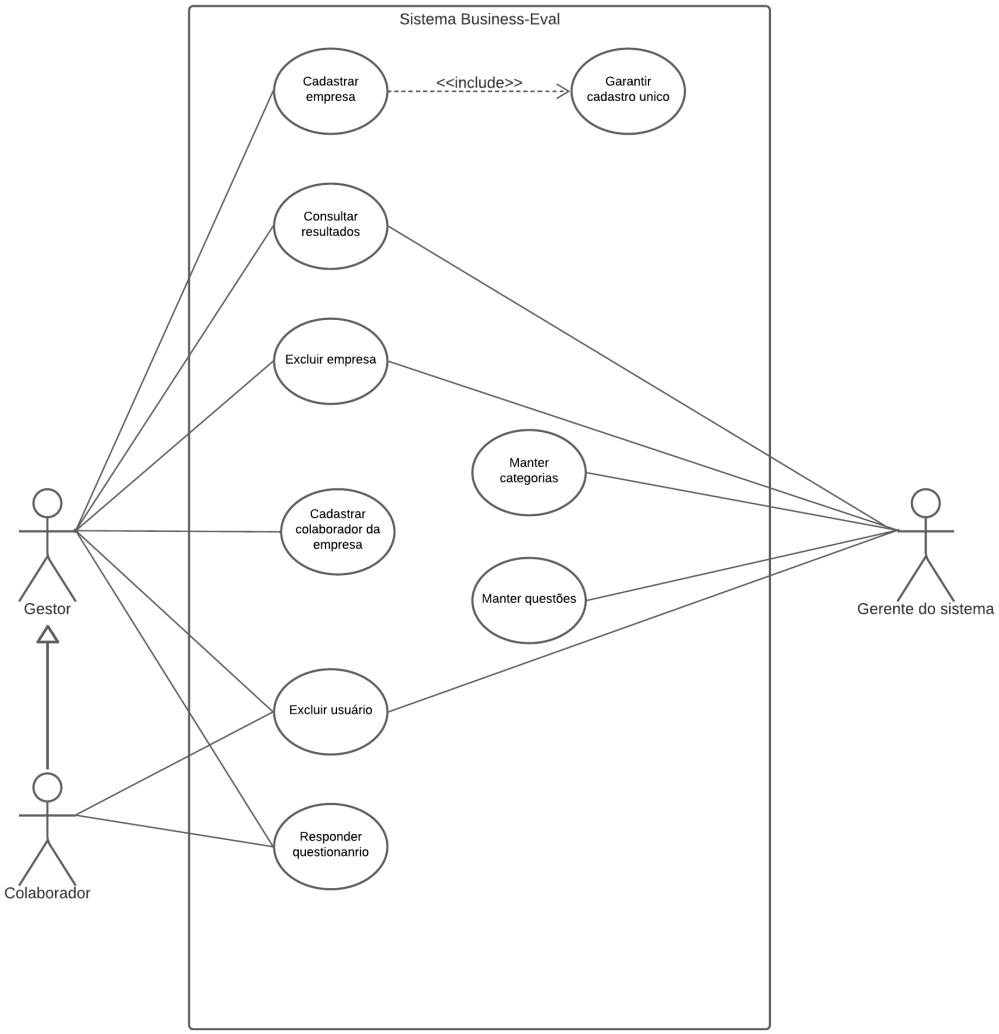

Figura 1. Diagrama: Diagrama de caso de uso.

4.1.1 [RF001] Realizar login

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir que todos os usuários consigam realizar a autenticação pelo código de acesso enviado por email.

4.1.2 [RF002] Realizar cadastro de empresa

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir aos usuários cadastrar empresas para que sejam avaliadas.

4.1.3 [RF003] Cadastrar colaboradores para responder questionário

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir que os gestores consigam cadastrar colaboradores para responderem o questionário de determinada empresa.

4.1.4 [RF004] Responder questionário de diagnóstico

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir a resposta do usuário que cadastrou a empresa e os colaboradores cadastrados, por meio de uma escala de 0 a 10.

4.1.5 [RF005] Consultar resultados para criação do gráfico

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir a consulta dos resultados da avaliação, para que seja gerado o gráfico de radar no software de interface aos gestores e ao gerente do sistema.

4.1.6 [RF006] Manter categorias

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir ao gerente do sistema cadastrar, excluir e ordenar as categorias de questões.

4.1.7 [RF008] Manter questões

Prioridade: ☒ Essencial ☐ Importante ☐ Desejável

O sistema deve permitir ao gerente do sistema cadastrar, excluir e ordenar as questões por categoria.

4.1.8 [RF007] Excluir usuários

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

O sistema deve permitir a exclusão do usuário, seja o próprio usuário autenticado, seja este colaborador ou gestor de empresa. Além disso, ao gerente do sistema é permitido excluir todos os usuários, inclusive o próprio.

4.1.9 [RF009] Garantir cadastro único de empresa

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

O sistema deve garantir que não seja cadastrada mais de uma empresa com o mesmo CNPJ.

4.1.10 [RF010] Excluir empresas

Prioridade: ☐ Essencial ☒ Importante ☐ Desejável

O sistema deve permitir excluir empresa ao usuário gestor e ao gerente do sistema.

4.2 Requisitos não funcionais

Abaixo estão indicados os requisitos não funcionais da aplicação que definem os padrões da **API**.

4.2.1 [RNF001] Arquitetura do sistema

O sistema deve ser desenvolvido em linguagem Java, utilizando o ecossistema Spring e o paradigma de orientação a objetos.

4.2.2 [RNF002] Integração com banco de dados

O sistema deve se comunicar com um banco de dados **PostgreSQL** para persistência dos dados.

4.2.3 [RNF003] Disponibilidade

O sistema deve ter alta disponibilidade de acesso por meio da internet, tratando-se da plataforma web.

4.2.4 [RNF004] Segurança

O sistema deve ser seguro e garantir a privacidade dos dados das empresas e colaboradores, usando mecanismo de autenticação.

4.3 Métricas e Cronograma

Cronograma é uma ferramenta de gestão de atividades, normalmente em forma de tabela, que também contempla o tempo em que as atividades vão se realizar.

Cronograma de execução do Projeto - 2023										
Atividade do Projeto	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov
Análise de viabilidade										
Definição do Objetivo										
Planejamento										
Elicitação dos Requisitos										
Especificação dos recursos										
Gerência dos Requisitos										
Configuração de plataformas										
Codificação										
Testes e Validação										
Implantação										

Figura 2. Tabela: Cronograma de execução do projeto

5. ANÁLISE E DESIGN

5.1 Arquitetura do Sistema

O sistema Business-Eval ficou dividido em 3 camadas. A primeira delas se caracteriza como a interface web, que deverá ser acessada pelo navegador, na qual o usuário terá acesso dinamicamente aos dados fornecidos pelo sistema, onde será feita às requisições ao servidor e apresentará as respostas, ressaltando que essa camada não está sendo abordada nesse documento. A segunda e a terceira camadas, servidor Tomcat e servidor **PostgreSQL**, respectivamente, são as camadas apresentadas por meio desta documentação, sendo a primeira responsável por manter os serviços da **API** em funcionamento, onde deve ocorrer todo o processamento dos dados. Na terceira camada, encontra-se o servidor **PostgreSQL**, responsável pelo armazenamento dos dados coletados pela aplicação, tais como informações de empresas, questões e respostas. (Exemplo)

Abaixo está representada a arquitetura do sistema por meio de diagrama:

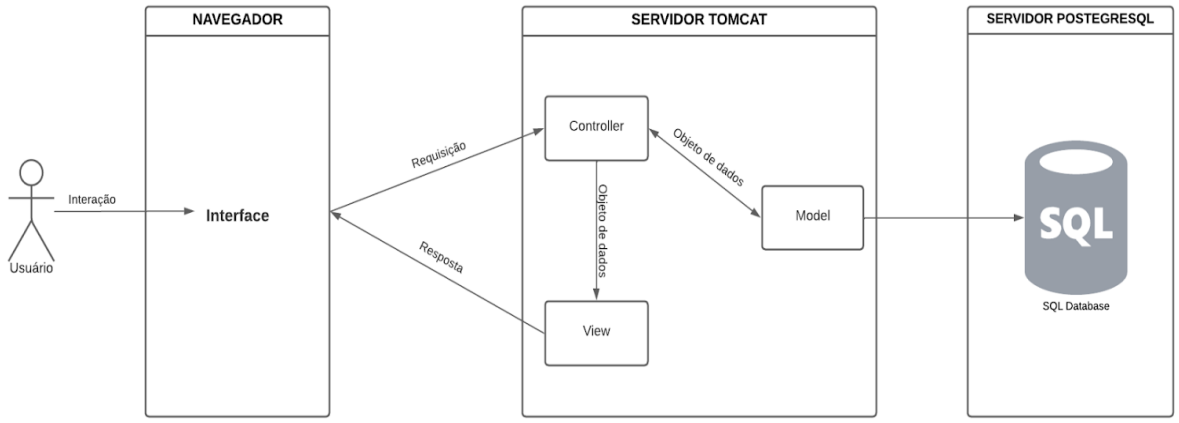

Figura 3. Diagrama: Arquitetura do sistema

5.2 Modelo do Domínio

Abaixo é apresentado o diagrama de classes, representando o formato que os dados serão trabalhados no sistemas através do paradigma de orientação a objetos, ficando subdividido em 6 classes, sendo estas: **User**, **Business**, **BusinessUser**, **Answer**, **Question** e **Category**. Também possui um enumerador chamado Authority, demonstrando seus atributos e tipo, além de também demonstrar os métodos definidos.

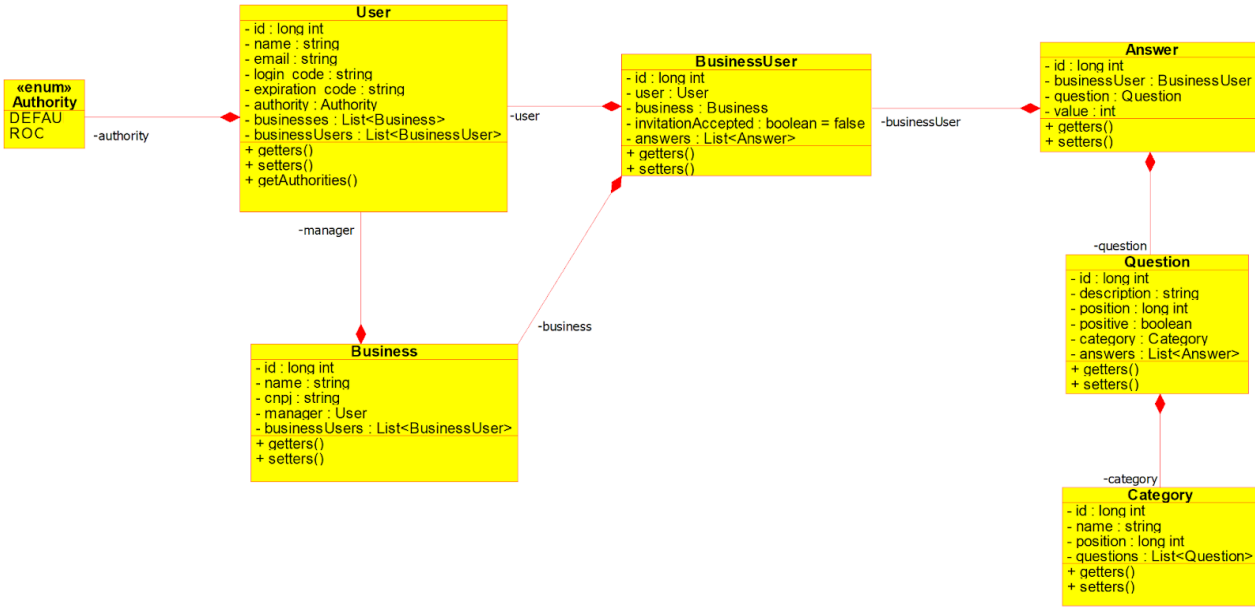

Figura 4. Diagrama: Diagrama de classes

5.3 Sequência de Eventos

A seguir, são apresentados alguns diagramas de sequência que representam o comportamento do sistema com relação aos casos de uso **Cadastrar Empresa**, **Cadastrar Colaborador**, **Excluir Empresa**, **Excluir Usuários**, **Responder Questionário** e **Consultar Resultados**, respectivamente. A estrutura adotada para o sistema foi um modelo dividido em **Controller** responsável por receber as requisições, **Service** responsável pela validação e tratamento dos dados e **Repository** que no diagrama é identificado com **Banco de Dados**, mas se trata de objeto java que realiza as funções de **CRUD** diretamente no banco de dados.

A figura 5 apresenta os eventos que ocorrem no sistema durante o processo de realização login; já a figura 6 apresenta os eventos do processo de cadastro de empresa; seguindo na figura 7 são exibidas as etapas de se cadastrar um colaborador; adiante na figura 8

são expostos os eventos de exclusão de uma empresa; a figura 9 demonstra a sequência dos eventos de se exclusão do próprio usuário no sistema; a figura 10 expõe os gatilhos do sistemas ao se responder uma questão; por fim, a figura 11 retrata os procedimentos do sistema para consulta de resposta do usuário.

5.3.1 Realizar Login

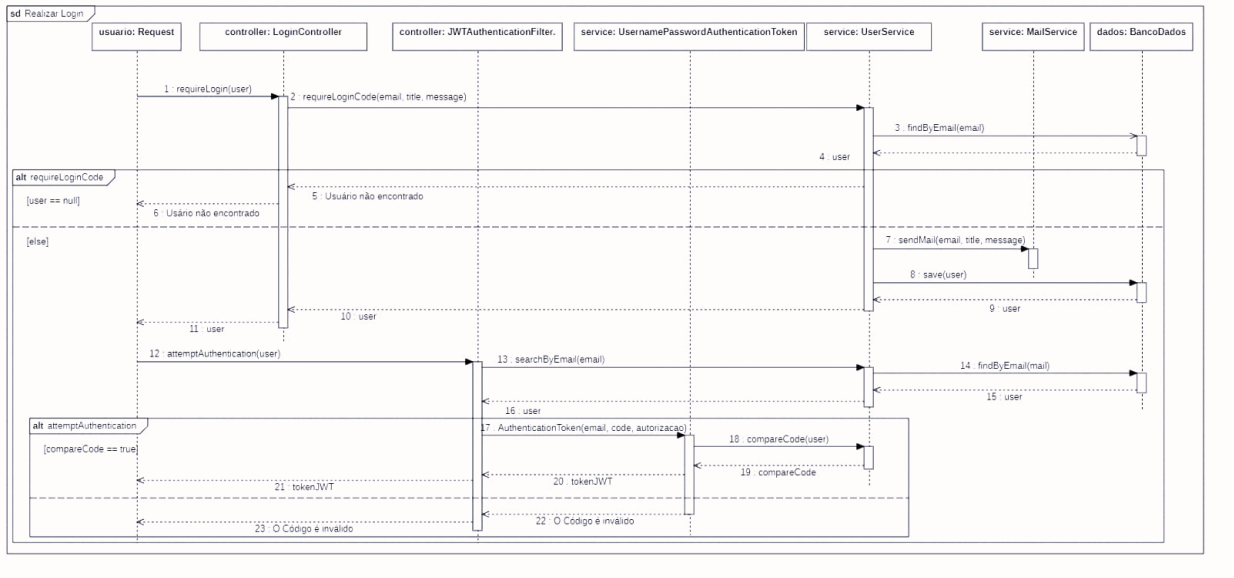

Figura 5. Diagrama: Diagrama de Sequência (Realizar Login)

5.3.2 Cadastrar Empresa

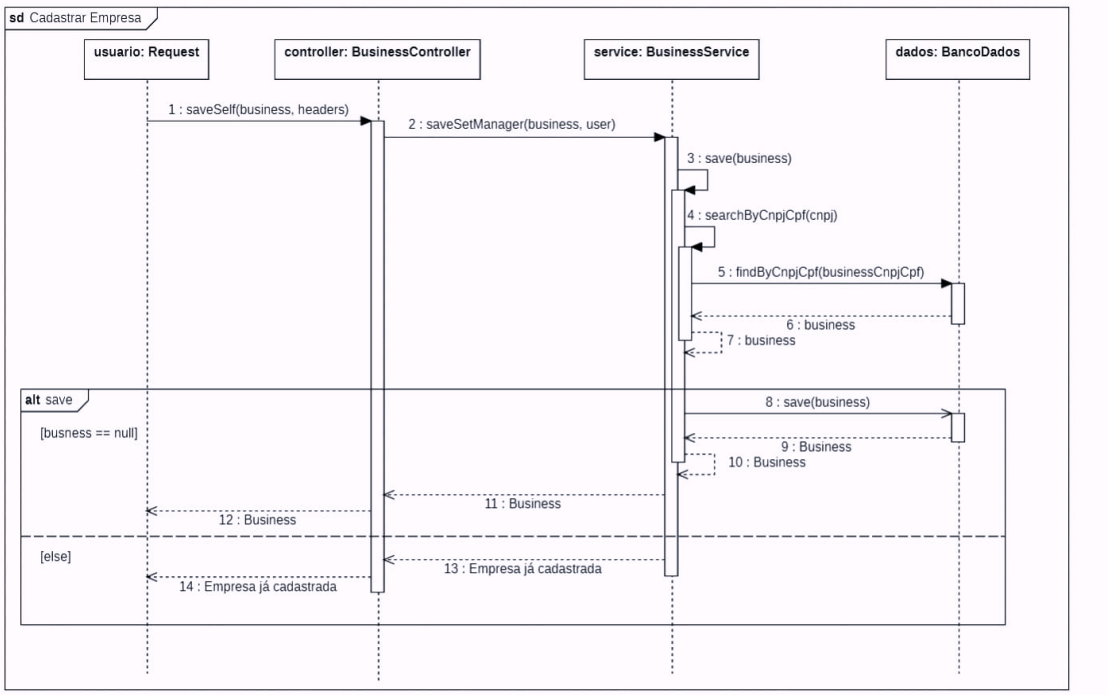

Figura 6. Diagrama: Diagrama de Sequência (Cadastrar Empresa)

5.3.3 Cadastrar Colaborador

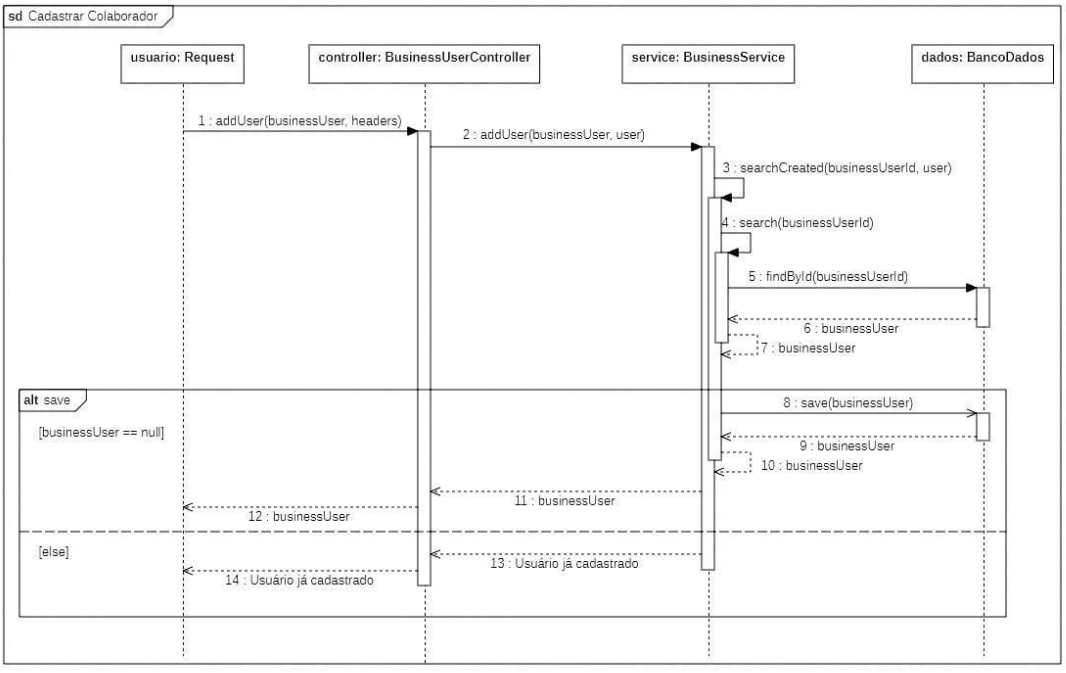

Figura 7. Diagrama: Diagrama de Sequência (Cadastrar Colaborador)

5.3.4 Excluir Empresa

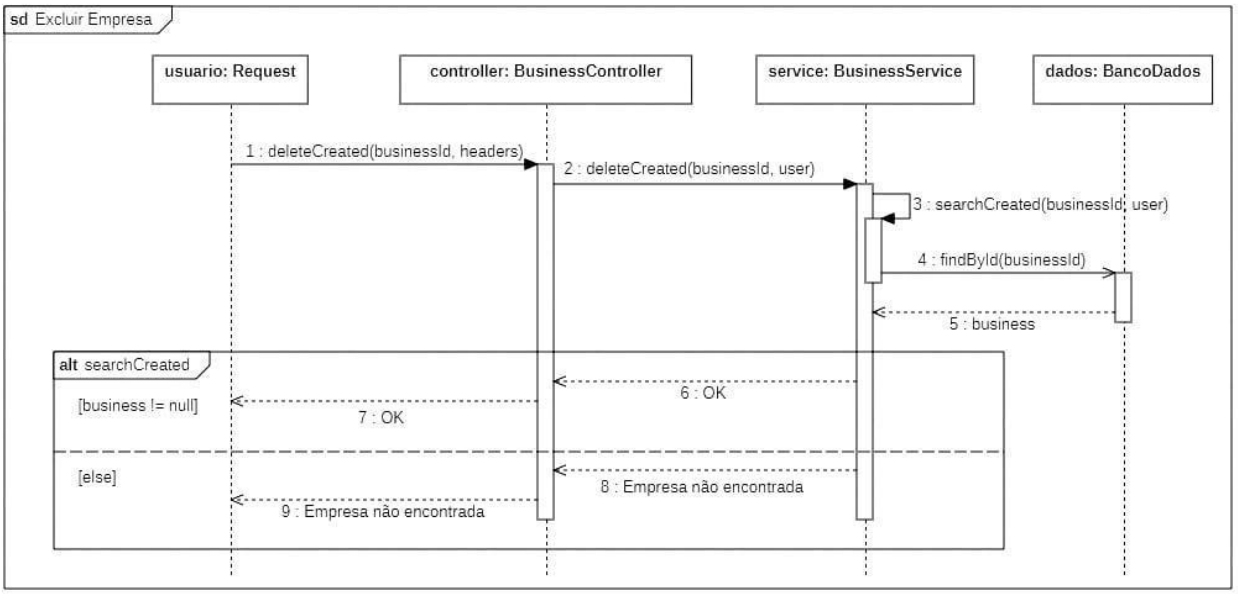

Figura 8. Diagrama: Diagrama de Sequência (Excluir Empresa)

5.3.5 Excluir Usuário

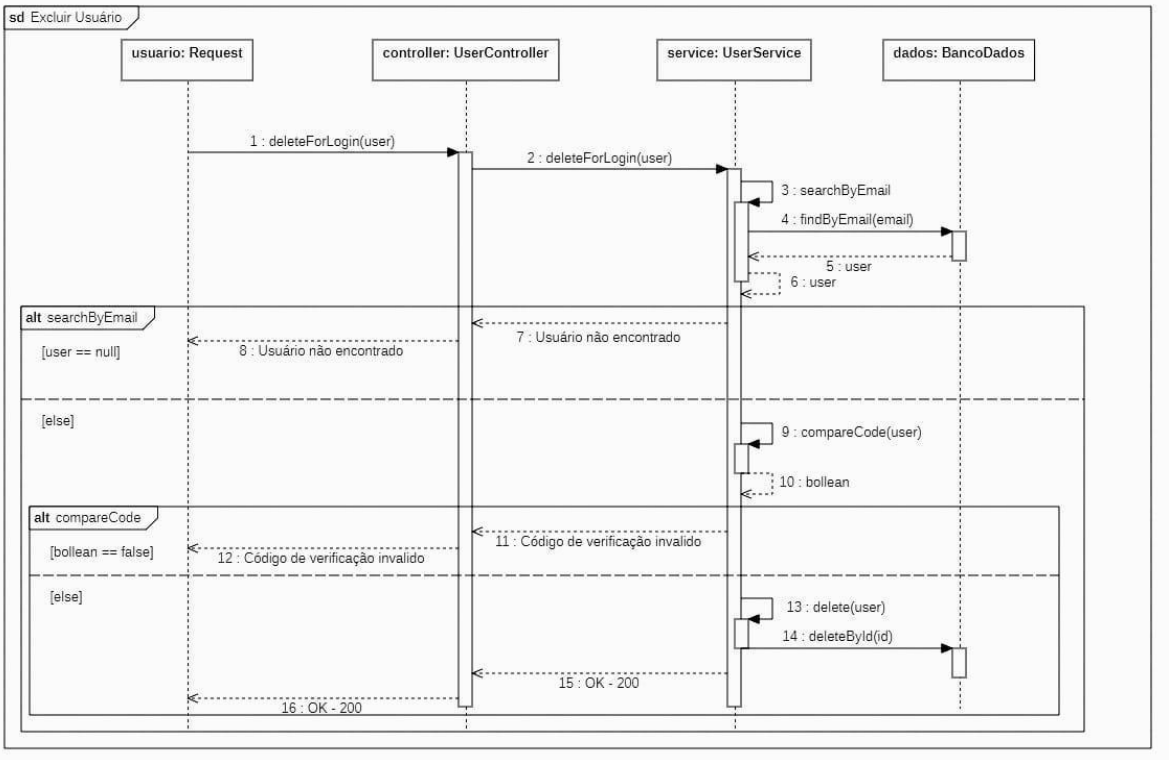

Figura 9. Diagrama: Diagrama de Sequência (Excluir Usuário)

5.3.6 Responder Questionário

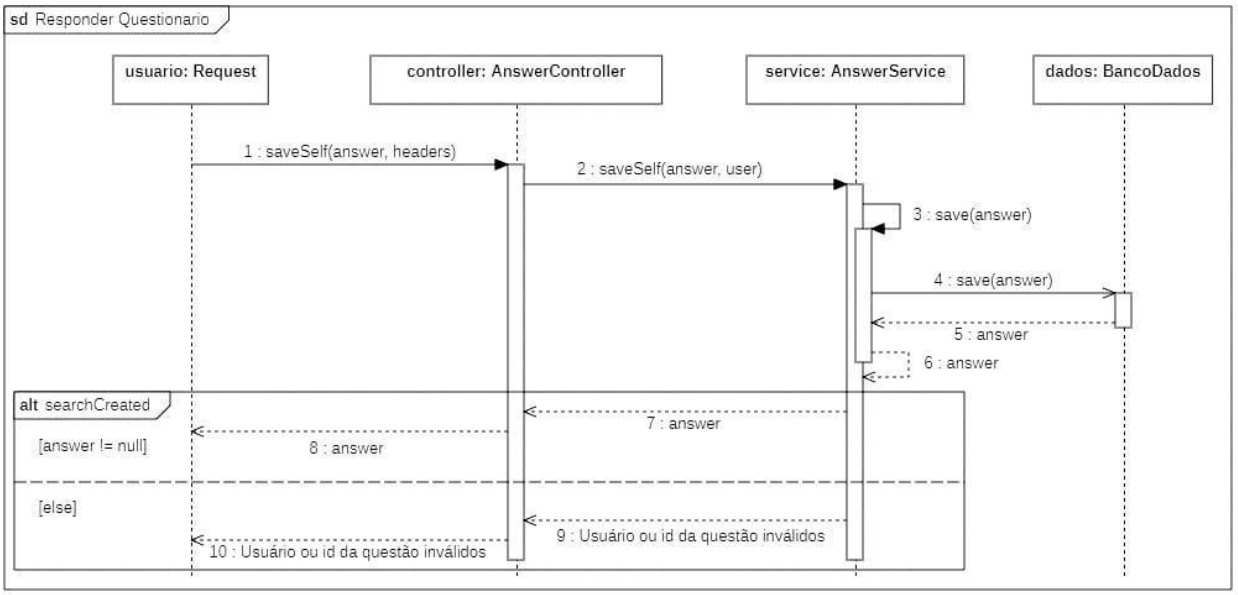

Figura 10. Diagrama: Diagrama de Sequência (Responder Questionário)

5.3.7 Consultar Resultados

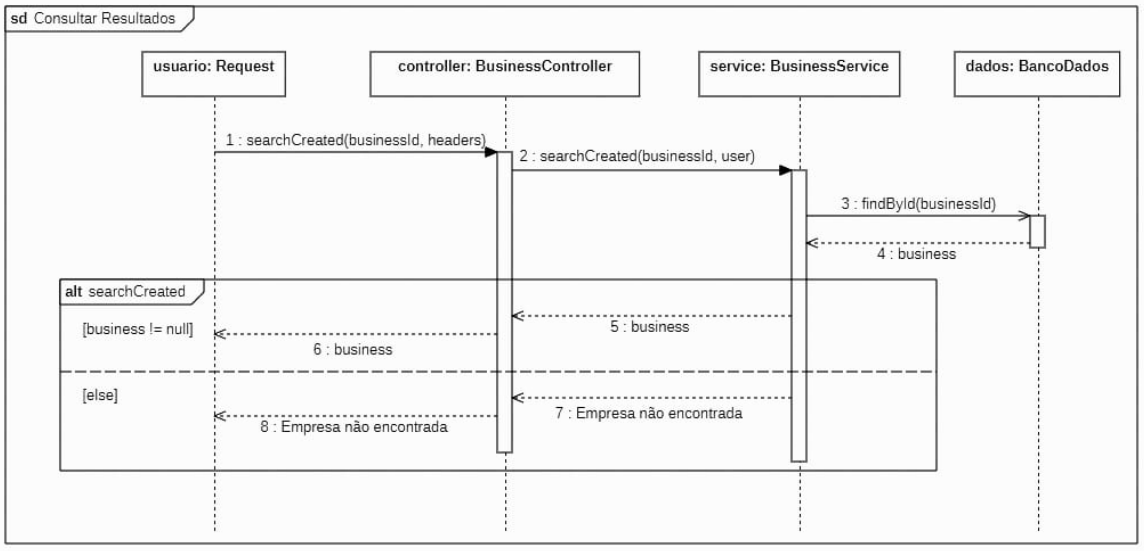

Figura 11. Diagrama: Diagrama de Sequência (Consultar Resultado)

5.4 Modelo de Dados

Este tópico apresenta como foi elaborada a base de dados que pertencente ao sistema, partindo da modelagem até a codificação.

5.4.1 Modelo Lógico da Base de Dados

Neste item é apresentado o modelo lógico da base de dados, o modelo entidade-relacionamento da base de dados, desenvolvido na ferramenta **MySQL Workbench.**

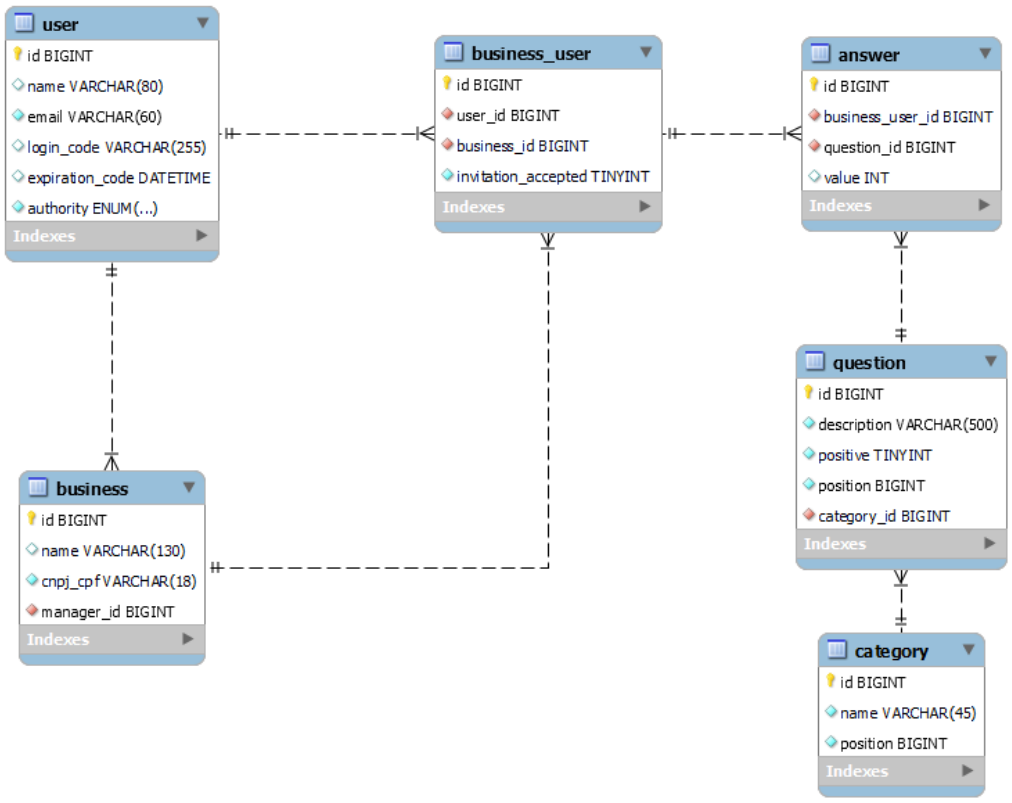

Figura 12. Diagrama: Modelo de Entidade e Relacionamento

5.4.2 Codificação do Modelo Físico

Abaixo é apresentado o código utilizado para a criação das tabelas da base de dados no formato **SQL** compatível com o **SGBD PostgreSQL.**

5.4.2.1 Tabela de Usuários

```
CREATE TYPE authorities AS ENUM('DEFAULT', 'ROOT');
CREATE TABLE sys_user (
   id SERIAL CONSTRAINT pk_id_user PRIMARY KEY,
   name varchar(80),
   email varchar(60) NOT NULL UNIQUE,
   login_code varchar(255),
   expiration_code TIMESTAMP WITH TIME ZONE,
   authority authorities NOT NULL DEFAULT 'DEFAULT'
);
CREATE CAST (CHARACTER VARYING as authorities) WITH INOUT AS IMPLICIT;
```

5.4.2.2 Tabela de Empresas

```
CREATE TABLE business (
   id SERIAL CONSTRAINT pk_id_business PRIMARY KEY,
   name varchar(130),
   cnpj_cpf varchar(18) NOT NULL UNIQUE,
   manager_id bigint NOT NULL,
   FOREIGN KEY (manager_id) REFERENCES sys_user (id)
);
```

5.4.2.3 Tabela de Relacionamento de Usuários e Empresas

```
CREATE TABLE business_user (
   id SERIAL CONSTRAINT pk_id_business_user PRIMARY KEY,
   user_id bigint NOT NULL,
   business_id bigint NOT NULL,
   invitation_accepted BOOLEAN NOT NULL DEFAULT FALSE,
   FOREIGN KEY (user_id) REFERENCES sys_user (id),
   FOREIGN KEY (business_id) REFERENCES business (id)
);
```

5.4.2.4 Tabela de Categorias

```
CREATE TABLE category (
   id SERIAL CONSTRAINT pk_id_category PRIMARY KEY,
   name varchar(45),
   position bigint
);
```

5.4.2.4 Tabela de Questões

```
CREATE TABLE question (
   id SERIAL CONSTRAINT pk_id_question PRIMARY KEY,
   description varchar(500),
   position bigint,
   category_id bigint NOT NULL,
   FOREIGN KEY (category_id) REFERENCES category (id)
);
```

5.4.2.5 Tabela de Respostas

```
CREATE TABLE answer (
   id SERIAL CONSTRAINT pk_id_answer PRIMARY KEY,
   value_answer integer DEFAULT 0,
   question_id bigint NOT NULL,
   business_user_id bigint NOT NULL,
   FOREIGN KEY (question_id) REFERENCES question (id),
   FOREIGN KEY (business_user_id) REFERENCES business_user (id)
);
```

5.5 Ambiente de desenvolvimento

O ambiente de desenvolvimento conta com diversas ferramentas utilizadas para auxiliar no desenvolvimento, depuração, gestão e implantação. O ambiente se compõe das ferramentas aqui listadas:

1. **Java 17**: Linguagem de programação utilizada para desenvolvimento da **API**.
2. **SQL**: Linguagem de programação para armazenar e processar informações na base de dados.
3. **Spring**: Ecossistema de recursos integrados para gerenciar dependências e configurações, contando com bibliotecas de recursos próprios.
4. **STS** (Spring Tools Suite): **IDE** utilizado para facilitar a codificação e refatoração dos códigos, também conta com mecanismo de debug, e atalhos de execução e teste.
5. **Postman**: Uma **API** cliente que realiza requisições **HTTP** e **HTTPs**, usando interface de fácil interação, permitindo realizar testes de requisições manualmente e documentar os casos de teste e exportá-los em formato **JSON**.
6. **MySQL Workbench**: Ferramenta que permite a modelagem do modelo lógico da base de dados.
7. **DIA**: Software de criação de diagramas, utilizado para criação do diagrama de classes, possuindo o recurso de exportação do modelo em código usual.
8. **StarUML**: Sistema de desenvolvimento de diagramas, utilizado para para criação dos diagramas de sequência e caso de uso.
9. **Docker**: Plataforma para automatização e implantação de serviços em containers, usado para gerenciar os serviços do servidor **Tomcat**, carregado pelo interpretador **JRE**, e servidor de dados **PostgreSQL**.

5.6 Sistemas e componentes externos utilizados

Para o desenvolvimento da **API** são utilizados diversos recursos externos, bibliotecas parceiras e recursos do ecossistema Spring. A seguir serão listados os componentes e recursos utilizados:

1. **Maven**: Ferramenta de automação e gerência de projetos Java.
2. **Spring Boot**: Sistema gerenciador de configuração de dependências.
3. **Spring Data JPA**: Biblioteca de recursos de consulta simplificada e integração com a base de dados.

4. **Spring Validation**: Biblioteca de validação de objetos de dados.
5. **Spring Security**: Biblioteca de recursos de segurança e autenticação.
6. **Spring Mail:** Biblioteca de serviço integração com correio eletrônico.
7. **Spring DevTools**: Recurso responsável por reiniciar servidor sempre que é realizada alteração no código.
8. **Lombok**: Biblioteca responsável por automatizar a criação de métodos padrões nas classes.
9. **ModelMapper**: Biblioteca responsável por mapear dados de objetos, transpondo dados para objetos de outras classes.
10. **Fly Migration**: Biblioteca responsável por executar códigos de migração **SQL** e por versionamento das modificações na base de dados.

6. RECURSO E SERVIÇOS

Logo abaixo serão apresentados os **endpoints** implementados para utilização dos serviços do sistema através de requisições **HTTP** e **HTTPs**. É válido destacar que todas as requisições realizadas possuem o parâmetro *Authorization* com o valor referente ao **token JWT** definido em seu cabeçalho, exceto as requisições do subtópico Login.

6.1 Login

Os **endpoints** apresentados dentro deste subtópico estão relacionados ao processo de autenticação da API.

1. Require Login:
    - **Endpoint:** */requirelogin.*
    - **Método**: *POST*.
    - Ele é responsável por solicitar o código de login que é enviado ao *email* passado no corpo da requisição.
2. Login:
    - **Endpoint:** */login.*
    - **Método**: *POST*.
    - Ele é responsável por solicitar a autenticação passando o *email* e o código de acesso (*loginCode*) no corpo da requisição, retornando um **token JWT**.

6.2 Users

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso padrão.

1. Search Self:
    - **Endpoint:** */users.*
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados do usuário autenticado no sistema.
2. Edit Self:

- **Endpoint:** */users/{id}*.
- **Método**: *PUT*.
- Ele é responsável por editar os dados do usuário autenticado no sistema. Necessário passar o identificador do usuário (*id*) como parâmetro na URI.

3. Delete Self:
    - **Endpoint:** */users.*
    - **Método**: *DELETE*.
    - Ele é responsável por excluir os dados do usuário autenticado por meio de autenticação mediante um código enviado por email, assim como no processo de login.

6.3 Businesses

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso padrão.

1. Created Set Manager:
    - **Endpoint:** */businesses.*
    - **Método**: *POST*.
    - Ele é responsável por salvar "empresa" e definir "gerente" pelo usuário autenticado, passando o nome da empresa (*name*) e o CNPJ ou CPF da empresa (*cnpjCpf*) no corpo da requisição.
2. Search Created:
    - **Endpoint:** */businesses/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de empresa cadastrada pelo usuário autenticado no sistema. Necessário passar o identificador da empresa (*id*) como parâmetro na URI.
3. Search Created CNPJ_CPF:
    - **Endpoint:** */businesses/cnpjcpf.*
    - **Método**: *GET*.

- Ele é responsável por consultar os dados de empresa cadastrada pelo usuário autenticado no sistema através do CNPJ ou CPF (*text*) informado no corpo da requisição.

4. List Created:
    - **Endpoint:** */businesses*.
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todas as empresas cadastradas pelo usuário autenticado no sistema.
5. Edit Created:
    - **Endpoint:** */businesses/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por editar os dados de empresa cadastrada pelo usuário autenticado no sistema. Necessário passar o identificador da empresa (*id*) como parâmetro na URI.
6. Delete Self:
    - **Endpoint:** */businesses/{id}*.
    - **Método**: *DELETE*.
    - Ele é responsável por excluir os dados de empresa cadastrada pelo usuário autenticado no sistema. Necessário passar o identificador da empresa (*id*) como parâmetro na URI.

6.4 BusinessesUsers

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso padrão.

1. Search Created:
    - **Endpoint:** */businessesuser/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados do colaborador por empresa cadastrada pelo usuário autenticado no sistema. Necessário passar o identificador do relacionamento colaborador empresa (*id*) como parâmetro na URI.

2. List Self All:
    - **Endpoint:** */businessesuser*.
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todos os relacionamentos do usuário autenticado como colaborador de empresas cadastradas no sistema.
3. Add User:
    - **Endpoint:** */businessesuser*.
    - **Método**: *POST*.
    - Ele é responsável por adicionar um usuário como colaborador de uma empresa cadastrada pelo usuário autenticado, passando o email (*userEmail*) do colaborador e o identificador da empresa (*businessId*), o título para o convite a ser enviado no email do colaborador (*customTitle*) e a mensagem (*customMessage*) no corpo da requisição.
4. Invitation Accepted:
    - **Endpoint:** */businessesuser/accepted/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por definir como aceito o atributo (invitationAccepted), que é usado para identificar se o usuário aceitou responder questionário da empresa que foi registrado como colaborador.
5. Delete Self:
    - **Endpoint:** */businessesuser/{id}*.
    - **Método**: *DELETE*.
    - Ele é responsável por excluir o relacionamento de colaborador com a empresa cadastrada pelo usuário autenticado no sistema. Necessário passar o identificador do relacionamento colaborador empresa (*id*) como parâmetro na URI.

6.5 Answers

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso padrão.

1. Search Created:

- **Endpoint:** */answer/{id}*.
- **Método**: *GET*.
- Ele é responsável por consultar a resposta do colaborador para determinada questão do formulário por empresa. Necessário passar o identificador da resposta (*id*) como parâmetro na URI.

2. Save Self:
    - **Endpoint:** */answer*.
    - **Método**: *POST*.
    - Ele é responsável por salvar a resposta do colaborador para determinada questão do formulário por empresa. Necessário passar o identificador do relacionamento colaborador empresa (*businesUserId*), o identificador da questão respondida (*questionId*) e valor selecionado na resposta (*value*) no corpo da requisição.
3. Edit Created Value:
    - **Endpoint:** */answer/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por salvar a resposta do colaborador para determinada questão do formulário por empresa. Necessário passar o identificador do relacionamento colaborador empresa (*businesUserId*), o identificador da questão respondida (*questionId*) e valor selecionado na resposta (*value*) no corpo da requisição, além de passar o identificador da resposta (*id*) como parâmetro na URI.
4. Delete Self:
    - **Endpoint:** */answer/{id}*.
    - **Método**: *DELETE*.
    - Ele é responsável por excluir o relacionamento de colaborador com a empresa cadastrada pelo usuário autenticado no sistema. Necessário passar o identificador da resposta (*id*) como parâmetro na URI.

6.6 Categories

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso padrão.

1. Search:
    - **Endpoint:** */categories/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de categoria cadastrada. Necessário passar o identificador da categoria (*id*) como parâmetro na URI.
2. Search Name:
    - **Endpoint:** */categories/name.*
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de categoria cadastrada, informando o nome da categoria (*text*) no corpo da requisição.
3. Search Name Contains:
    - **Endpoint:** */categories/name/contains.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de categorias cadastradas que contenham parte do texto (*text*) informado no corpo da requisição em seus nomes.
4. List All:
    - **Endpoint:** */categories.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todas as categorias cadastradas no sistema.

6.7 Questions

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso padrão.

1. Search:
    - **Endpoint:** */questions/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de questão cadastrada. Necessário passar o identificador da questão (*id*) como parâmetro na URI.
2. Search Name:

- **Endpoint:** */questions/description.*
- **Método**: *GET*.
- Ele é responsável por consultar os dados de questão cadastrada, informando a descrição da questão (*text*) no corpo da requisição.

3. Search Name Contains:
    - **Endpoint:** */questions/description/contains.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de questões cadastradas que contenham parte do texto (*text*) informado no corpo da requisição em suas descrições.
4. List All:
    - **Endpoint:** */questions.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todas as questões cadastradas no sistema.

6.8 Users Admin

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso superior.

1. Search:
    - **Endpoint:** */admin/users/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados do usuário autenticado no sistema. Necessário passar o identificador do usuário (*id*) como parâmetro na URI.
2. Search Email:
    - **Endpoint:***/admin/users/email.*
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados do usuário cadastrado, informando o email do usuário (*text*) no corpo da requisição.
3. Search Email Contains:
    - **Endpoint:** */admin/users/email/contains.*
    - **Método**: *GET*.

- Ele é responsável por listar os dados de usuários cadastrados que contenham parte do texto (*text*) informado no corpo da requisição em seus emails.

4. Search Name Contains:
    - **Endpoint:** */admin/users/name/contains.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de usuários cadastrados que contenham parte do texto (*text*) informado no corpo da requisição em seus nomes.
5. Edit:
    - **Endpoint:** */admin/users/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por editar os dados do usuário cadastrados. Necessário passar o identificador do usuário (*id*) como parâmetro na URI.
6. Delete:
    - **Endpoint:** */admin/users/{id}*.
    - **Método**: *DELETE*.
    - Ele é responsável por excluir os dados de usuário cadastrados. Necessário passar o identificador do usuário (*id*) como parâmetro na URI.

6.9 Businesses Admin

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso superior.

1. Create:
    - **Endpoint:** */admin/businesses.*
    - **Método**: *POST*.
    - Ele é responsável por cadastrar uma empresa. Necessário passar nome da empresa (*name*), o CNPJ ou CPF (*cnpjCpf*) e o identificador do usuário que gerencia (*managerId*) no corpo da requisição.
2. Search:
    - **Endpoint:** */admin/businesses/{id}*.
    - **Método**: *GET*.

- Ele é responsável por consultar os dados de uma empresa cadastrada. Necessário passar o identificador da empresa (*id*) como parâmetro na URI.

3. Search CNPJ_CPF:
    - **Endpoint:** */admin/businesses/cnpjcpf.*
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de empresa cadastrada através do CNPJ ou CPF (*text*) informado no corpo da requisição.
4. Search Name Contains:
    - **Endpoint:** */admin/users/name/contains.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de empresas cadastradas que contenham parte do texto (*text*) informado no corpo da requisição em seus nomes.
5. List All:
    - **Endpoint:** */admin/businesses.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todas as empresas cadastradas.
6. Edit:
    - **Endpoint:** */admin/businesses/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por editar os dados de uma empresa. Necessário passar nome da empresa (*name*), o CNPJ ou CPF (*cnpjCpf*), o identificador do usuário que gerencia (*managerId*) no corpo da requisição e o identificador da empresa (*id*) como parâmetro na URI.
7. Delete:
    - **Endpoint:** */admin/businesses/{id}*.
    - **Método**: *DELETE*.
    - Ele é responsável por excluir os dados de uma empresa cadastrada. Necessário passar o identificador da empresa (*id*) como parâmetro na URI.

6.10 BusinessesUsers Admin

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso superior.

1. Search:
    - **Endpoint:** */admin/businessesuser/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados do colaborador por empresa cadastrada. Necessário passar o identificador do relacionamento colaborador empresa (*id*) como parâmetro na URI.
2. Save:
    - **Endpoint:** */admin/businessesuser*.
    - **Método**: *POST*.
    - Ele é responsável por adicionar um usuário como colaborador de uma empresa cadastrada, passando o email (*userEmail*) do colaborador e o identificador da empresa (*businessId*), o título para o convite a ser enviado no email do colaborador (*customTitle*) e a mensagem (*customMessage*) no corpo da requisição.
3. Delete:
    - **Endpoint:** */admin/businessesuser/{id}*.
    - **Método**: *DELETE*.
    - Ele é responsável por excluir o relacionamento de um colaborador com a empresa cadastrada. Necessário passar o identificador do relacionamento colaborador empresa (*id*) como parâmetro na URI.

6.11 Categories Admin

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso superior.

1. Save:
    - **Endpoint:** */admin/categories/{id}*.
    - **Método**: *POST*.
    - Ele é responsável por cadastrar uma categoria, passando o nome (*name*) no corpo da requisição.
2. Edit:

- **Endpoint:** */admin/categories/{id}*.
- **Método**: *PUT*.
- Ele é responsável por editar os dados de uma categoria, passando o nome (*name*) no corpo da requisição. Necessário passar o identificador da categoria (*id*) como parâmetro na URI.

3. Edit Position:
    - **Endpoint:** */admin/categories/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por editar a posição de uma categoria, passando o nome (*name*) e a posição (*position*) no corpo da requisição. Necessário passar o identificador da categoria (*id*) como parâmetro na URI.
4. Search:
    - **Endpoint:** */admin/categories/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de categoria cadastrada. Necessário passar o identificador da categoria (*id*) como parâmetro na URI.
5. Search Name:
    - **Endpoint:** */admin/categories/name.*
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de categoria cadastrada informando o nome da categoria (*text*) no corpo da requisição.
6. Search Name Contains:
    - **Endpoint:** */admin/categories/name/contains.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de categorias cadastradas que contenham parte do texto (*text*) informado no corpo da requisição em seus nomes.
7. List All:
    - **Endpoint:** */admin/categories*.
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todas as categorias cadastradas no sistema.

6.12 Questions Admin

Os **endpoints** apresentados dentro deste subtópico estão relacionados aos serviços disponíveis para usuários com nível de acesso superior.

1. Save:
    - **Endpoint:** */admin/question/{id}*.
    - **Método**: *POST*.
    - Ele é responsável por cadastrar uma questão, passando a descrição (*description*) e o identificador da categoria (*categoryId*) no corpo da requisição.
2. Edit:
    - **Endpoint:** */admin/question/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por editar os dados de uma questão, passando a descrição (*description*) e o identificador da categoria (*categoryId*) no corpo da requisição. Necessário passar o identificador da questão (*id*) como parâmetro na URI.
3. Edit Position:
    - **Endpoint:** */admin/question/{id}*.
    - **Método**: *PUT*.
    - Ele é responsável por editar a posição de uma questão, passando a descrição (*description*), o identificador da categoria (*categoryId*) e a posição (position) no corpo da requisição. Necessário passar o identificador da questão (*id*) como parâmetro na URI.
4. Search:
    - **Endpoint:** */admin/questions/{id}*.
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de questão cadastrada. Necessário passar o identificador da questão (*id*) como parâmetro na URI.
5. Search Name:
    - **Endpoint:** */admin/questions/description.*
    - **Método**: *GET*.
    - Ele é responsável por consultar os dados de questão cadastrada informando a descrição da questão (*text*) no corpo da requisição.

6. Search Name Contains:
    - **Endpoint:** */admin/questions/description/contains.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de questões cadastradas que contenham parte do texto (*text*) informado no corpo da requisição em suas descrições.
7. List All:
    - **Endpoint:** */admin/questions.*
    - **Método**: *GET*.
    - Ele é responsável por listar os dados de todas as questões cadastradas no sistema.

7. TESTES

A etapa de testes é uma das mais importantes no processo de desenvolvimento de produtos de software, segundo Tosetto (2008). Através dela, podemos avaliar a forma que o sistema se comporta para então concluir se é necessário realizar alterações, melhoria de performance, correções de bugs ou se o software conseguiu alcançar o padrão de funcionamento esperado. Este é um processo constante durante a implementação da **API**, onde pode-se testar diversos componentes desse sistema, sendo eles individuais ou de interação com um ou mais componentes.

Melo (2015) destaca a diversidade de abordagens para a realização de testes em softwares, enfatizando que diversas técnicas foram historicamente aprimoradas por meio de linguagens estruturadas, as quais desempenham um papel crucial. Entre as categorias de testes mencionadas, incluem-se: caixa-branca, caixa-preta, testes alpha, beta e gama, teste de unidade, teste de integração, teste de sistema, teste de verificação e teste de aceitação. Já os testes de verificação são realizados comumente por testadores humanos, seguindo planos de teste e abordagens exploratórias, de usabilidade, integração do usuário e aceitação do usuário.

No projeto Business Eval, foi escolhido o método manual de teste de verificação de software, que foi realizado através do console da **IDE STS**, e por meio do software de **Postman**, uma **API** cliente que permite realizar os testes de forma prática usando interface gráfica. Os casos de teste realizados foram exportados no formato **JSON** para permitir a reprodução em outros dispositivos, também servindo como backup.

8. RESULTADOS

No presente segmento, serão expostas demonstrações de requisições e respostas do sistema Business-Eval, as quais foram realizadas por meio da ferramenta Postman, cujo propósito é facilitar a execução e visualização de requisições **HTTP** e **HTTPs**. A seguir, serão disponibilizados “prints" dos resultados das execuções. É importante ressaltar que nem todos os **endpoints** disponíveis no sistema estão representados aqui devido à vasta quantidade de requisições com características semelhantes. Portanto, as demonstrações se concentram nos **endpoints** mais relevantes para a utilização do sistema.

8.1 Login - Require Login

Nesta requisição é tratada a solicitação de código de login ao sistema, que realiza o envio do código para o e-mail informado (*email*), retornando os dados do usuário com o e-mail cadastrado.

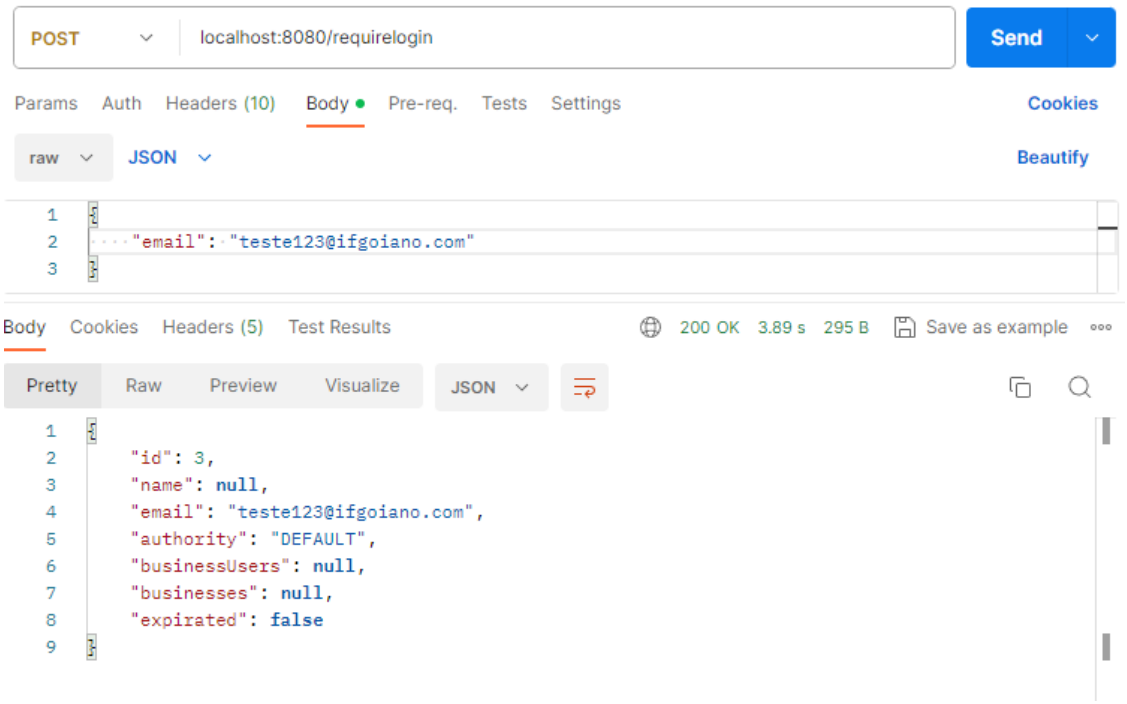

Figura 13. Captura de Tela: Resultados (Login - Require Login)

8.2 Login - Login

Nesta requisição é tratada a autenticação do usuário através e-mail cadastrado (*email*) e do código de login (*loginCode*), retornando o token de autenticação que será utilizado no cabeçalho das demais requisições.

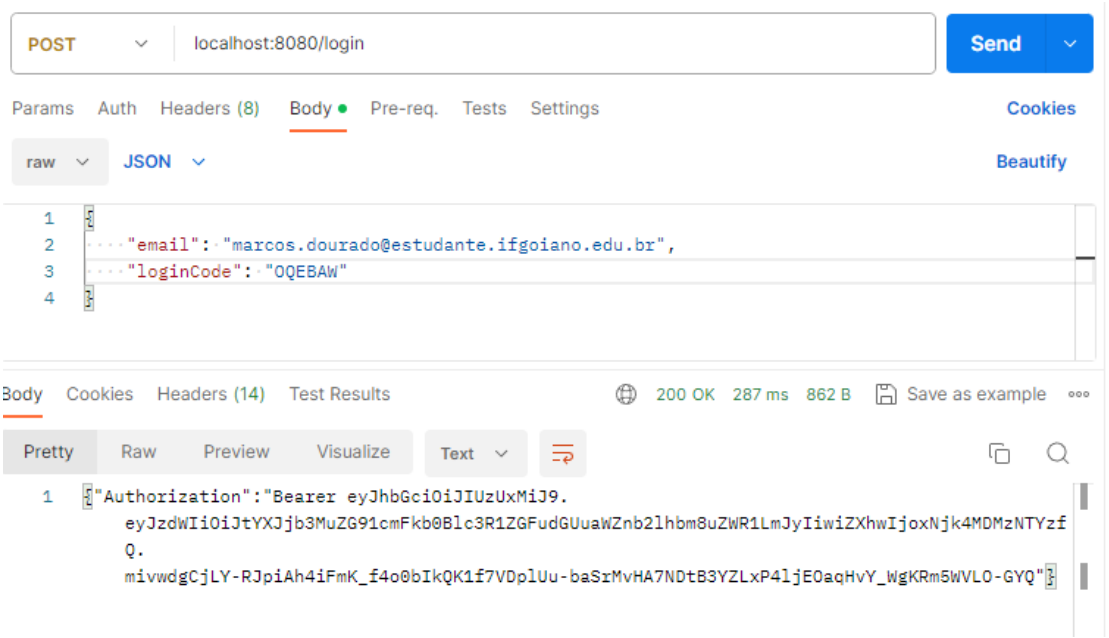

Figura 14. Captura de Tela: Resultados (Login - Login)

8.3 Category -List all

Nesta requisição é solicitada uma lista com todas as categorias de questões cadastradas no sistema, retornando a lista de categorias com as suas respectivas questões.

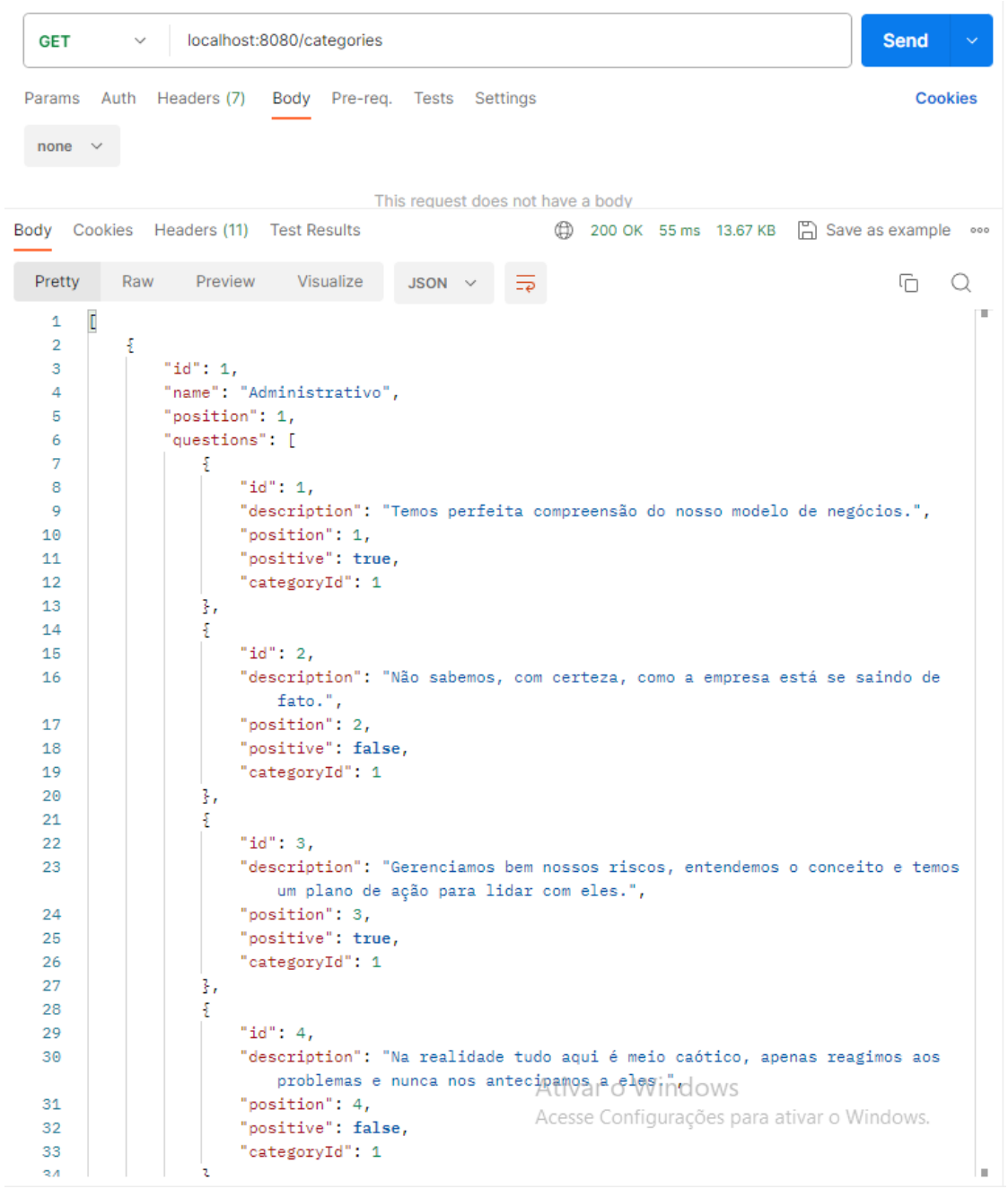

Figura 15. Captura de Tela: Resultados (Category -List all)

8.4 Question - List all

Nesta requisição é solicitada uma lista com todas as questões cadastradas no sistema, retornando uma lista de todas as questões cadastradas com identificador de categoria (*categoryId*).

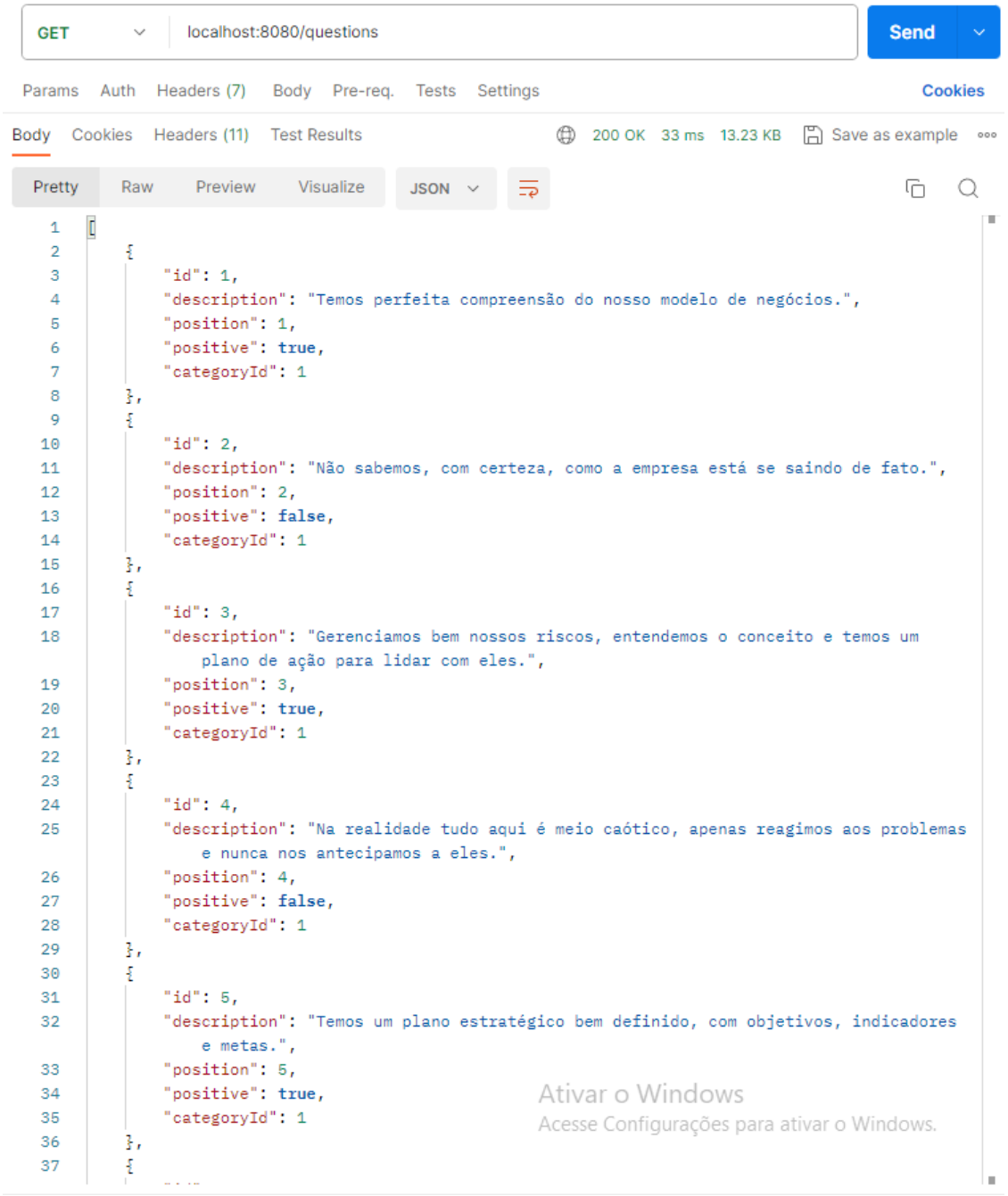

Figura 16. Captura de Tela: Resultados (Question -List all)

8.5 User - Search self

Nesta requisição é solicitado os dados do usuário autenticado, retornando os dados do usuário, juntamente às empresas que ele gerencia e os vínculos de colaborador em outras empresas.

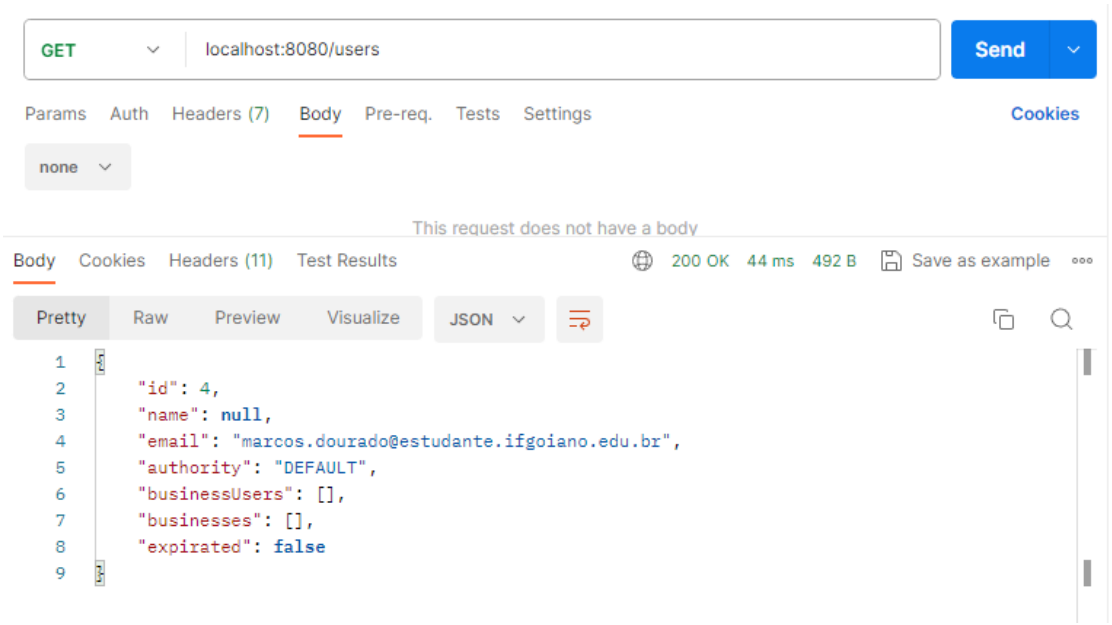

Figura 17. Captura de Tela: Resultados (User - Search self)

8.6 User - Edit self

Nesta requisição é solicitado a edição do usuário autenticado em que é passado o identificador do usuário no endpoint, além dos novos dados do usuário no corpo da requisição, retornando os dados do atualizados do usuário.

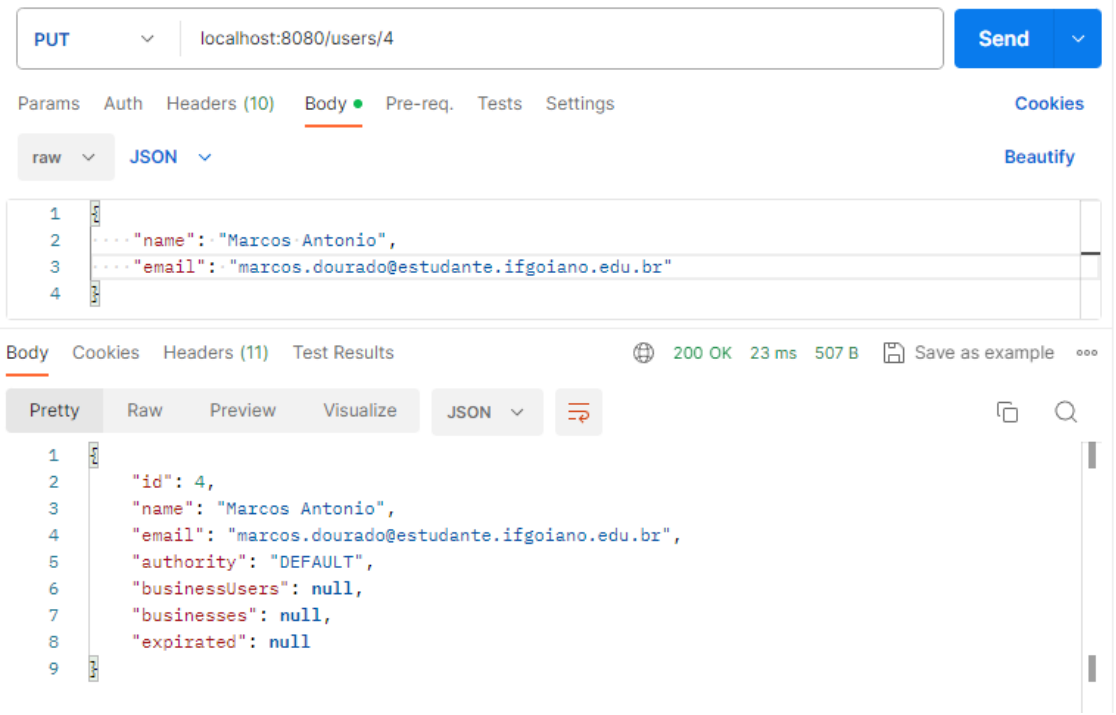

Figura 18. Captura de Tela: Resultados (User - Edit self)

8.7 User - Delete self

Nesta requisição é solicitado a exclusão do usuário autenticado em que é passado o dados de autenticação (*email* e *loginCode*) do usuário no corpo da requisição, retornando o status aceito (202) se estiver correto.

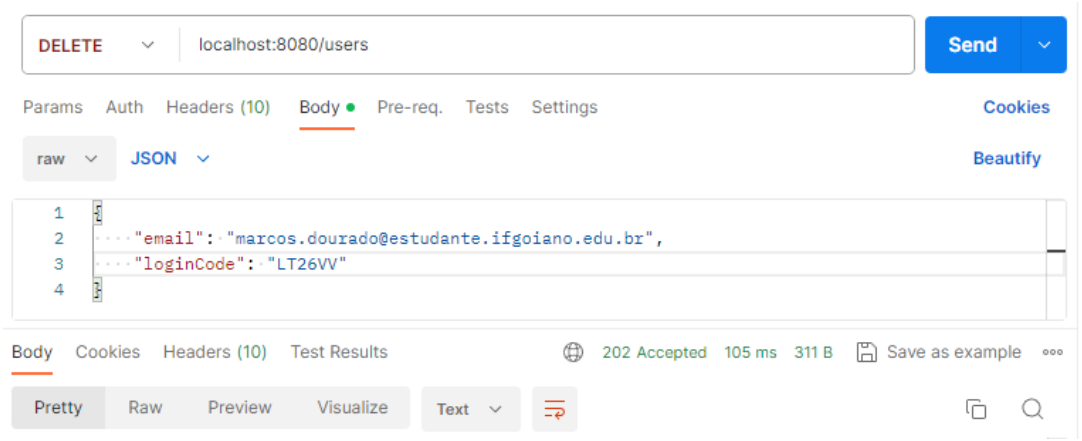

Figura 19. Captura de Tela: Resultados (User - Delete self)

8.8 Business - Create

Nesta requisição é solicitado a criação de uma empresa pelo usuário gestor, são passados os dados da empresa (*name* e *cnpjCpf*) no corpo da requisição, retornando os dados da empresa criada.

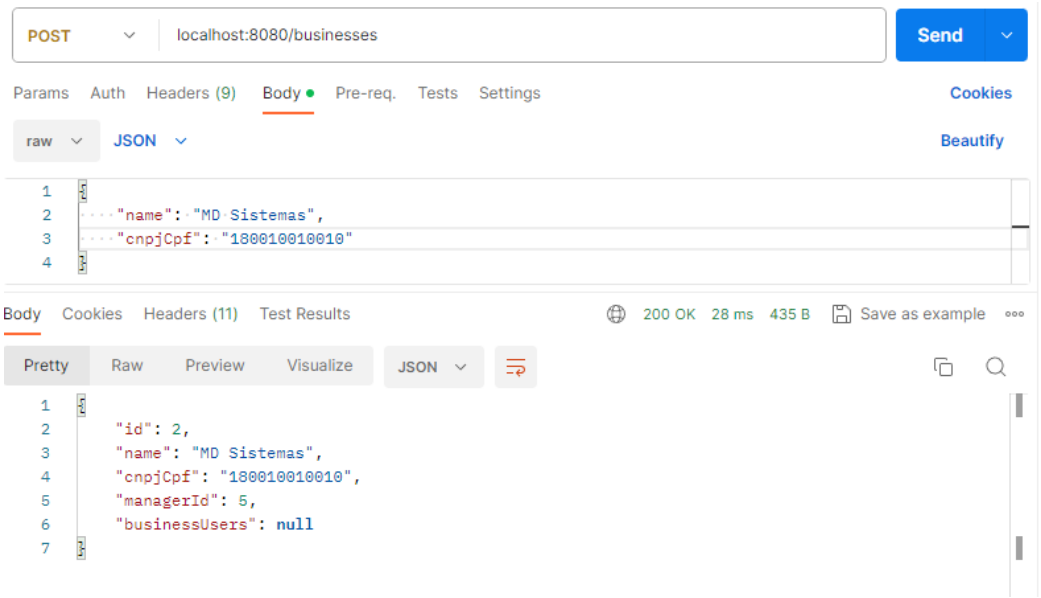

Figura 20. Captura de Tela: Resultados (Business - Create)

8.9 Business - List Created

Nesta requisição é solicitado a lista de empresas criadas pelo usuário autenticado, retornando os dados das empresas criadas.

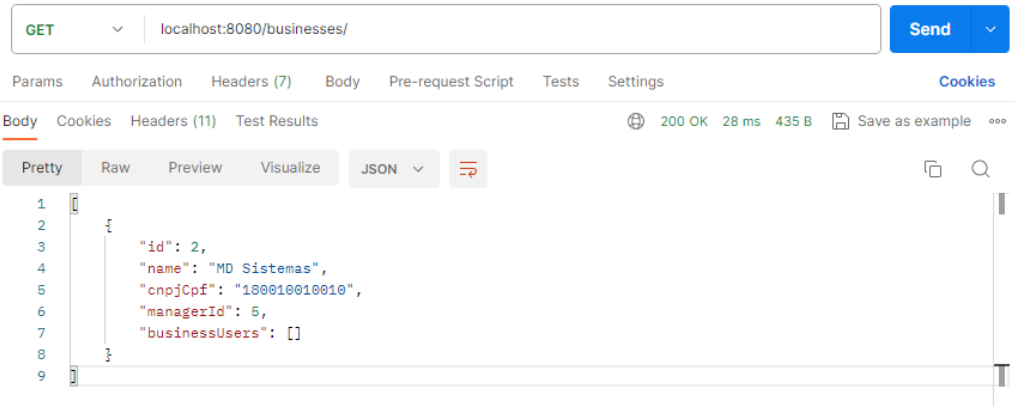

Figura 21. Captura de Tela: Resultados (Business - List Created)

8.10 BusinessUser - Search

Nesta requisição é solicitado a consulta de colaborador pelo usuário gestor de empresa, em que é passado o identificador de relacionamento usuário-empresa no endpoint, retornando os dados do relacionamento do colaborador

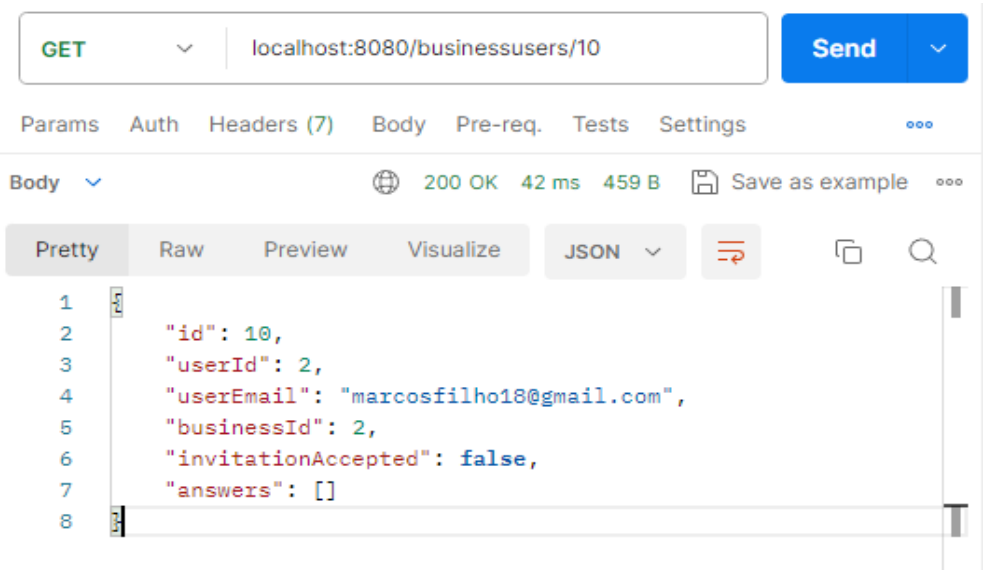

Figura 22. Captura de Tela: Resultados (BusinessUser - Search)

8.11 BusinessUser - Add self

Nesta requisição é adicionado pelo usuário gestor de empresa, ele mesmo, como colaborador de uma empresa, onde é passado o identificador da empresa (*businessId*) no corpo da requisição, retornando os dados do relacionamento usuário e empresa.

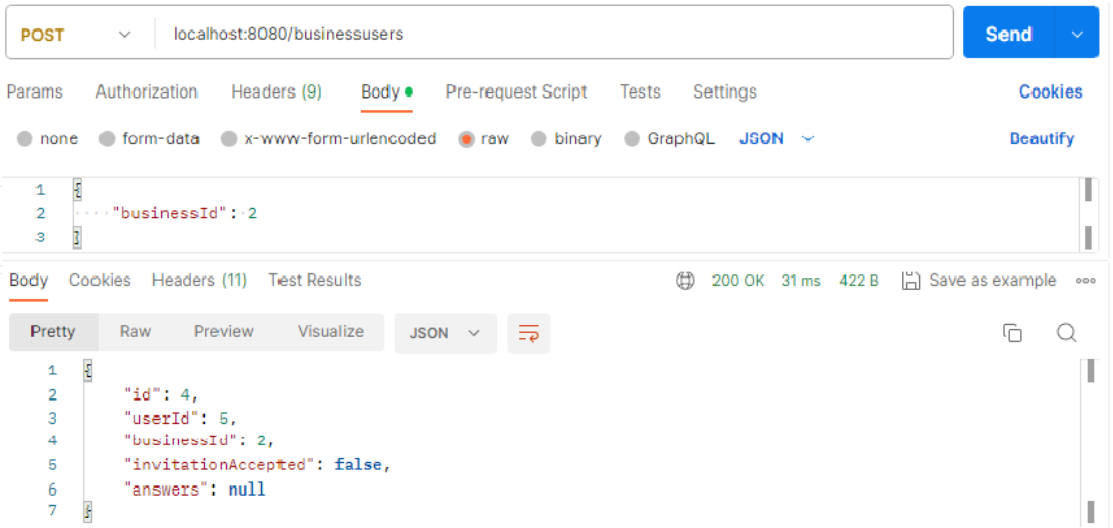

Figura 23. Captura de Tela: Resultados (BusinessUser - Add self)

8.12 BusinessUser - Accepted Invitation

Nesta requisição é alterado o status do usuário colaborador de uma empresa informando que o convite foi aceito, onde é passado o identificado do relacionamento usuário e empresa no corpo da requisição, retornando os dados do relacionamento.

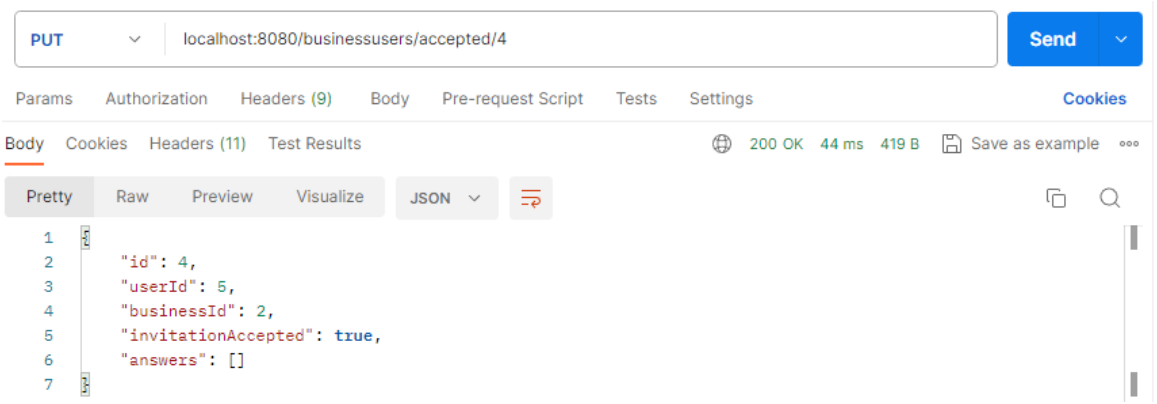

Figura 24. Captura de Tela: Resultados (BusinessUser - Accepted Invitation)

8.13 Answer - Search created

Nesta requisição é solicitada uma consulta de uma resposta do usuário logado em que é passado o identificador da resposta no endpoint, retornando os dados da resposta.

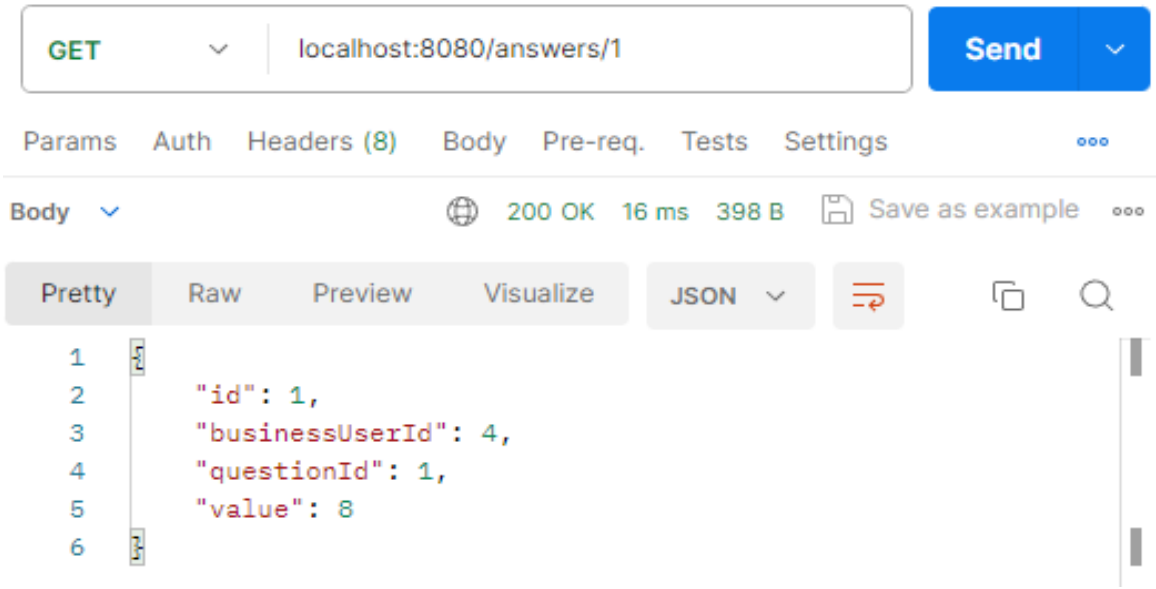

Figura 25. Captura de Tela: Resultados (Answer - Search created)

8.14 Answer - Save self

Nesta requisição é adicionada uma resposta do usuário logado em que é passado o identificador do relacionamento usuário e empresa (*businessUserId*), o identificador da questão (*questionId*) e o valor da resposta (*value*), retornando os dados da resposta.

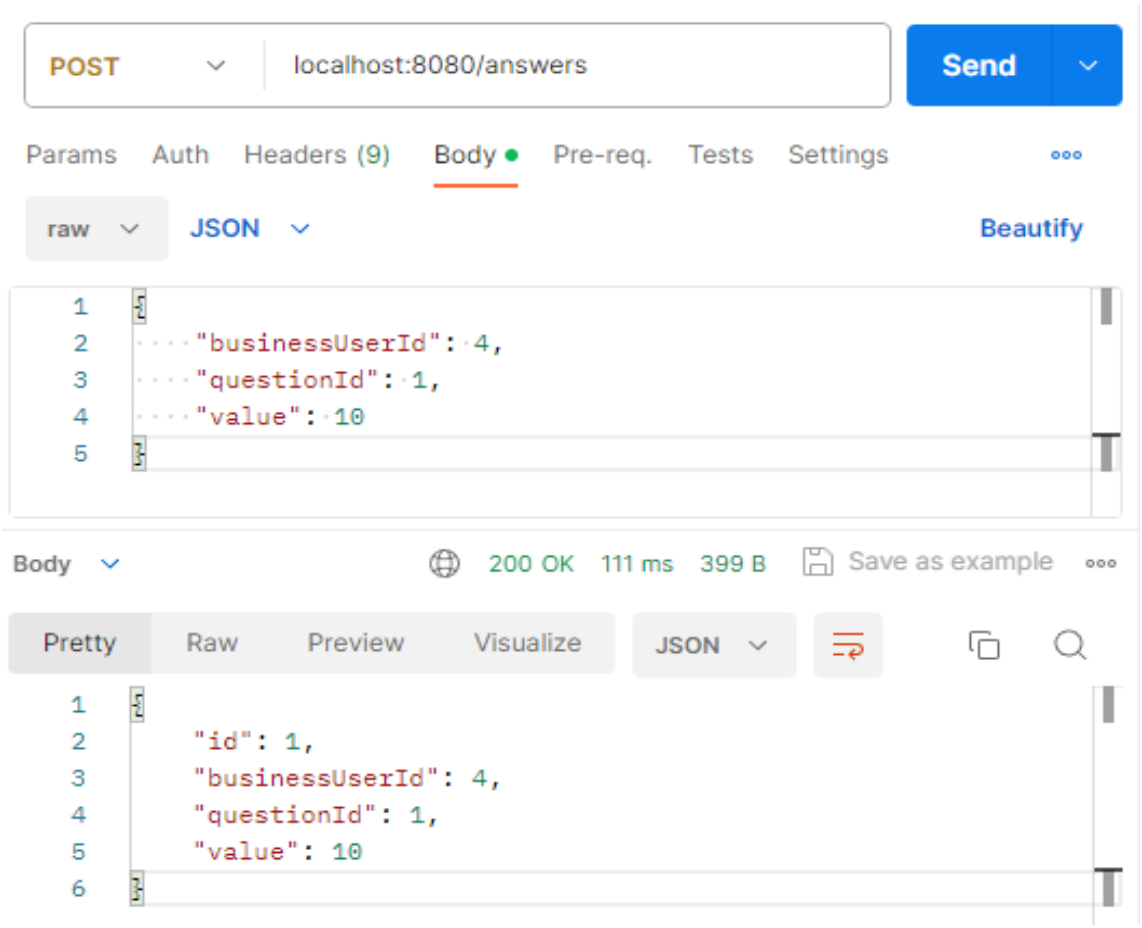

Figura 26. Captura de Tela: Resultados (Answer - Save self)

8.15 Answer - Edit created

Nesta requisição é solicitada a edição de uma resposta do usuário logado em que é passado o identificador da resposta no endpoint e os dados da resposta no corpo da requisição, retornando os dados da resposta.

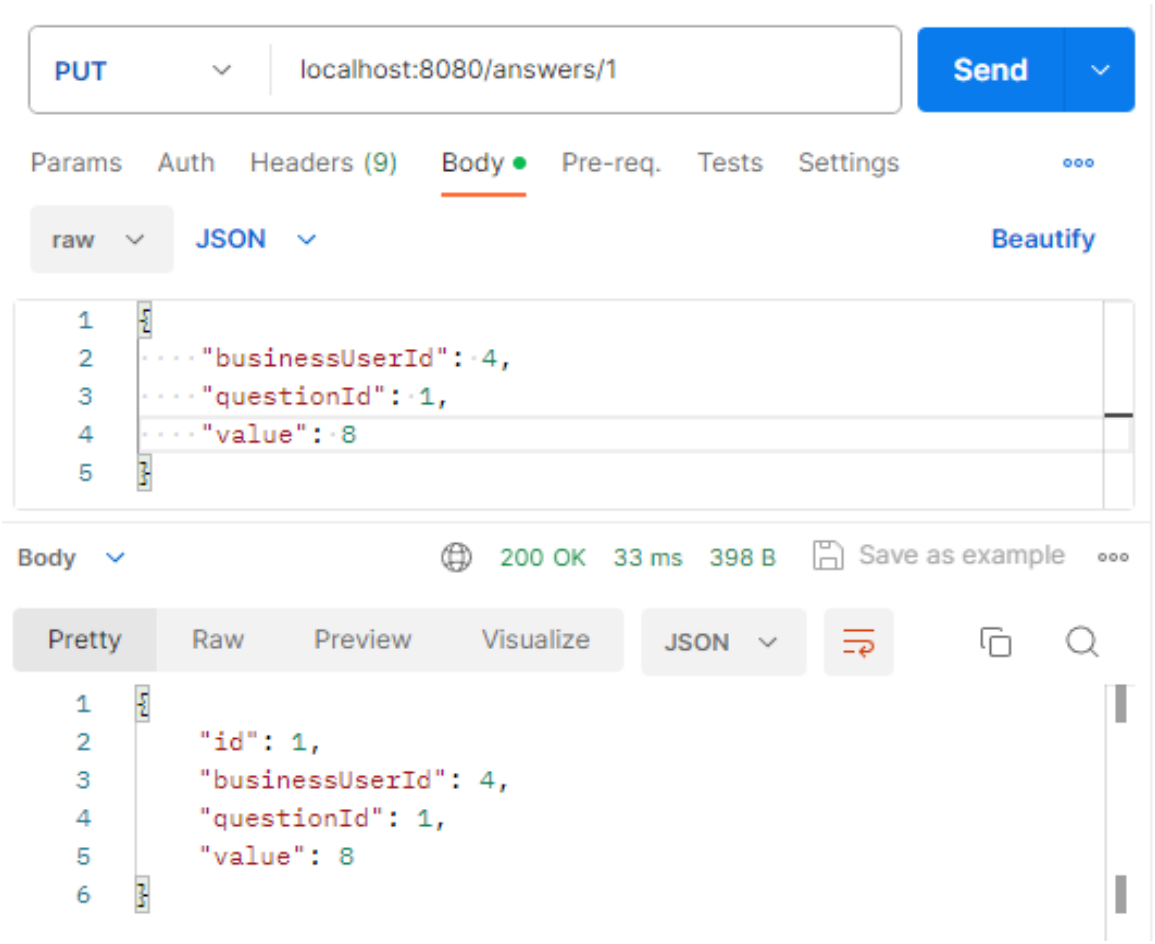

Figura 27. Captura de Tela: Resultados (Answer - Edit created)

8.16 Category admin - Create

Nesta requisição é adicionado uma nova categoria de questões pelo usuário gerente do sistema, em que é passado o nome da categoria (*name*) no corpo da requisição, retornando os dados da categoria.

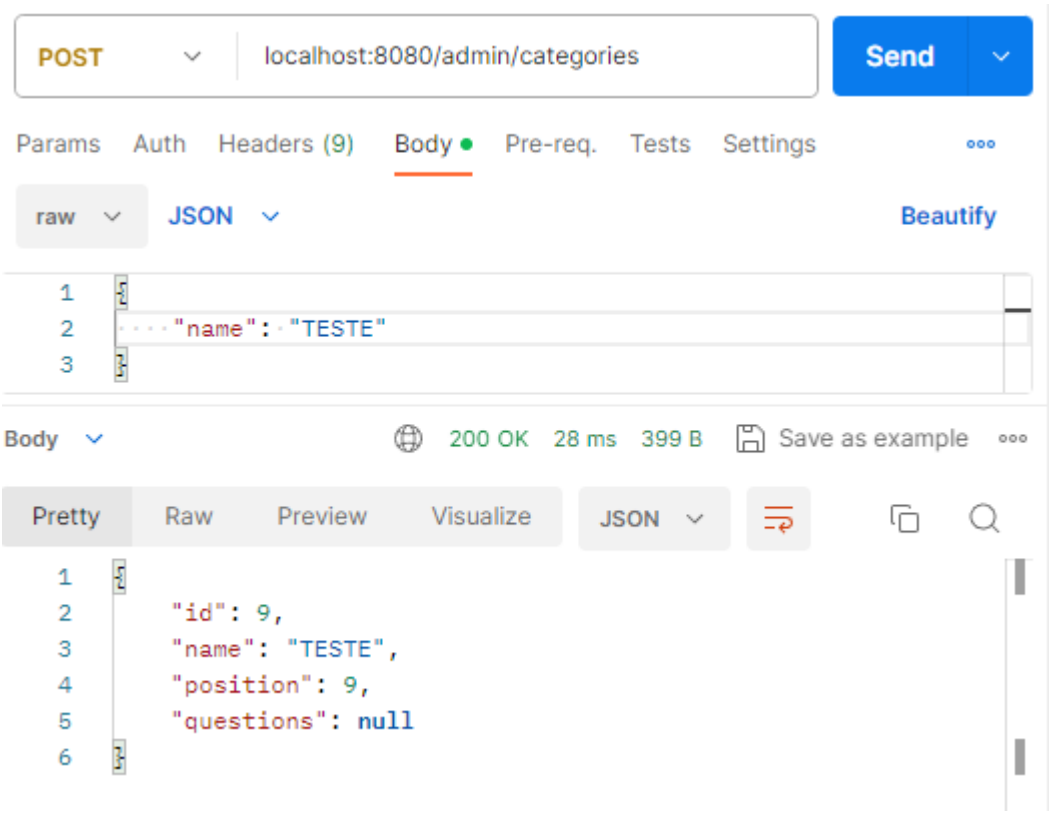

Figura 28. Captura de Tela: Resultados (Category admin - Create)

8.17 Category admin - Edit

Nesta requisição é solicitada a edição dos dados de uma categoria de questões pelo usuário gerente do sistema, em que são passados o identificador da categoria no endpoint, o nome da categoria (*name*) e a posição (*position*) no corpo da requisição, retornando os dados da categoria.

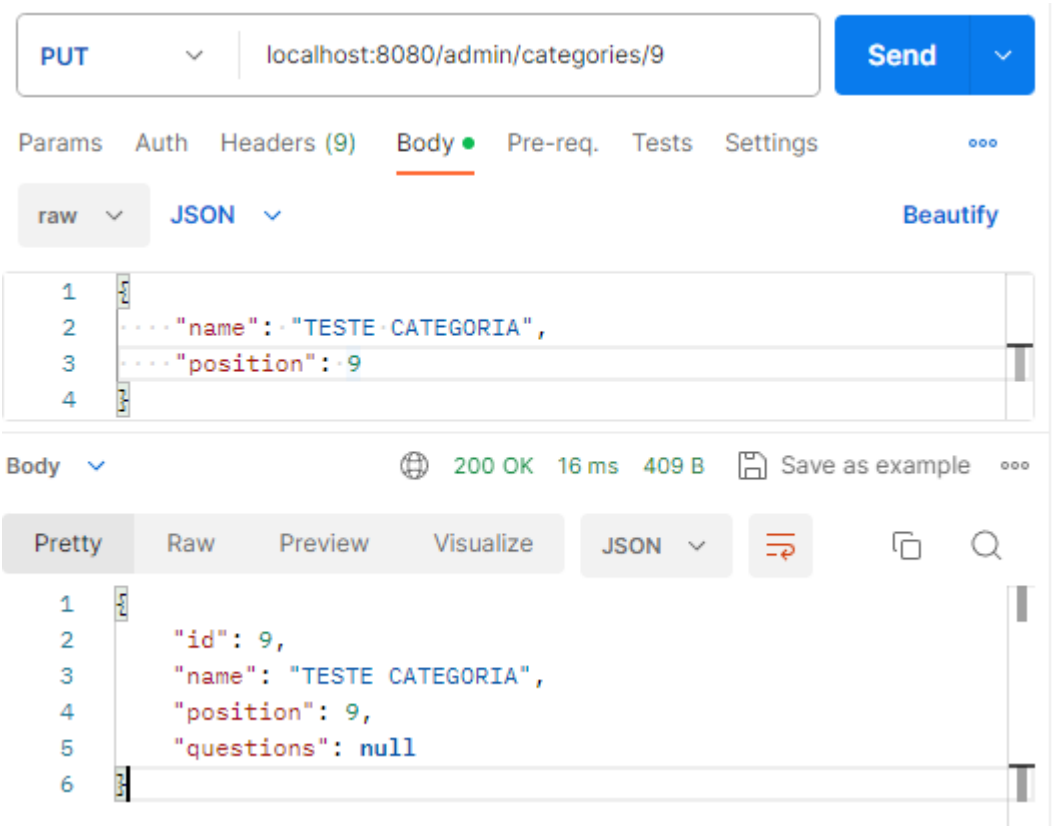

Figura 29. Captura de Tela: Resultados (Category admin - Edit)

8.18 Category admin - Delete

Nesta requisição é solicitada a exclusão de uma categoria de questões pelo usuário gerente do sistema, em que é passado o identificador da categoria no endpoint, retornando o status aceito (202).

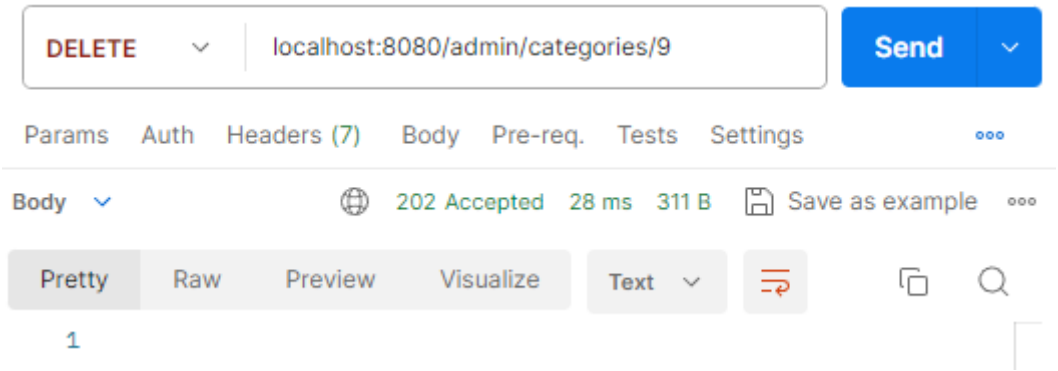

Figura 30. Captura de Tela: Resultados (Category admin - Delete)

8.19 Category admin - Edit position

Nesta requisição é solicitada a edição da posição de uma categoria de questões pelo usuário gerente do sistema em que são passados o identificador da categoria no endpoint, o nome da categoria (*name*) e a posição nova (*position*) no corpo da requisição, retornando os dados da categoria.

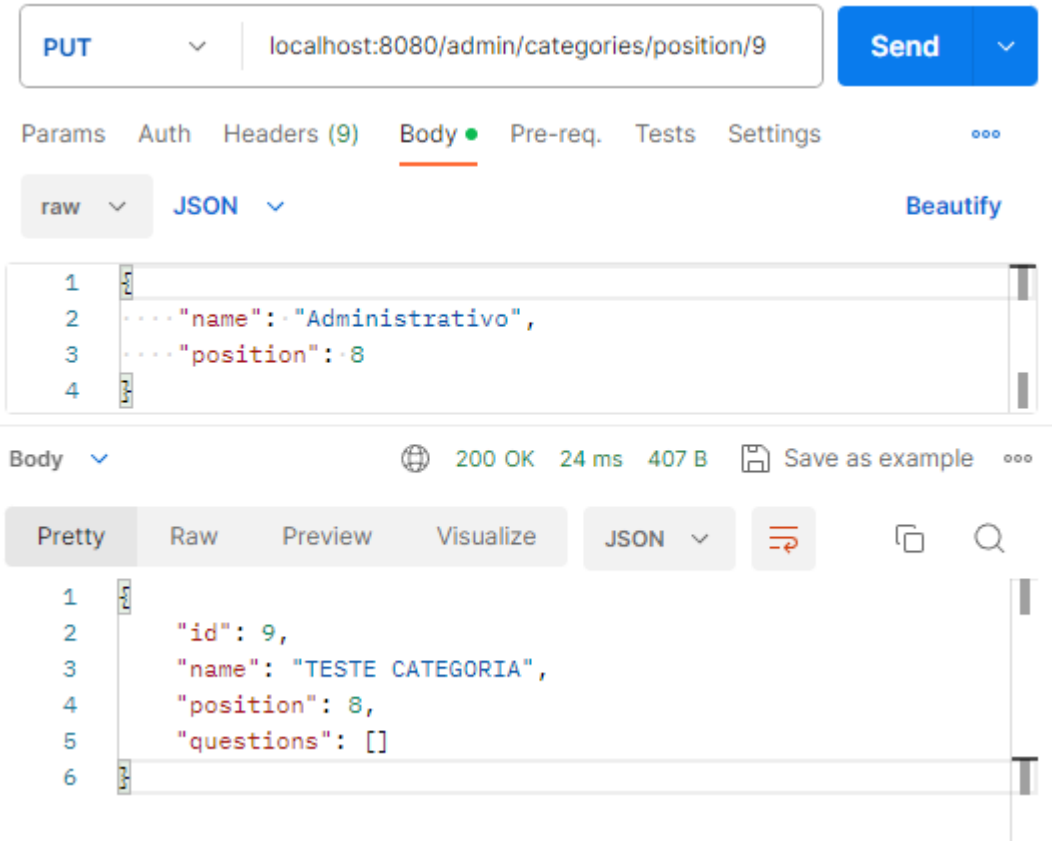

Figura 31. Captura de Tela: Resultados (Category admin - Edit position)

8.20 Question admin - Create

Nesta requisição é adicionado uma nova questão pelo usuário gerente do sistema em que são passadas a descrição (*description*), a conotação, se for positiva (*positive*) e o identificador da categoria (*categoryId*) no corpo da requisição, retornando os dados da questão.

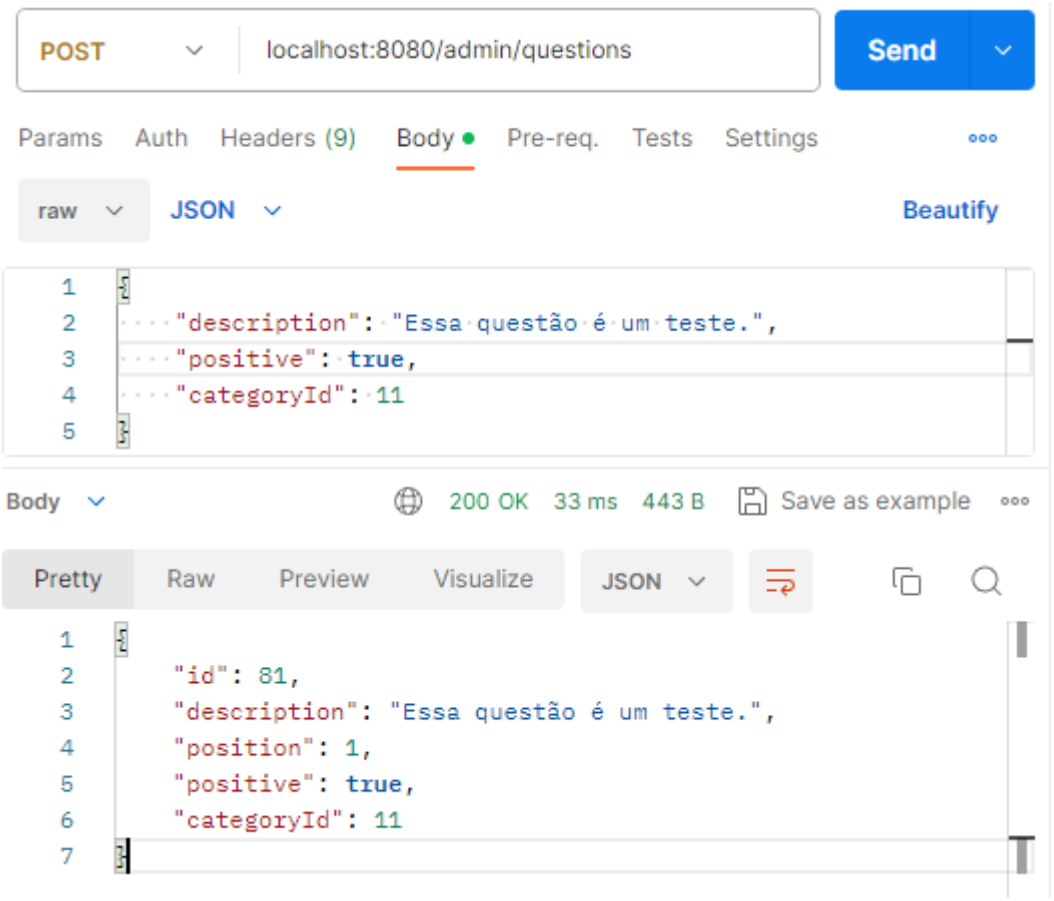

Figura 32. Captura de Tela: Resultados (Question admin - Create)

8.21 Question admin - Edit

Nesta requisição é solicitada a edição de uma questão pelo usuário gerente do sistema em que são passados o identificador da questão no endpoint, a descrição (*description*), a posição (*position*), a conotação, se for positiva (*positive*) e o identificador da categoria (*categoryId*) no corpo da requisição, retornando os dados da questão.

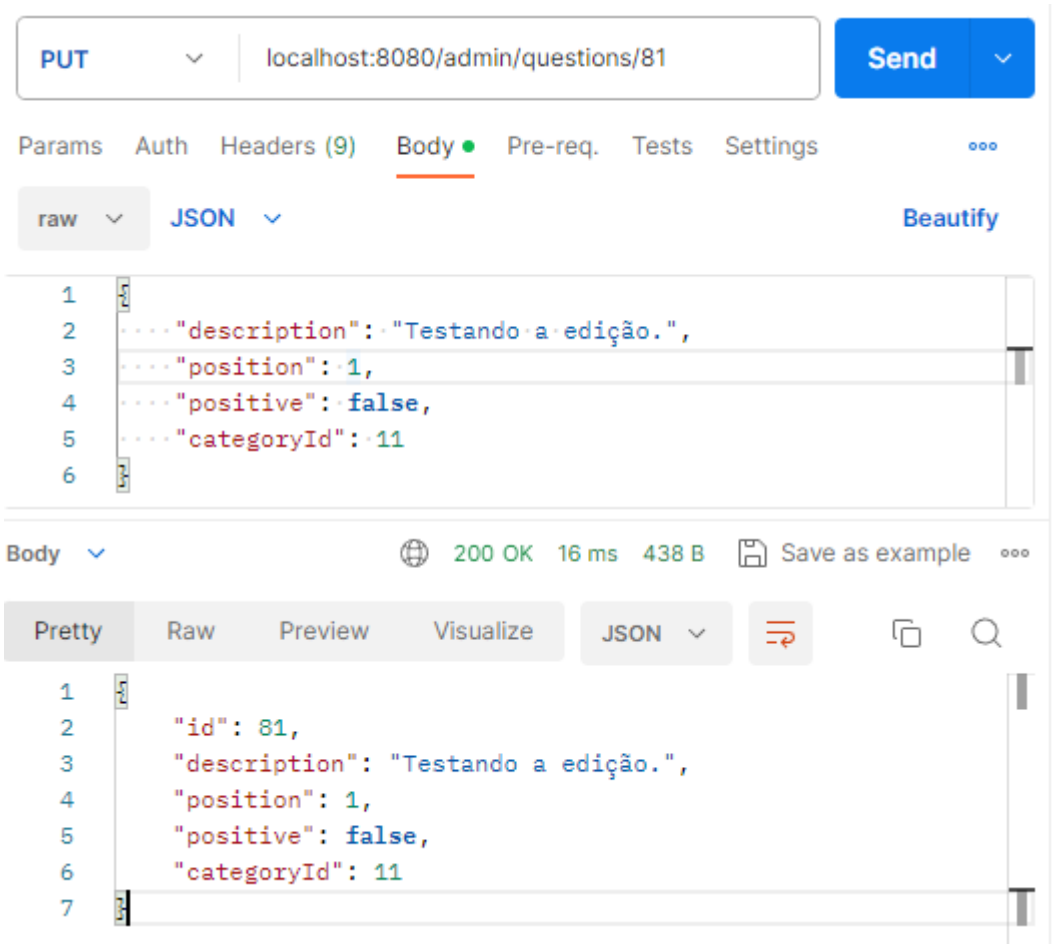

Figura 33. Captura de Tela: Resultados (Question admin - Edit)

8.22 Question admin - Delete

Nesta requisição é solicitada a exclusão de uma questão pelo usuário gerente do sistema, em que é passado o identificador da questão no endpoint, retornando o status aceito (202).

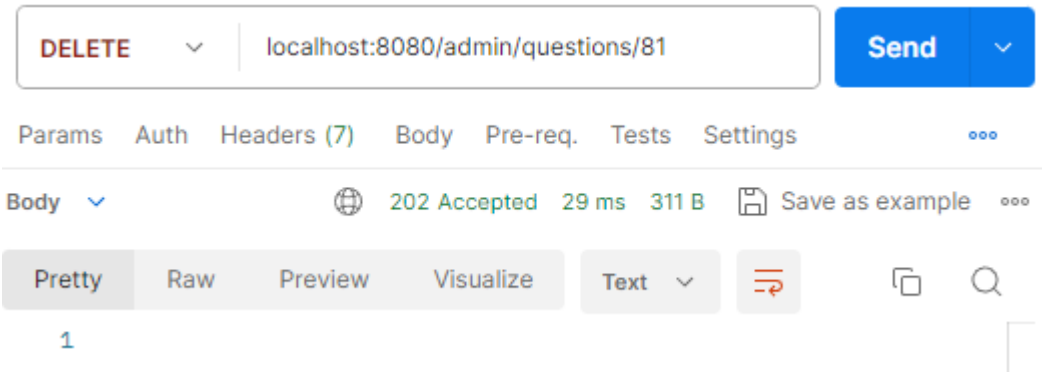

Figura 34. Captura de Tela: Resultados (Question admin - Delete)

8.23 Question admin - Edit position

Nesta requisição é solicitada a edição da posição de uma questão pelo usuário gerente do sistema, em que são passados o identificador da questão no endpoint, a descrição (*description*) e a posição nova (*position*), a conotação, se for positiva (*positive*) e o identificador da categoria (*categoryId*) no corpo da requisição, retornando os dados da questão.

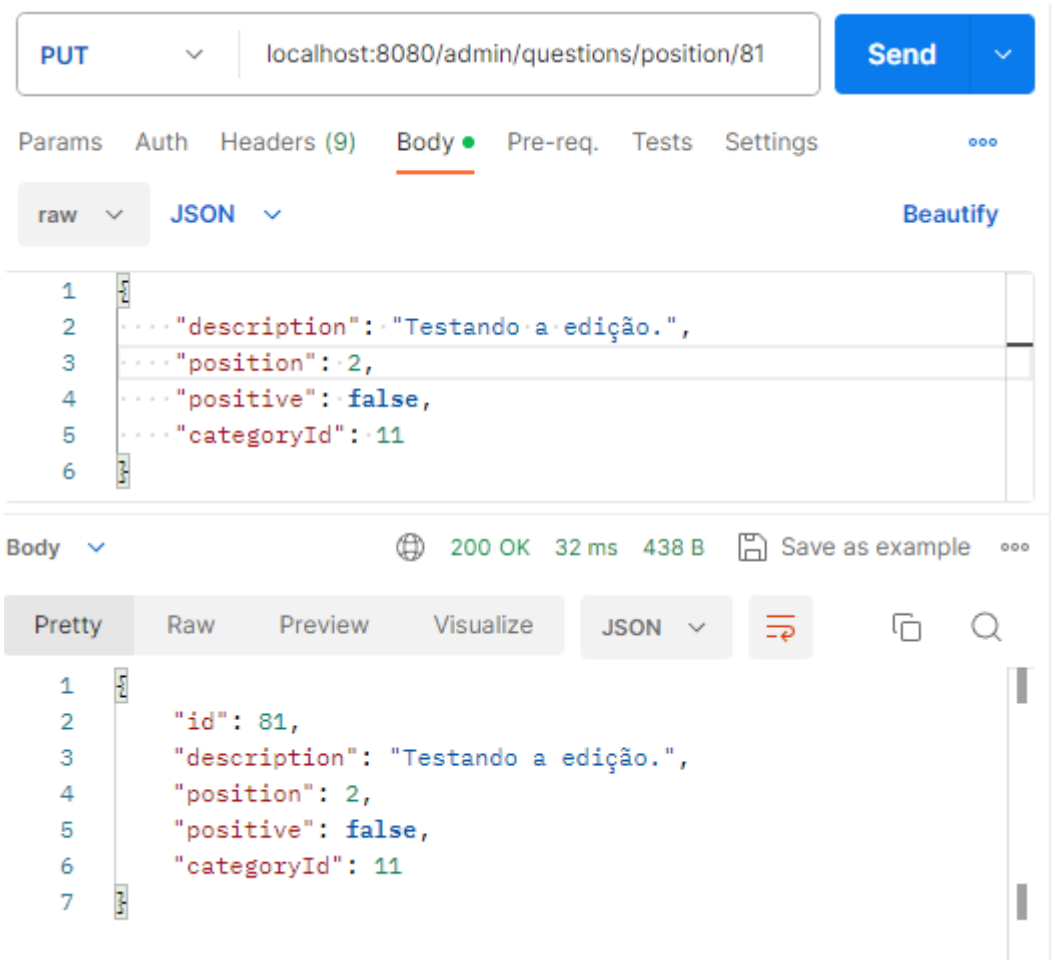

Figura 35. Captura de Tela: Resultados (Question admin - Edit position)

8.24 User admin - Create

Nesta requisição é adicionado um novo usuário pelo usuário gerente do sistema, em que são passados o nome (*name*) e o e-mail (*email*) no corpo da requisição, retornando os dados do usuário.

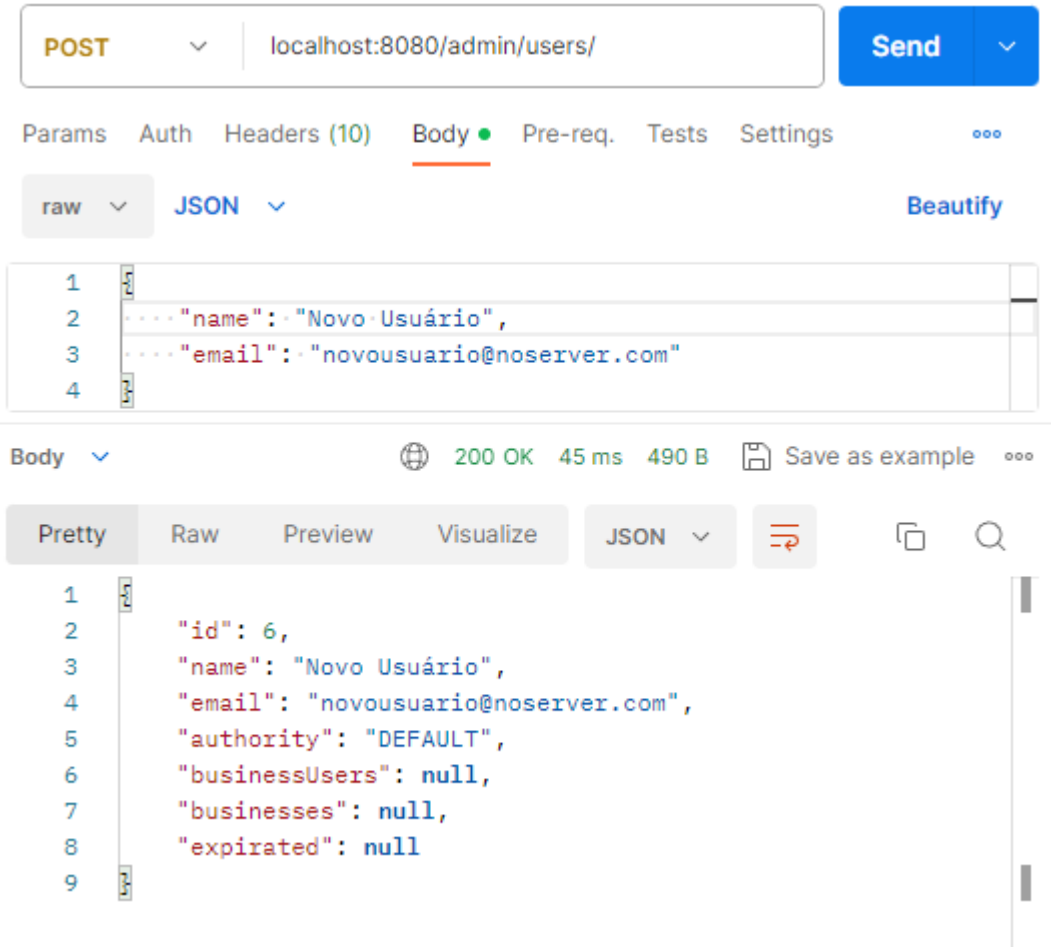

Figura 36. Captura de Tela: Resultados (User admin - Create)

8.25 User admin - List all

Nesta requisição é solicitada a lista de usuários pelo usuário gerente do sistema, retornando uma lista com os dados de todos os usuários.

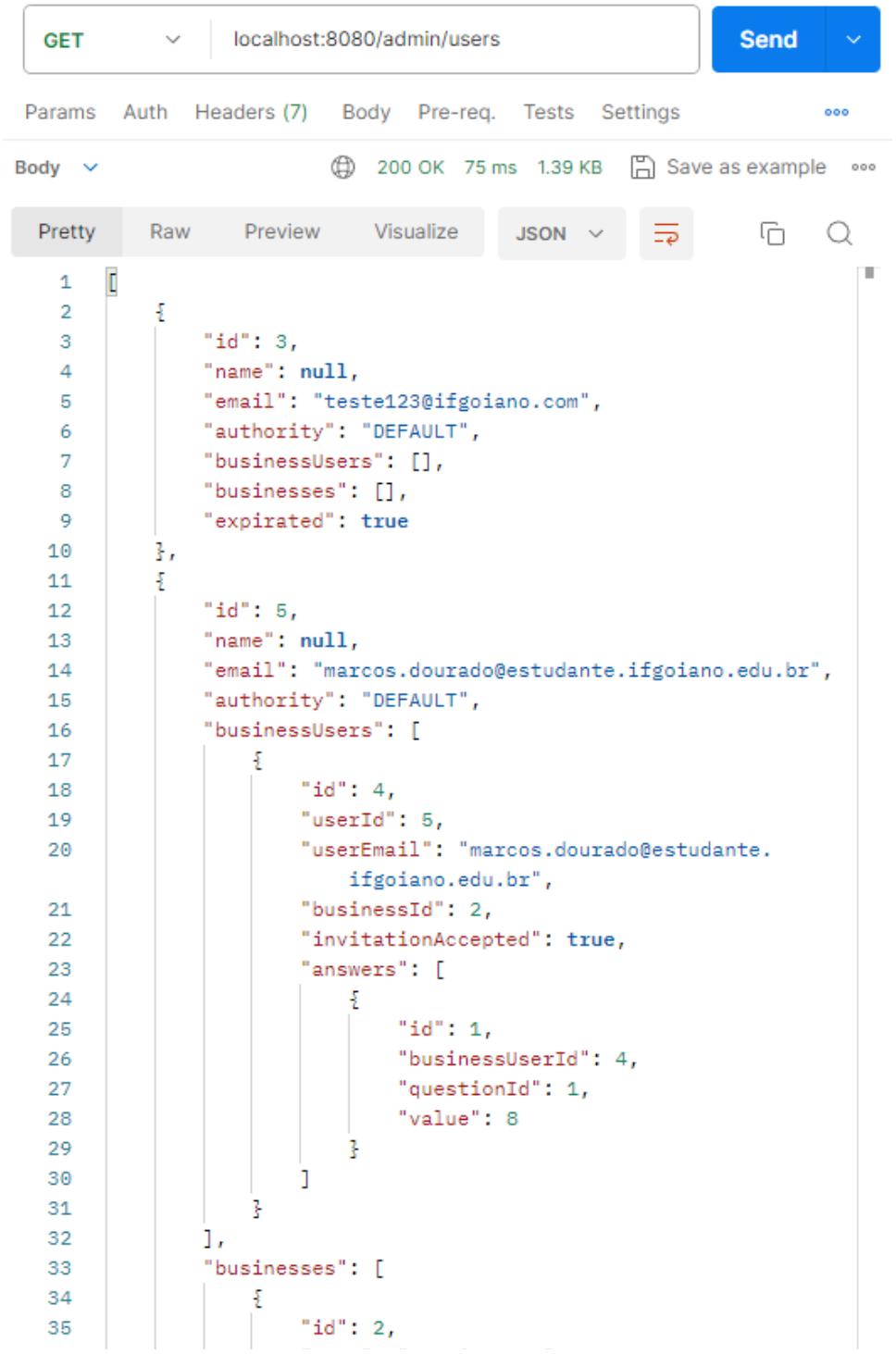

Figura 37. Captura de Tela: Resultados (User admin - List all)

8.26 User admin - Set authority

Nesta requisição é solicitada a edição das autorizações de acesso um usuário pelo usuário gerente do sistema, em que são passados o identificador da questão no endpoint e a nova autorização (*text*) no corpo da requisição, retornando os dados do usuário.

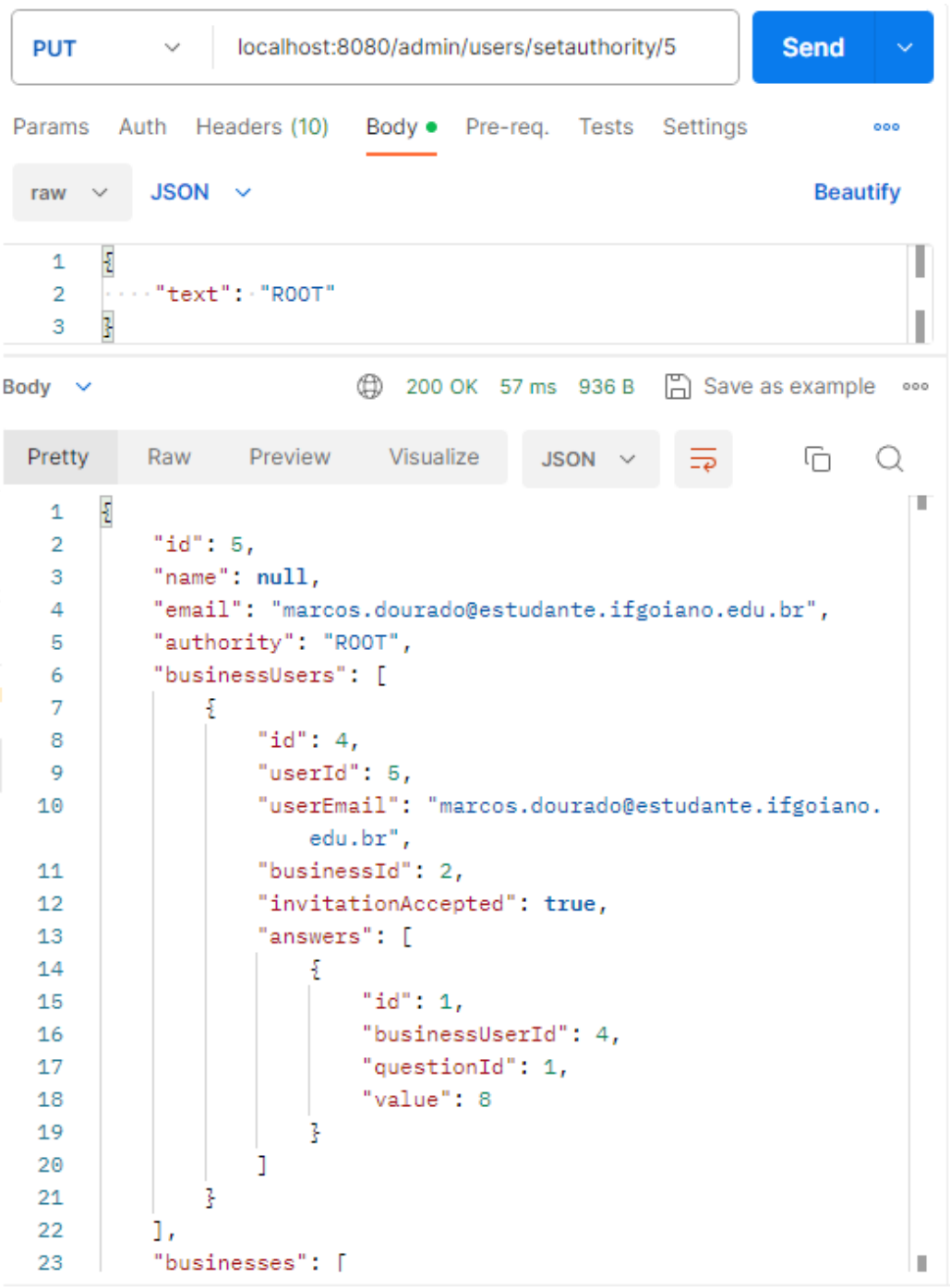

Figura 38. Captura de Tela: Resultados (User admin - Set authority)

8.27 User admin - Delete

Nesta requisição é solicitada a exclusão de um usuário pelo usuário gerente do sistema, em que é passado o identificador do usuário no endpoint, retornando o status aceito (202).

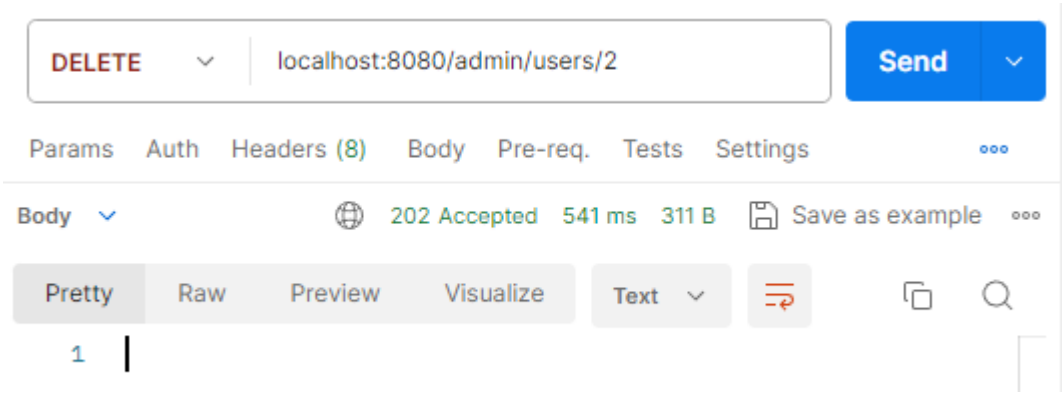

Figura 39. Captura de Tela: Resultados (User admin - Delete)

9. CONCLUSÃO

O projeto aqui descrito tem por objetivo documentar uma **API Rest**, fitando nos mecanismos de processamento de dados, desenvolvimento de funcionalidades que possibilitam as operações de leitura, cadastro, edição e exclusão de diversas entidades que compõem um sistema completo e que capta dados de diversos setores de uma empresa, possibilita a consulta personalizada com a justificativa de gerar graficamente informações que auxiliam na tomada de medidas estratégicas para aprimoramento do ambiente empresarial.

A priori, com o intuito de suprir a demanda de uma aplicação que possibilite, de forma simples e intuitiva, a captação de dados empresariais, esse projeto auxiliaria no aprimoramento dos ambientes empresariais, através de estratégias que impactam no desenvolvimento desse setor importante para a economia.

Para compreender a importância desse trabalho, foram realizadas pesquisas de atores, tais como Dos Santos (2020), Asencio (2015) e Chiusoli (2017) a respeito dos impactos estratégicos gerados pelo diagnóstico realizado pela percepção do colaborador, o que possibilitou inquirir sobre a relevância do projeto para o setor empresarial.

Devido a relevância, é esperado que a implantação desse sistema possa suprir a necessidade de uma ferramenta tecnológica que agilize o processo de diagnóstico, que em muitos casos é realizado em formulários físicos ou eletrônicos que armazenam os dados em planilhas eletrônicas, onde a consulta e normalização dos dados acaba sendo complexa.

Para facilitar todo o processo de diagnóstico, a ferramenta Business Eval permite que, por meio da plataforma web, seja realizado o cadastro rápido e simplificado das empresas e colaboradores, como também o preenchimento de formulário de forma simples, dinâmica e intuitiva, contando com um mecanismo de visualização instantânea das informações por meio gráfico e a manutenção dos dados por meio do acesso de gerência do sistema.

O desenvolvimento da **API** aqui apresentada implicou o uso de diversas tecnologias, tais como **Spring Boot**, **Spring Security**, **Json Web Token**, **PostgreSQL**, **Fly Migration** e outras bibliotecas de desenvolvimento **Java**. Ao utilizar a linguagem **Java** em conjunto com essas tecnologias, foi possível realizar a implementação completa da aplicação. Assim, o processo de desenvolvimento proporcionou, em todas as fases, o engajamento na construção do software, desde o planejamento até a conclusão plena e funcional do sistema."

10. REFERÊNCIAS

ASENCIO, L. C., Guarnizo, S. F. G., Caiche, W. C. & Medina, V. V. M. (2017). El diagnóstico organizacional, contextualizado en los negocios fabriles de la provincia de Santa Elena-Ecuador 2015-2016. Innova Research Journal, 137-147. https://doi.org/10.33890/innova.v2.n5.2017.237

CANTOS, M. (2017). Organizational Diagnosis in Publics Schools of Basic Education of Canton Cañar-Ecuador, to Promote its Effectiveness. Población y Desarrollo, 23(44), 86-92. http://dx.doi.org/10.18004/pdfce/2076-054x/2017.023(44)086-092

PORTO, M. A. G. (2006). A importância dos sistemas de informações gerenciais para as organizações. Anais do XIII SIMPEP - Simpósio de Engenharia de Produção, Bauru, SP, Brasil, 6 a 8 de Novembro de 2006.

LEITE, J.C.S.P.; LEONARDI, M.C. Business Rules as organizational policies. In: Proceedings of the 9th International Workshop on Software Specification & Design. ISE-Shima, Japan. 1ed. USA: IEEE CSP, Los Alamitos, CA. p. 68-76, Apr. 1998.

MASCARENHAS, A. O. Gestão Estratégica de Pessoas: Evolução, Teoria e Crítica. São Paulo: Cengage Learning. 2008

ARAÚJO, L.C.G. Organizações, sistemas e métodos e as modernas ferramentas de gestão organizacional. São Paulo: Atlas, 2005

DOS SANTOS, G. L; Prá, R.; Pereira Moraes, J.; Da Silva, M. Diagnóstico Organizacional Como Ferramenta Estratégica na Gestão de Pessoas - Estudo em Empresa Catarinense do Ramo Imobiliário: Organizational Diagnosis As a Strategic Tool In Management of People – Study in Catarinian Company of Branch Real State. Revista Visão: Gestão Organizacional, Caçador (SC), Brasil, v. 9, n. 2, p. 155-167, 2020. DOI: 10.33362/visao.v9i2.1731. Disponível em: https://periodicos.uniarp.edu.br/index.php/visao/article/view/1731.

MELO, W. **Qualidade + testes de Software = Qualidade de Software**. Tiespecialistas, 2015. Disponível em: https://www.tiespecialistas.com.br/qualidade-testes-de-softwares-qualidade-de-software/. Acesso em: 20 jun. 2023.

CHIUSOLI, Claudio. A Teoria e Prática como uma Proposta de um Diagnóstico Empresarial. Revista de Ciências Jurídicas e Empresariais, v. 18, n. 1, p. 40-50, nov. 2017. DOI: 10.17921/2448-2129.2017v18n1p40-50. Disponivel em: https://www.researchgate.net/publication/339805003_A_Teoria_e_Pratica_como_uma_Proposta_de_um_Diagnostico_Empresarial

TOSETTO, M. Gestão Sociotécnica do Teste de Software em Projetos de Sistemas de Informação. Revista de Gestão da Tecnologia e Sistemas de Informação, Journal of Information Systems and Technology Management, Vol. 5, No. 2, 2008, p. 325-346. ISSN online: 1807-1775. DOI: 10.4301/S1807-17752008000200007. Disponivel em: https://www.scielo.br/j/jistm/a/xMpw9Y8PWgkSg6C7Zdd9Lkh/?lang=pt&format=pdf