



**FACULDADE DE TECNOLOGIA DE TAUBATÉ**

**GUILHERME COSTA DE AGUIAR**

**SISTEMA INTEGRADOR DE DADOS BÁSICOS DO  
MUNÍCIPE**

**TAUBATÉ**

**2023**



**FACULDADE DE TECNOLOGIA DE TAUBATÉ**

**GUILHERME COSTA DE AGUIAR**

**SISTEMA INTEGRADOR DE DADOS BÁSICOS DO  
MUNÍCIPE**

Trabalho de Graduação apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Centro Estadual de Educação Tecnológica Paula Souza para a obtenção do diploma de Tecnólogo em Análise e Desenvolvimento de Sistemas.

**Orientador: Prof. Esp. Luís Felipe Féres Santos**

**Coorientador: Prof. Me. Luiz Eduardo Souza Evangelista**

**TAUBATÉ**

**2023**

**GUILHERME COSTA DE AGUIAR**

## **SISTEMA INTEGRADOR DE DADOS BÁSICOS DO MUNÍCIPE**

Trabalho de Graduação apresentado a Faculdade de  
Tecnologia de Taubaté, como parte das exigências  
para a obtenção do diploma de Tecnólogo em Análise  
e Desenvolvimento de Sistemas.

**Orientador: Prof. Esp. Luís Felipe Féres Santos**

Taubaté, 23 de junho de 2023.

### **BANCA EXAMINADORA**

---

Prof<sup>a</sup>. D.ra **Divani Barbosa Gavinier**  
Faculdade de Tecnologia de Taubaté

---

Prof<sup>a</sup>. Esp. **Juliana Oliveira de Souza**  
Faculdade de Tecnologia de Taubaté

---

Prof. Esp. **Luís Felipe Féres Santos**  
Faculdade de Tecnologia de Taubaté

---

Prof. M.e **Luiz Eduardo Souza Evangelista**  
Faculdade de Tecnologia de Taubaté

Dedico este trabalho a minha família e amigos, em especial, minha mãe Maria e meu pai Celso, que sempre me incentivaram e proporcionaram condições para meus estudos e desenvolvimento.

## **AGRADECIMENTOS**

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Aos colegas de classe, com quem nesses anos de estudo tive a felicidade de conviver.

A todos os funcionários da Faculdade de Tecnologia de Taubaté, pela atenção durante toda duração do nosso curso, sempre demonstrando disponibilidade em ajudar.

Aos professores que contribuíram dividindo conhecimento e experiências, em especial ao professor Luís Felipe Féres Santos e ao professor Luiz Eduardo Souza Evangelista que apoiaram o desenvolvimento deste trabalho.

A todos de minha família que durante o curso me deram todo suporte necessário e incentivo para que a essa jornada pudesse ser concluída.

“Quem acende uma luz é o primeiro a se beneficiar  
da claridade.”  
(G. K. Chesterton)

## RESUMO

A administração pública tem se modernizado e cada vez mais se utiliza de sistemas informatizados para realizar seu propósito, criando assim diversos sistemas. Essa grande quantidade de sistemas gera uma variedade de base de dados desordenada e não normalizada. Diversas bases de dados sem integração geram retrabalho e informações imprecisas, prejudicando o planejamento e o estabelecimento de políticas públicas mais assertivas. O propósito deste projeto é viabilizar uma forma de centralizar os dados básicos dos munícipes atendidos pelos serviços da administração pública municipal e criar uma interface de integração com os diversos sistemas já existentes no ente. Para realização desse projeto foi feita pesquisa de plataformas tecnológicas modernas que pudessem viabilizar o desenvolvimento dos sistemas propostos. Além disso, foi realizado levantamento de requisitos e análise para a escolha das tecnologias utilizadas no projeto. A criação do sistema foi baseada em Spring, a utilização de uma API RESTFul para integração e implementação de regras de segurança baseadas na biblioteca Spring Security. Aqui também é proposto a criação de uma aplicação web para a administração da aplicação integradora, utilizando a própria interface de integração criada. Por fim, são apresentados os resultados obtidos utilizando uma base de dados para testes, a integração da aplicação de administração e ainda a integração com ferramentas externas.

**Palavras-Chave:** Integração, API, banco de dados, administração pública.

## **ABSTRACT**

Public administration has been modernized and increasingly uses computerized systems to accomplish its purpose, thus creating different systems. The large number of systems produces a variety of disordered and non-standardized databases. Several databases without integration generate rework and inaccurate information, hindering the planning and establishment of more assertive public policies. The purpose of this project is to enable a way to centralize the basic data of citizens served by municipal public administration services and to create an integration interface with the various systems that already exist in the entity. To carry out this project, research was carried out on modern technological platforms that could enable the development of the proposed systems. In addition, a research of requirements and analysis was carried out for the choice of technologies used in the project. The creation of the system was based on Spring, the use of a RESTful API for integration and implementation of security rules based on the Spring Security library. Here it is also proposed the creation of a web application for the administration of the integrator application, using the integration interface created. Finally, the results obtained using a database for testing, the integration of the administration application and also the integration with external tools are presented.

**Keyword:** Integration, API, database, public administration



## LISTA DE ILUSTRAÇÕES

Figura 1 - Gráfico de quantidade de municípios por número de sistemas integrados	16
Figura 2 - Gráfico de quantidade de municípios por principais sistemas integrados	17
Figura 3 - Gráfico de municípios com sistemas integrados agrupados por quantidade	18
Figura 4 - Visão geral do Spring Framework	22
Figura 5 - Arquitetura do sistema	31
Figura 6 - Diagrama de caso de uso	33
Figura 7 - Diagrama de atividade: Inclusão de novo município	34
Figura 8 - Modelo conceitual	35
Figura 9 - Modelo físico	36
Figura 10 - Modelo físico - Aplicação de administração	36
Figura 11 - Estrutura do projeto da aplicação integradora	37
Figura 12 - Organização dos pacotes da aplicação de administração	38
Figura 13 - IDE Spring Tools durante o desenvolvimento	39
Figura 14 - Postman	40
Figura 15 - Planilha de controle dos endpoints	41
Figura 16 - Tela de documentação da API - Swagger	45
Figura 17 - Resposta da requisição GET	46
Figura 18 - Resposta da requisição POST após validação	47
Figura 19 - Resposta da requisição POST após validação	48
Figura 20 - Resposta da requisição GET	49
Figura 21 - Partes e conteúdo do JWT	50
Figura 22 - Tela de login	51
Figura 23 - Tela inicial	52
Figura 24 - Tela com a lista de munícipes	53
Figura 25 - Tela de cadastro de novo sistema externo	54
Figura 26 - Tela de alteração de município	55
Figura 27 - Tela do dashboard 1	56
Figura 28 - Tela do dashboard 2	56

## LISTA DE TABELAS

Tabela 1: Métodos HTTP no padrão RESTful	27
Tabela 2 - Requisitos funcionais	29
Tabela 3 - Requisitos não funcionais	30

## LISTA DE ABREVIATURAS E SIGLAS

<b>API</b>	Application Programming Interface
<b>BI</b>	Business Intelligence
<b>CBC</b>	Cadastro Base do Cidadão
<b>EE</b>	Enterprise Edition
<b>HTTP</b>	HyperText Transfer Protocol
<b>IDE</b>	Integrated Development Environment
<b>JSON</b>	JavaScript Object Notation
<b>JWT</b>	Json Web Token
<b>REST</b>	Representational State Transfer
<b>SGDB</b>	Sistema Gerenciador de Banco de Dados
<b>SQL</b>	Standard Query Language
<b>TI</b>	Tecnologia da Informação
<b>URI</b>	Uniform Resource Identifier
<b>XML</b>	eXtensible Markup Language

## SUMÁRIO

1	INTRODUÇÃO.....	13
1.1	OBJETIVOS .....	14
1.2	CONTEXTUALIZAÇÃO .....	15
1.3	SOLUÇÕES EXISTENTE NO MERCADO .....	19
2	FUNDAMENTAÇÃO TEÓRICA .....	21
2.1	QUALIDADE DE DADOS .....	21
2.2	JAVA.....	21
2.3	SPRING FRAMEWORK .....	22
2.3.1	SPRING BOOT .....	23
2.3.2	SPRING DATA .....	23
2.3.3	SPRING SECURITY .....	23
2.4	IDE ECLIPSE .....	24
2.5	MARIADB .....	24
2.6	POSTMAN.....	24
2.7	ADMINLTE .....	25
2.8	MICROSOFT POWER BI .....	25
2.9	WEB SERVICES .....	25
2.10	APPLICATION PROGRAMMING INTERFACE .....	26
2.11	SWAGGER.....	26
2.12	SWDOC.....	26
2.13	ARQUITETURA RESTFUL.....	27
3	DESENVOLVIMENTO.....	29
3.1	LEVANTAMENTO DE REQUISITOS .....	29
3.2	ARQUITETURA DO SISTEMA.....	31
3.3	DIAGRAMA DE CASO DE USO .....	32
3.4	DIAGRAMA DE ATIVIDADE.....	33
3.5	MODELAGEM DO BANCO DE DADOS.....	34
3.5.1	MODELO CONCEITUAL .....	34
3.5.2	MODELO FÍSICO .....	35
3.6	ORGANIZAÇÃO DO AMBIENTE DE DESENVOLVIMENTO.....	36
3.6.1	IDE <i>SPRING TOOLS</i> E ORGANIZAÇÃO DOS PACOTES .....	36
3.6.2	ORGANIZAÇÃO DO AMBIENTE DE TESTE .....	39

3.7	CONFIGURAÇÃO DA DOCUMENTAÇÃO COM SWAGGER.....	42
3.8	CONFIGURAÇÃO DA CAMADA DE SEGURANÇA.....	42
3.9	INTERFACE DO USUÁRIO.....	43
4	RESULTADOS OBTIDOS .....	44
4.1	APLICAÇÃO INTEGRADORA.....	44
4.1.1	SWAGGER.....	44
4.1.2	REPOSTAS DAS REQUISIÇÕES DA API .....	45
4.1.3	APLICAÇÃO SEGURA UTILIZANDO JWT .....	48
4.2	APLICAÇÃO DE ADMINISTRAÇÃO .....	50
5	CONCLUSÃO .....	57
	REFERÊNCIAS .....	58
	APÊNDICE A – VÍDEO DE DEMONSTRAÇÃO DOS RESULTADOS OBTIDOS .....	63

## 1 INTRODUÇÃO

Nesse momento de consolidação do uso da tecnologia, quando todos os setores da sociedade procuram soluções tecnológicas para atender suas demandas, o setor público também se moderniza através de sistemas computacionais, almejando mais eficiência em seus propósitos.

De acordo com Khoubati, Kalhoro e Bukhari (2008), as organizações do setor público possuem um conjunto de sistemas de informação complexos e com formato de dados diversificado, plataformas computacionais heterogêneas e bancos de dados não conectados.

Segundo o Tribunal de Contas de São Paulo (2022), em pesquisa de autoria própria, apenas 3% dos municípios pesquisados possuem integração com todos os módulos elencados na pesquisa.

Diante deste cenário diverso sem a devida interoperabilidade, observa-se: A impossibilidade de extrair dados confiáveis em razão de registros múltiplos onde deveriam ser únicos; O retrabalho em setores que realizam cadastro de dados básicos dos munícipes; A dificuldade na migração de dados durante a substituição dos fornecedores de sistemas informatizados. Este trabalho busca formas de permitir a normalização e centralização dos dados básicos dos munícipes nas soluções informatizadas em operação na administração municipal.

Sabendo que a responsabilidade final dos dados armazenados nestes sistemas ainda é do órgão público, e que é de interesse dele a visão geral do cenário, independentemente de quem é o prestador do serviço de informatização, uma abordagem centralizada e reguladora pode sanar as inconsistências e prover a interoperabilidade entre sistemas no aspecto que tange os dados básicos dos munícipes.

Khoubati, Kalhoro e Bukhari (2008) destacam a importância de integrar os sistemas de informação e aplicações existentes nas organizações do setor público, a fim de estabelecer uma estrutura eficiente de tecnologia da informação (TI).

Com o objetivo de centralizar e regular o acesso aos dados básicos dos munícipes, é proposto o desenvolvimento em linguagem *Java* com o *framework Spring MVC* uma estrutura centralizada, dotada de um banco de dados relacional utilizando o Sistema Gerenciador de Banco de Dados (SGBD) MariaDB, com uma plataforma *web* de administração das regras e autorizações de acesso, além de *Application*

*Programming Interface* (APIs) no padrão *Representational State Transfer* (REST) disponibilizadas aos sistemas terceirizados, para que estes possam enviar e receber dados de forma padronizada.

A unidade dos dados básicos dos cidadãos reflete em um melhor serviço prestado ao usuário final, evitando retrabalho ao cadastrar e atualizar os mesmos dados em vários sistemas. Também provê dados mais confiáveis para as análises e planejamento de políticas públicas.

Como prover a consistência dos dados básicos dos cidadãos no âmbito da administração municipal através de integração de sistemas terceirizados de forma padronizada?

Para atingir seus objetivos, este trabalho apresenta artigos acerca da consistência de dados e então embasa a escolha das tecnologias para o desenvolvimento da aplicação proposta. Cria uma lista de requisitos funcionais e requisitos não funcionais em acordo com as necessidades da administração pública municipal e aos objetivos do trabalho. Elabora os diagramas pertinentes e descreve o desenvolvimento dos sistemas propostos para a solução.

Nesta introdução, uma visão geral sobre o problema a ser resolvido é apresentada, bem como a importância desse trabalho de graduação dentro desse assunto. Além disso, são explanados os objetivos, a contextualização do tema e apresenta solução semelhante no mercado. O segundo capítulo, por sua vez, embasa mais profundamente o tema de pesquisa através de bibliografias e trata, ainda, o lado técnico do trabalho em relação ao desenvolvimento da solução proposta. Em seguida, no terceiro capítulo, aspectos como os requisitos, diagramas, a estrutura da aplicação, a interface de usuário, o ambiente de desenvolvimento e testes compõem tópicos essenciais do funcionamento do sistema apresentado neste trabalho de graduação. No capítulo subsequente, são apresentados os resultados obtidos após o desenvolvimento do sistema proposto. No capítulo seguinte, a conclusão do que fora alcançada com este projeto é disposta e, por fim, as fontes de pesquisas utilizadas.

## **1.1 OBJETIVOS**

O objetivo geral deste trabalho é desenvolver um sistema que permita padronizar básicos de usuários dos sistemas informatizados da administração pública municipal. Utilizando um modelo desacoplado, pretende-se desenvolver duas aplicações: A aplicação principal e a aplicação de administração. A aplicação principal

possui o objetivo de centralizar, padronizar dados e fornecer meio de comunicação com sistemas terceiros através de APIs. A aplicação de administração tem como objetivo gerenciar os acessos à aplicação principal e recuperar dados estatísticos de sua utilização através de *dashboards*.

A proposta apresentada neste trabalho possui a finalidade de solucionar os problemas advindo de múltiplas bases de dados, causados pela diversidade de sistemas comumente contratados pelas administrações públicas municipais.

Como objetivos específicos, este trabalho traz a pesquisa de ferramentas tecnológicas para utilização no desenvolvimento das aplicações, o levantamento de requisitos que nortearão a produção do sistema, a elaboração dos diagramas necessários para apoiar o desenvolvimento. Além da criação do protótipo através de uma estrutura centralizada em linguagem Java com o framework Spring Boot, dotada de um banco de dados relacional utilizando o SGBD MariaDB. e uma plataforma de administração das regras e autorizações de acesso. Inclui disponibilizar APIs no padrão REST para sistemas terceirizados E por fim, apresentar as principais considerações.

## 1.2 CONTEXTUALIZAÇÃO

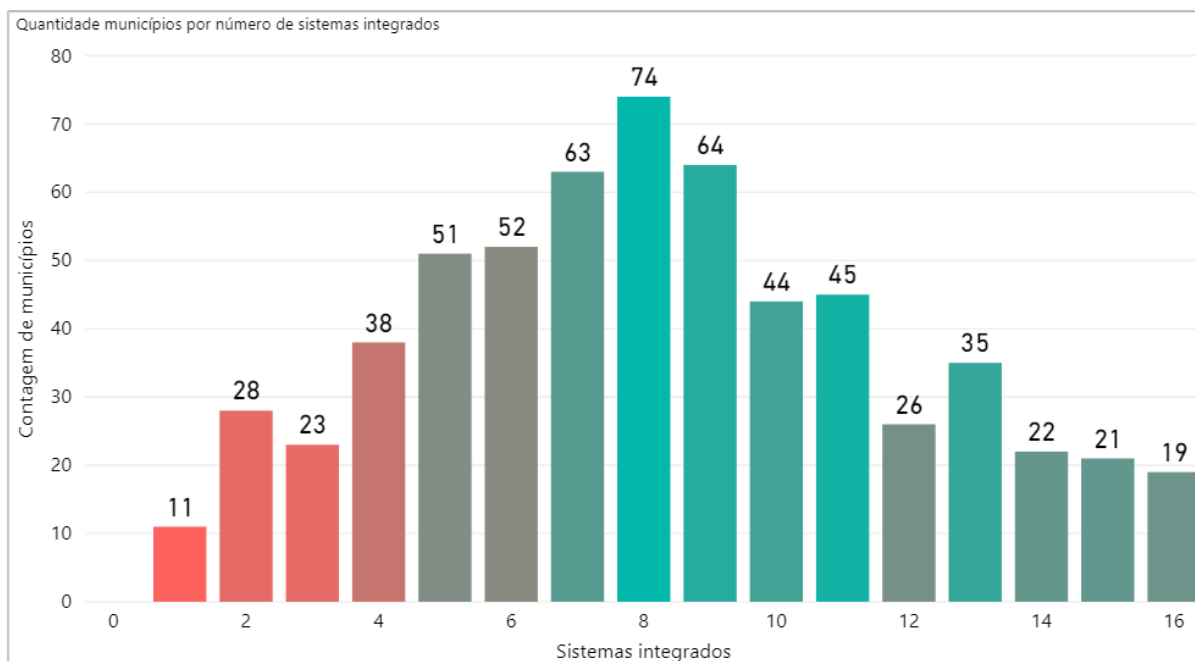
Com o rápido crescimento populacional nas cidades, a demanda por serviços públicos sofre um crescimento equivalente, exigindo do poder público soluções mais eficientes. Aliado a isso, o surgimento de novas tecnologias proporciona um ambiente favorável à inovação, e conseqüentemente uma maior oferta de soluções tecnológicas para atender as demandas da sociedade, inclusive do poder público. A busca cada vez maior por essas inovações, quando feita com pouco planejamento, pode criar um ambiente de dados heterogêneo dentro de uma organização.

A Abep-Tic (2022), em pesquisa realizada anualmente, aponta que somente entre 2021 e 2022 houve um aumento de 14% na pontuação dos estados no índice que mede a oferta de serviços públicos digitais.

De acordo com dados pesquisa do Índice de Efetividade da Gestão Municipal (IEG-M) realizado pelo Tribunal de Contas do Estado de São Paulo (2022), apenas 19 municípios do 616 pesquisados possuíam todos os sistemas integrados, conforme demonstra o gráfico da figura 1 a seguir.



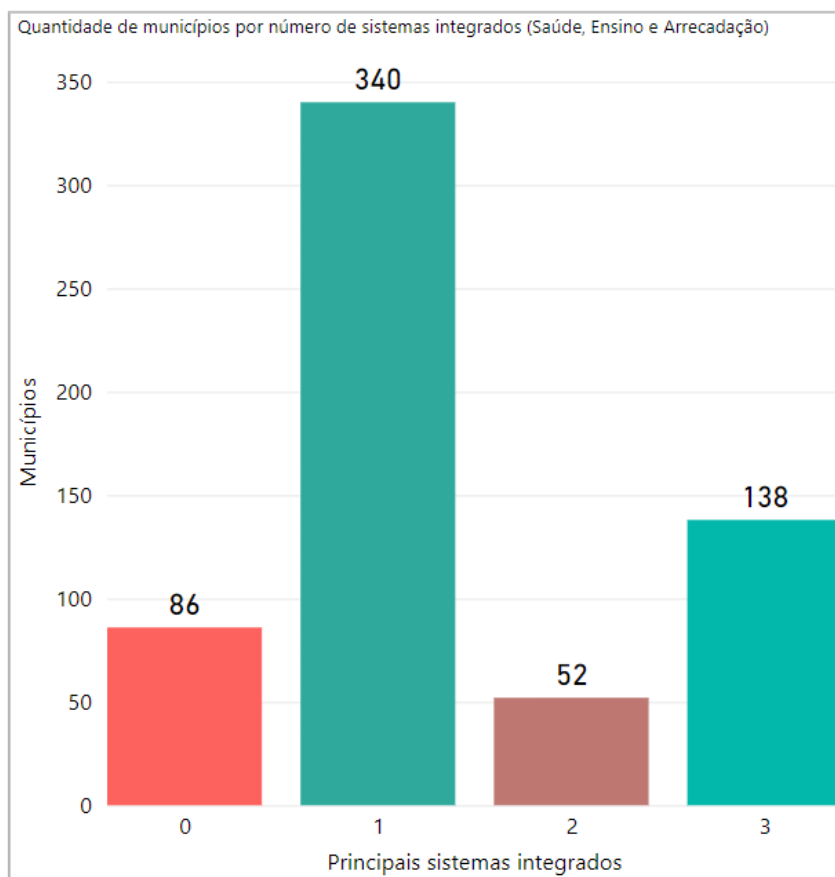
Figura 1 - Gráfico de quantidade de municípios por número de sistemas integrados



Fonte: De autoria própria (2023). Elaborado com dados de Tribunal de Contas do Estado de São Paulo (2022).

Quando se observa apenas os sistemas de Saúde, Ensino e Arrecadação, que possuem grande fluxo de dados, observa-se que a maioria dos municípios pesquisados dispões de apenas dois sistemas com alguma integração, como demonstra o gráfico do da figura 2 adiante.

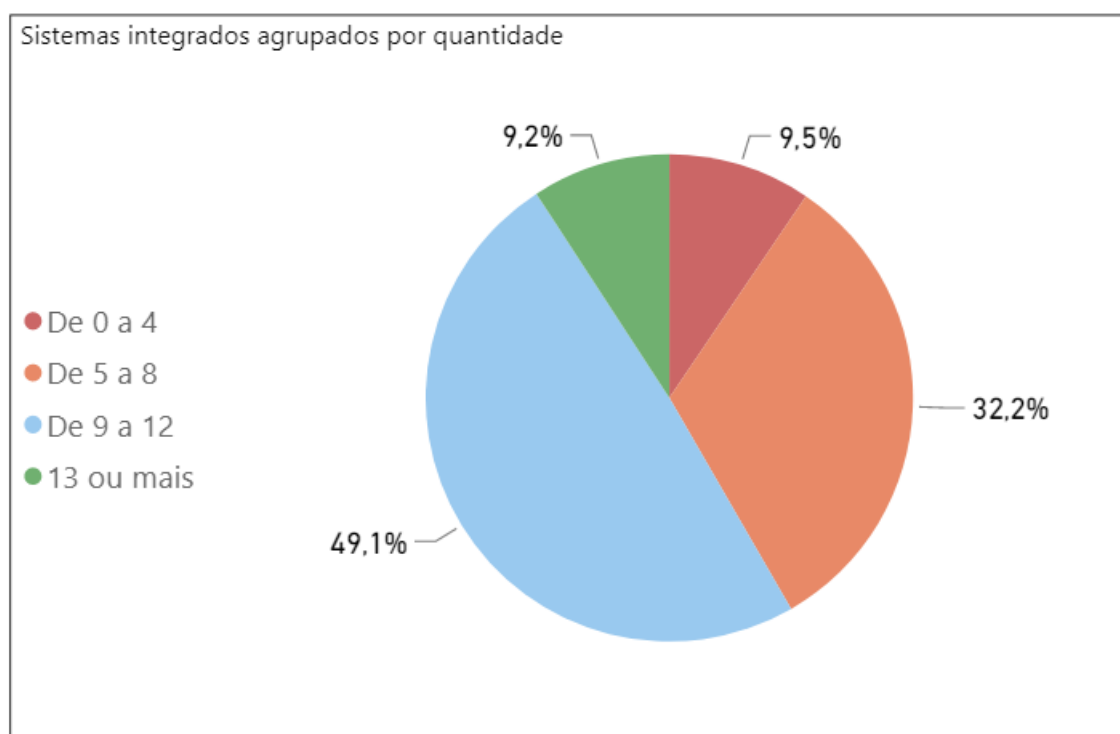
Figura 2 - Gráfico de quantidade de municípios por principais sistemas integrados



Fonte: De autoria própria (2023). Elaborado com dados de Tribunal de Contas do Estado de São Paulo (2022).

Além disso, conforme mostra o gráfico da figura 3, através de uma ótica segmentada por grupos quantitativos, nota-se que 9,5% dos municípios possuíam em 2021 apenas quatro ou menos sistemas integrados, que 32,2% apresentavam entre cinco e oito sistemas, que a maior parcela representando 49,1% dos municípios alcançou a marca de nove a doze sistemas e por fim, apenas 9,2% possuíam grande parte dos sistemas integrados, sendo treze ou mais.

Figura 3 - Gráfico de municípios com sistemas integrados agrupados por quantidade



Fonte: De autoria própria (2023). Elaborado com dados de Tribunal de Contas do Estado de São Paulo (2022).

Este cenário apresenta uma série de problemas relacionados à gestão dos dados básicos dos municípios, tais como a impossibilidade de extrair informações confiáveis em razão de registros múltiplos, o retrabalho na inserção desses dados em setores de atendimento ao público e a dificuldade na migração de dados durante a substituição dos fornecedores de sistemas informatizados.

Ainda, os dados presentes em diversas bases podem gerar dificuldade quando se fala do correto tratamento de dados, matéria da Lei Geral de Proteção de Dados. Esta legislação, através do artigo 60 que altera o artigo 7º da Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet), garante o direito ao titular dos dados solicitar a exclusão definitiva de dados pessoais nos termos da lei. Devido as múltiplas bases de dados, assegurar que a solicitação exclusão de dados do titular foi realmente executada passar a ser dificultoso.

Nesse contexto, é importante ressaltar que a responsabilidade dos dados armazenados é tanto do órgão público quanto de seus fornecedores.

De acordo com Luca e Bassi (2023), para assegurar uma gestão sustentável e eficiente, que utilize dados para solucionar problemas urbanos, as cidades precisam

integrar e analisar dados gerados e capturados de diversas fontes.

Uma abordagem centralizada e reguladora se faz necessária para sanar as inconsistências, prover a interoperabilidade entre sistemas no aspecto que tange os dados básicos dos municípios e ainda apoiar o processo de tratamento de dados em consonância com as legislações vigentes.

### 1.3 SOLUÇÕES EXISTENTE NO MERCADO

Em pesquisa realizada pelo autor em setembro de 2022, foi identificada uma solução disponível no mercado que se assemelha com a proposta deste trabalho. O Cadastro Base do Cidadão (CBC), de acordo com Gov.br (2023a), tem como objetivo principal unificar e aprimorar os dados referentes aos cidadãos no âmbito governamental. Para isso, utiliza como base os cadastros já existentes nas diferentes bases de dados governamentais, especialmente o Cadastro de Pessoa Física mantido pela Secretaria de Receita Federal. A proposta do CBC não é substituir ou eliminar os cadastros existentes, mas sim harmonizar as informações neles contidas.

Ainda segundo Gov.br (2023b), o acesso ao CBC é restrito aos órgãos que fazem parte do Sistema de Administração dos Recursos de Tecnologia da Informação (SISP), subordinado ao Ministério da Gestão e Inovação dos Serviços Públicos. Atualmente possui 247 órgãos cadastrados.

Gov.br (2023b) lista as diretrizes para acesso ao CBC:

1. Para o cidadão

APIs são a base para a troca automática e segura de informações entre os sistemas e **não estão disponíveis** para acesso pelo cidadão. [...]

2. Para órgãos não SISP

Para utilizar a API, é necessário que o órgão solicitante faça parte do Sistema de Administração dos Recursos de Tecnologia da Informação (SISP) [...]

3. Para a iniciativa privada

O acesso à API pelo Programa Conecta gov.br está disponível **apenas** para órgãos do SISP.

4. Para órgãos do SISP

Caso o órgão solicitante faça parte do SISP e deseje utilizar a API, é preciso realizar a adesão ao Programa Conecta gov.br e obter a autorização do órgão cedente dos dados **Receita Federal - RFB**, de acordo com os seguintes passos: [...]

A pesquisa realizada constata que a barreira de entrada para utilização do CBC é demasiadamente alta. Além disso, as funcionalidades disponibilizadas aos órgãos

aderentes são restritas no que diz respeito à inserção e atualização dos dados, permitindo apenas a forma de consulta.

## 2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo serão expostos os fundamentos que permeiam o desenvolvimento do sistema, com a apresentação de textos sobre a importância da qualidade de dados, passando pelas tecnologias e ferramentas propostas para a construção da solução.

### 2.1 QUALIDADE DE DADOS

De acordo com Rêgo (2013), dados são a matéria-prima necessária para almejar a utilização das informações afim de tomar decisões ágeis corretas.

A importância da qualidade dos dados de um sistema computacional fica evidente quando Barbieri (2020) aponta que dados sem qualidade comumente causam problemas em tomadas de decisões e em aspectos de segurança.

Barbieri (2020) ainda define um conjunto de características de qualidade de dados, considerando algumas dimensões: Integridade dos dados, completude e validade, unicidade, acurácia, consistência e sincronização, temporalidade, facilidade de uso e acessibilidade, segurança e privacidade, entre outros.

### 2.2 JAVA

Segundo Oracle (2022), “Java é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995[...]”.

Mendes (2009, p. 17) conceitua:

A linguagem Java é considerada simples porque permite o desenvolvimento de sistemas em diferentes sistemas operacionais e arquiteturas de hardware, sem que o programador tenha que se preocupar com detalhes de infraestrutura. Dessa forma, o programador consegue desempenhar seu trabalho de uma forma mais produtiva e eficiente.

Ainda de acordo com Mendes (2009), a linguagem Java foi concebida seguindo o paradigma de orientação a objetos, diferenciando da programação estruturada e trazendo formas mais próximas do mundo real para o gerenciamento de um sistema.

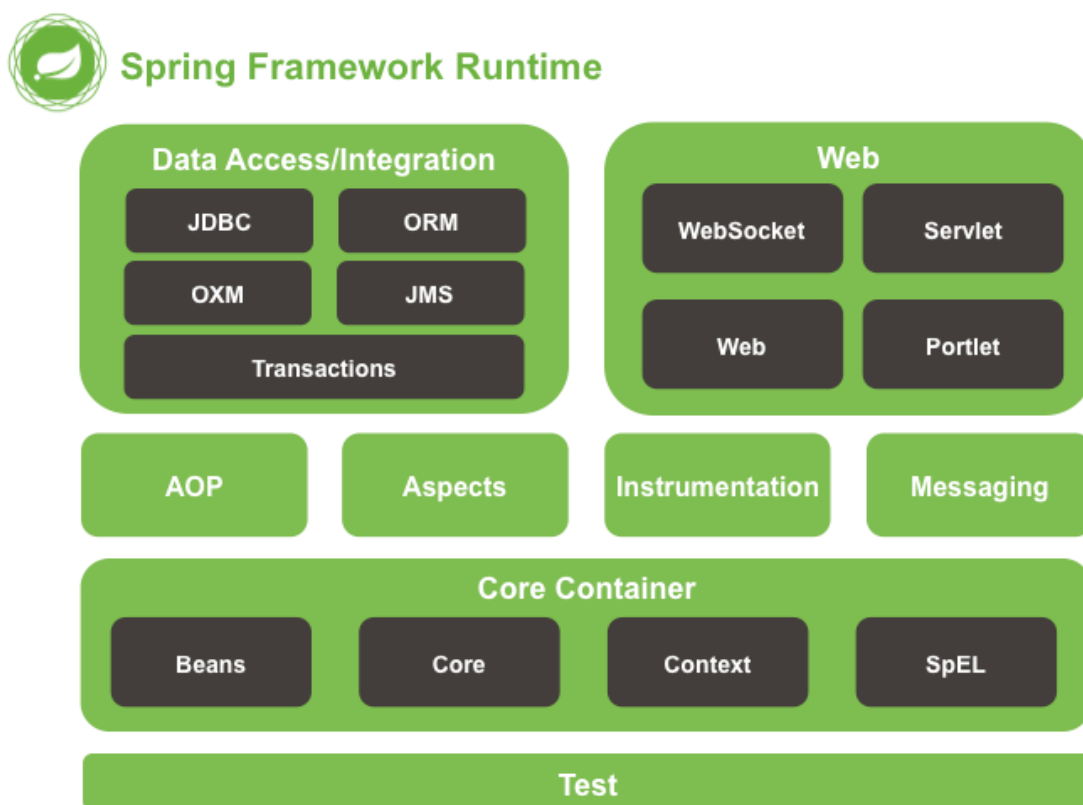
A plataforma Java *Enterprise Edition* (EE) contém um conjunto de tecnologias que reduzem a complexidade e custo de desenvolvimento de aplicações centradas em servidores. Essa plataforma oferece uma coleção de APIs que permitem o desenvolvimento e execução de aplicações robustas e confiáveis no lado do servidor, de acordo com Apache (2022).

## 2.3 SPRING FRAMEWORK

Segundo Santana (2021), o Spring é um *framework* Java que possui uma grande quantidade de projetos relacionados, como o Spring Boot e o Spring Data. É um projeto robusto e estável, com sua primeira versão publicada em 2002.

Spring (2022) em sua documentação oficial explica que, o Spring Framework constitui-se de uma coleção de recursos organizados em aproximadamente 20 módulos. Esses módulos são reunidos em grupos maiores que possuem conceitos similares em seus propósitos. Os grupos de módulos são: o *Core Container*, *Data Access/Integration*, *Web*, *AOP (Aspect Oriented Programming)*, *Instrumentation*, *Messaging* e *Test*, conforme mostrado no diagrama na figura 4 a seguir.

Figura 4 - Visão geral do Spring Framework



Fonte: Spring (2022).

Ainda de acordo com Spring (2022), este *framework* fornece um modelo amplo de programação e configuração para aplicativos desenvolvidos em Java, permitindo o

desenvolvedor se concentrar na lógica de negócios no nível de aplicativo, sem se preocupar com detalhes desnecessários dos ambientes de implantação.

### **2.3.1 SPRING BOOT**

Santana (2021) explica que o Spring Boot é uma maneira simplificada de criar projetos baseados no Spring Framework. A utilização do Spring Boot também facilita a configuração das aplicações através de arquivos específicos ou diretamente na aplicação, descartando a necessidade do uso de arquivos *eXtensible Markup Language* (XML).

Spring (2022) elenca as principais características do Spring Boot: Possibilita a criação de aplicativos Spring autônomos, permite incorporar servidores Tomcat, Jetty ou Undertow, disponibiliza dependências para simplificar sua configuração de compilação, possibilita a configuração automática de bibliotecas Spring e de terceiros e, por fim, fornece recursos prontos para produção, como métricas, verificações de integridade e configuração externa.

### **2.3.2 SPRING DATA**

O Spring Data é um projeto que possui diversos módulos relacionados ao acesso de dados em múltiplos sistemas de armazenamento de dados. Não se trata de uma especificação ou padrão, é uma abstração que tenta trazer consistência ao acesso de dados. (PHALTANKAR, 2022)

Kainulainen (2012) complementa ao afirmar que objetivo do Spring Data é fornecer uma maneira fácil para criar aplicações que usem tanto banco de dados relacionais quanto novas tecnologias de armazenamento, como banco de dados não relacionais ou baseados em nuvem, tudo isso utilizando o framework Spring.

Spring (2022) cita como características do Spring Data, o poderoso repositório de abstrações de mapeamento de objetos, suporte para auditoria transparente (criação, última alteração) e integração avançada com controladores do Spring MVC.

### **2.3.3 SPRING SECURITY**

Spring (2023) define o Spring Security como um framework que provê autenticação, autorização e proteção contra ameaças comuns. É a estrutura padrão para proteger aplicativos baseados no Spring framework.

Segundo Mularien (2010) “O Spring Security 3 fornece uma riqueza de recursos



que permitem muitas práticas de segurança sejam declaradas ou configuradas de maneira direta.”

Mularien (2010, p. 18) ainda justifica a utilização do Spring Security:

O Spring Security existe para preencher uma lacuna no universo das bibliotecas Java de terceiros, muito como o Spring Framework originalmente fez quando foi introduzido pela primeira vez. Padrões como Java Authentication and Authorization Service (JAAS) ou Java EE Security oferecem algumas maneiras de realizar algumas das mesmas funções de autenticação e autorização, mas o Spring Security é um vencedor porque empacota tudo o que você precisa implementar uma solução de segurança de aplicativos de ponta a ponta de forma concisa e maneira sensata.

## 2.4 IDE ECLIPSE

O ambiente integrado de desenvolvimento, ou *Integrated Development Environment* (IDE) Eclipse é uma plataforma criada a partir do Projeto Eclipse, originalmente criado em novembro de 2001 pela IBM. A Eclipse Foundation foi criada em janeiro de 2004 como uma organização independente e sem fins lucrativos para agir como um mantenedor da comunidade Eclipse, de acordo com Eclipse (2022).

Partindo de um desenvolvimento colaborativo, a comunidade desenvolveu Spring Tools, que, de acordo com Eclipse (2022), são ferramentas para a IDE Eclipse para a escrita de grandes aplicativos, aplicativos modernos e micros serviços baseados no framework Spring Boot.

## 2.5 MARIADB

O MariaDB é um Sistema Gerenciador de Banco de Dados (SGDB), necessário para a implementação de um banco de dados relacional e confiável.

Segundo MariaDB (2022, tradução nossa) “O servidor Maria DB é um dos servidores de banco de dados mais populares do mundo. É feito pelos desenvolvedores originais do MySQL e garantido em continuar em código aberto. Usuário notáveis incluem Wikipedia, WordPress. Com e Goolge”.

## 2.6 POSTMAN

De acordo com Postman (2023, tradução nossa), a ferramenta Postman é um software utilizado para testar e documentar APIs (*Application Programming Interfaces*). Ele foi criado em 2012 por Abhinav Asthana, Ankit Sobti e Abhijit Kane, com o objetivo de facilitar o desenvolvimento e a integração de APIs. Com o Postman, é possível criar e enviar requisições *HyperText Transfer Protocol* (HTTP) para uma

API, testar suas funcionalidades, validar as respostas e documentar as informações obtidas. O software permite ainda automatizar testes de rotina e colaborar com outros membros da equipe na documentação e desenvolvimento da API.

## 2.7 ADMINLTE

De acordo com Adminlte.io (2023) o AdminLTE é um modelo HTTP para painel de administração de sistemas. Criado com código aberto e utilizando o Bootstrap, o AdminLTE disponibiliza uma gama de componentes responsáveis e reutilizáveis.

Mendes (2018) lista como benefícios na utilização do AdminLTE a compatibilidade com os principais navegadores de internet, o baixo consumo de recursos computacionais, o baixo tempo de resposta e ainda cita que, por ser um modelo de código aberto, a comunidade envolvida contribui na solução de dúvidas e problemas encontrados por seus utilizadores.

## 2.8 MICROSOFT POWER BI

O Microsoft Power BI é um conjunto integrado de serviços de software, aplicativos e conectores, que colaboram entre si para a conversão de diversas fontes de dados em informações coerentes, visualmente atraentes e interativas. Essas fontes podem variar desde planilhas do Excel até *data warehouses* híbridos, tanto locais quanto baseados em nuvem. (MICROSOFT, 2023).

Segundo Horwitz e Scardina (2022), os modelos de dados desenvolvidos com o auxílio do Power BI apresentam diversas formas de utilização nas organizações. Essas incluem a capacidade de contar histórias por meio de gráficos e visualizações de dados, explorar cenários hipotéticos dentro dos dados por meio de análises "e se" e criar relatórios que possibilitam responder a perguntas em tempo real e auxiliar na previsão para garantir o cumprimento das métricas da organização.

## 2.9 WEB SERVICES

Wittig (2015) define *web services* como serviços que podem ser controlados por uma interface web, podendo ser usada por outros sistemas e ou por pessoas, através de uma interface gráfica de usuário.

O cenário heterogêneo e dependente de integração requer soluções que possibilitem isso. Devmedia (2016) afirma que umas tecnologias que possibilitam essa integração são os *web services*, definindo-os como aplicações cliente-servidor que

operam através do protocolo HTTP com o objetivo de possibilitar serviços entre aplicações, mesmo que estas estejam sendo executadas em plataformas distintas. Esses softwares utilizam a linguagem XML como padrão para fornecer as marcações que podem ser interpretadas por outros sistemas, dessa forma, possibilitando inúmeros programas a interagir uns com os outros para fornecer soluções para o cenário de tecnologias e plataformas diversas.

## 2.10 APPLICATION PROGRAMMING INTERFACE

Uma *Application Programming Interface* (API), permite aos desenvolvedores explorar funções de um sistema de computador em outras aplicações. Com o passar dos anos, as APIs evoluíram apoiadas em tecnologias avançadas, como a velocidade das redes e segurança. (ZHU et al., 2014)

Ainda de acordo com Zhu (2014), companhias como Google, Facebook e Twitter têm utilizado APIs para expor seus serviços e permitir à desenvolvedores explorar suas funcionalidades.

## 2.11 SWAGGER

Swagger é um framework usado para descrever API usando linguagem comum que é familiar a todos. Pode ser comparado como uma planta baixa de uma casa, segundo Prompt Softech (2019).

De acordo com Swagger (2022), Swagger começou como uma simples especificação *open source* para criação de APIs RESTful, em 2010. Em 2015, o projeto foi adquirido pela empresa SmartBear Software, e então a especificação foi doada para a fundação Linux e renomeada para OpenAPI. Desde então, o Swagger se tornou o conjunto de ferramentas mais popular para explorar o potencial da especificação criada inicialmente.

## 2.12 SWDOC

Baeldung (2023) define o SwDoc como uma ferramenta disponível online que gera documentação de uma API em arquivo no formato PDF, a partir da especificação fornecida em um arquivo Swagger.json, usando o conversor Swagger2Markup para gerar arquivos compatíveis com o AsciiDoctor, e em seguida utiliza este último para analisar esses arquivos em um modelo de documento e os converter em arquivo PDF.

A solução disponibilizada pelo SwDoc possui fácil utilização e é uma boa

escolha para aqueles que já possuem uma documentação implementada pelo Swagger. (BAELDUNG, 2023).

## 2.13 ARQUITETURA RESTFUL

A *Representational State Transfer* (REST) é um popular estilo de arquitetura de software, comumente utilizado para a criação de *web services* e na integração de sistemas, segundo Lecheta (2015).

Devmedia (2016) complementa afirmando que o estilo REST determina como deve ser efetuada a Transferência de Estado Representacional, ou a *Representational State Transfer*, em inglês. Em outros termos, significa a representação correspondente a um conjunto de valores caracteriza uma determinada entidade em um determinado instante.

O REST utiliza o protocolo HTTP em seus serviços que respondem às requisições com dados geralmente nos formatos JSON ou XML, servindo de alternativa ao SOAP, de acordo com Ferreira (2015).

O modelo de arquitetura REST possui alguns princípios básicos, elencados por Ferreira (2015): Identificação de recursos, utilização de *Uniform Resource Identifiers* (URIs) legíveis, padronização na identificação dos recursos, não utilização do formato desejado da representação do recurso na URI, utilização dos métodos HTTP para manipulação dos recursos (GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS), suporte à diferentes representações e comunicação sem estado, ou *stateless*.

A seguir, na tabela 1 Agrawal (2018) exemplifica o uso padronizado dos métodos HTTP:

Tabela 1 - Métodos HTTP no padrão RESTful

Nome da rota	URL	Verbo HTTP	Descrição
Index	/blogs	GET	Exibe a lista de todos os blogs
New	/blog/new	GET	Mostra um formulário para criar novos blogs
Create	/blogs	POST	Adiciona um novo blog no banco de dados, então redireciona
Show	/blogs/:id	GET	Mostra as informações sobre um blog
Edit	/blogs/:id/edit	GET	Mostra um formulário para edição de um blog existente

Update	/blogs/:id	PUT	Atualiza um blog em particular, então redireciona
Destroy	/blogs/:id	DELETE	Apaga um blog em particular, então redireciona

Fonte: Traduzido e adaptado de Agrawal (2018).

Lecheta (2015) conclui: “Então, se um *web service* seguir estes princípios básicos, podemos dizer que, em termos gerais, este é um *web service* RESTful”.

### 3 DESENVOLVIMENTO

Neste capítulo serão apresentadas as atividades realizadas e os artefatos produzidos durante o desenvolvimento do sistema.

Neste ponto, cabe salientar que a solução proposta é composta de duas aplicações: uma denominada aplicação integradora, para disponibilizar os recursos e outra denominada aplicação de administração para consumir esses recursos. Aqui será apresentado conteúdo referente às duas aplicações, e quando se fizer necessário, será apontado a qual aplicação o conteúdo se refere.

#### 3.1 LEVANTAMENTO DE REQUISITOS

De acordo com Nilcain (2018), o levantamento de requisitos é a etapa inicial do ciclo de desenvolvimento de um software, onde são identificadas as funcionalidades de escopo do projeto.

Durante o desenvolvimento desse trabalho, foram identificados os requisitos funcionais demonstrados a seguir na tabela 2.

Tabela 2 - Requisitos funcionais

Requisitos funcionais		
Código	Descrição do requisito funcional	Detalhamento
RF001	Possuir aplicação web para administração dos aplicativos cadastrados	Inclusão, alteração e inativação dos aplicativos
RF002	Possuir aplicação web para acesso aos dados dos munícipes	Inclusão e alteração dos dados dos munícipes
RF003	Possuir aplicação web para administração dos usuários do sistema cadastrados	Inclusão, alteração e inativação dos usuários
RF004	Possuir dashboard para visualização das principais métricas	-
RF005	Possuir perfis de permissões aos usuários do sistema	-
RF006	Permitir a vinculação de usuários da aplicação integradora aos perfis de permissões	-
RF007	Possibilitar a inclusão de novos dados através de API na arquitetura REST	-
RF008	Deverá registrar o sistema externo que realizou alterações ou inclusões de dados	-
RF009	Deverá registrar o usuário que realizou alterações ou inclusões de dados	-

RF010	Permitir a recuperação dos dados de forma individual	Informar ID ou CPF
RF011	Permitir a recuperação dos dados em lote	Informar uma série de ID
RF012	Gerar ID único automaticamente para os munícipes cadastrados	-
RF013	Deverá validar o formato dos dados de entrada	Em conformidade com o tipo armazenado em banco de dados
RF014	Deverá verificar a duplicidade de munícipes no momento da inclusão	Utilizar o CPF como chave
RF015	Permitir somente a inclusão de CPF validado	-
RF016	Deverá validar dados de entrada de acordo com as tabelas base do sistema	CEP
RF017	Deverá armazenar as tentativas de inclusão ou alteração de dados não sucedidas	Tentativas não sucedidas em razão de dados de entrada incorretos ou duplicados
RF018	Possuir função que possibilite a exclusão de dados, em conformidade com a LGPD	-
RF019	Possuir função que possibilite o compartilhamento de dados de mediante autorização do munícipe	-
RF020	Possuir função que receba dos sistemas terceiros a solicitação de exclusão de dados, conforme a LGPD	-
RF021	Possuir função que notifique os sistemas terceiros da solicitação de exclusão de dados, conforme a LGPD	-
RF022	Possuir função de verificação de dados entrada	A função deve retornar se a inclusão foi nem sucedida
RF023	Não deverá permitir a exclusão de dados	Exceto por solicitação explícita do usuário ou por ordem judicial

Fonte: De autoria própria (2022).

A tabela 3, a seguir, mostra os requisitos não funcionais identificados durante a fase de levantamento de requisitos.

Tabela 3: Requisitos não funcionais

Requisitos não funcionais		
Código	Descrição do requisito não funcional	Detalhamento
RNF001	Possuir banco de dados relacional	-

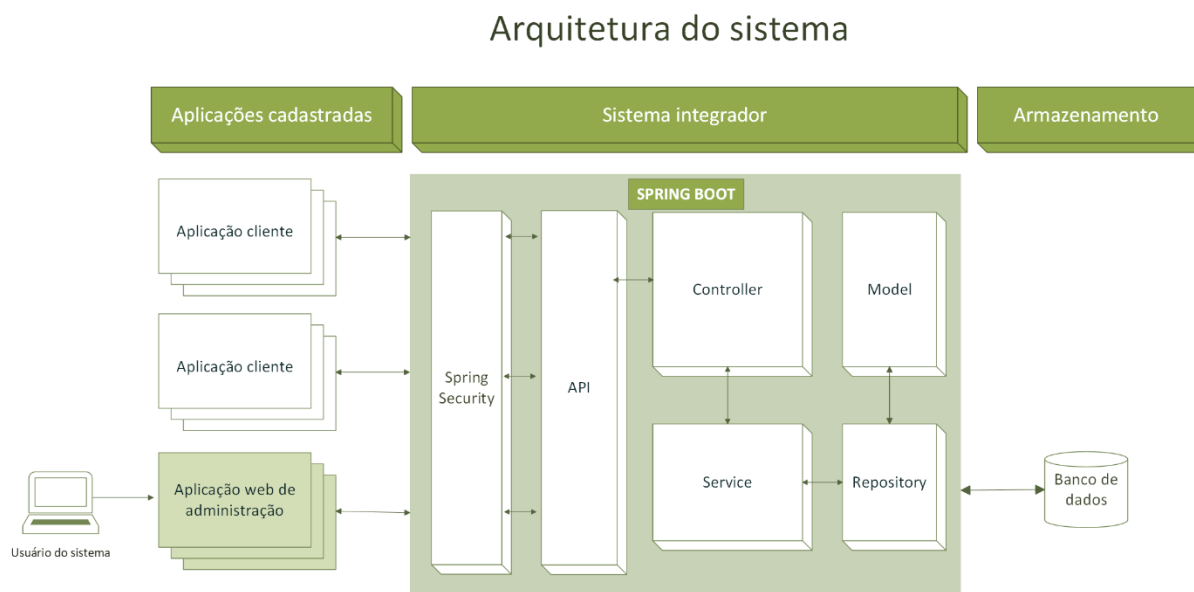
RNF002	A aplicação de administração deverá ser um sistema web	-
RNF003	Utilizar o padrão RESTful nas APIs disponibilizadas	-
RNF004	Possuir documentação para a utilização das APIs por parte dos aplicativos cadastrados	-
RNF005	Deverá atender a legislação vigente	-
RNF006	Possuir acesso somente aos aplicativos cadastrados	-

Fonte: De autoria própria (2022).

### 3.2 ARQUITETURA DO SISTEMA

A arquitetura da solução proposta neste trabalho foi definida de maneira a ser facilmente desacoplada e independente entre seus próprios componentes. A figura 5 mostra um esboço dos componentes do sistema e a forma de comunicação entre si.

Figura 5 - Arquitetura do sistema



Fonte: De autoria própria (2023).



### 3.3 DIAGRAMA DE CASO DE USO

Segundo Pressman (2011), um diagrama UML de caso de uso é uma visão de todos os casos de usos e como eles se relacionam, desta forma, fornecem uma visão macro do sistema.

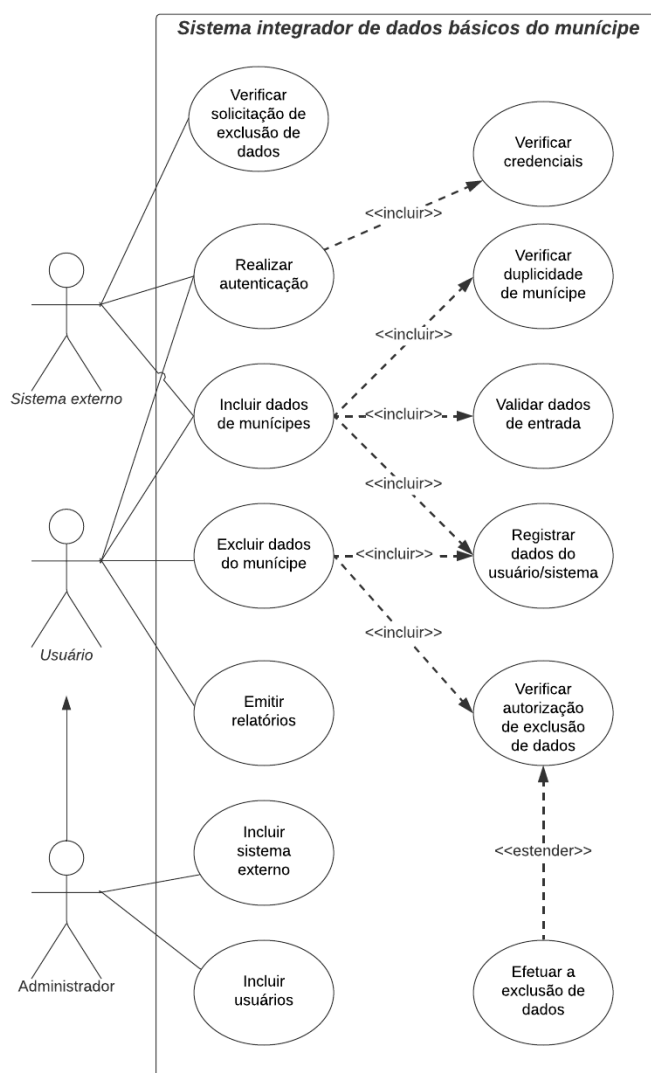
Pressman (2011, p. 731) também explica o caso de uso em si:

Um caso de uso descreve como um usuário interage com o sistema definindo os passos necessários para atingir um objetivo específico (por exemplo, gravar uma lista de músicas em um CD). Variações na sequência de passos descrevem vários cenários (por exemplo, o que acontece se as músicas da lista não couberem em um CD?).

A figura 6 apresenta o diagrama de caso de uso do sistema.

Neste diagrama, cada ator representa um dos perfis de acesso definidos para a aplicação integradora. O ator Sistema externo simboliza um sistema que pode ser da própria administração pública, de outro ente público ou de uma organização contratada para prestar serviços ao município. Os atores Usuário e Administrador representam aqueles que interagem com a aplicação integradora a partir da aplicação de administração.

Figura 6 - Diagrama de caso de uso



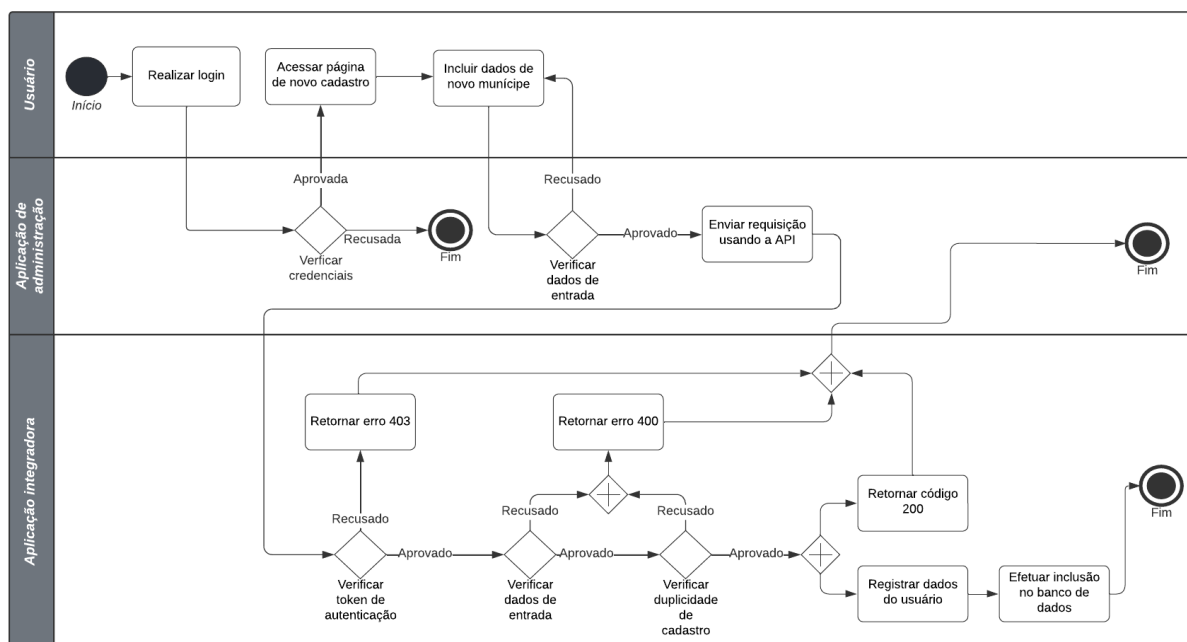
Fonte: De autoria própria (2022).

### 3.4 DIAGRAMA DE ATIVIDADE

Um diagrama de atividade representa o comportamento de um sistema ou parte dele por meio de um fluxo de controle que percorre as ações executadas pelo sistema, de acordo com Pressman (2011).

A figura 7 mostra o diagrama atividade do processo de inclusão de dados de um novo município a partir da aplicação de administração.

Figura 7 - Diagrama de atividade: Inclusão de novo munícipe



Fonte: De autoria própria (2023).

### 3.5 MODELAGEM DO BANCO DE DADOS

A modelagem de dados é o ato que envolve a exploração de estruturas voltadas para dados e pode ser utilizada com diversos propósitos, abrangendo desde modelos conceituais de alto nível até modelos físicos de dados. Na modelagem de dados são identificados tipos de entidades correspondentes às classes, enquanto os atributos de dados são associados a essas entidades da mesma forma que atributos e operações são associados às classes. São estabelecidas associações entre as entidades, abrangendo conceitos como relacionamento, herança, composição e agregação (DIAS NETO, 2020).

#### 3.5.1 MODELO CONCEITUAL

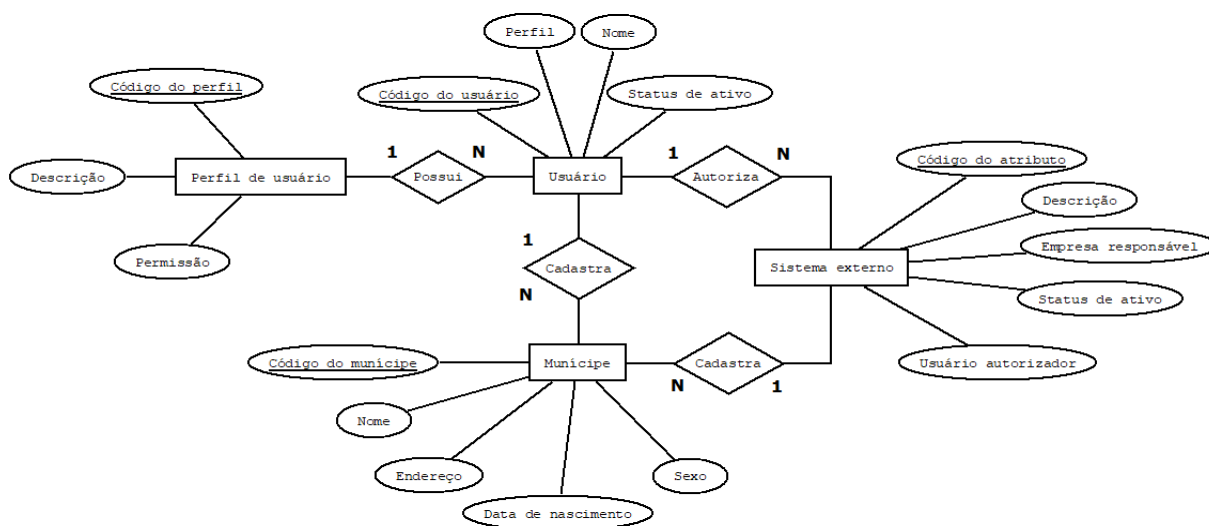
Fernigrini (2021) explica que um modelo conceitual normalmente abrange somente os principais conceitos (entidades) para armazenar as informações e os relacionamentos que existem entre as entidades. Este modelo é uma etapa inicial para a criação de um banco de dados.

Ainda de acordo com Fernigrini (2021), a especificação das entidades em alto nível, usando nomes comerciais no lugar de nomes técnicos, permite o bom entendimento do público geral, como gestores de negócios e usuários.

A figura 8 possui a representação do modelo conceitual utilizado como base

para a criação do banco de dados do sistema. Foi utilizado um modelo relacional, em atendimento ao RNF001.

Figura 8 - Modelo conceitual



Fonte: De autoria própria (2022).

### 3.5.2 MODELO FÍSICO

Segundo Fernigrini (2021), um modelo de dados físico é derivado de um modelo lógico para determinado SGBD. A grande diferença é que aqui são utilizados os nomes das tabelas e colunas ao invés das entidades e atributos. Além disso, são fornecidos os tipos dos dados, as restrições e tabelas adicionais. Isso possibilita ajustar-se aos limites e convenções do banco de dados pretendido.

A figura 9 possui a representação do modelo físico do banco de dados da aplicação integradora.

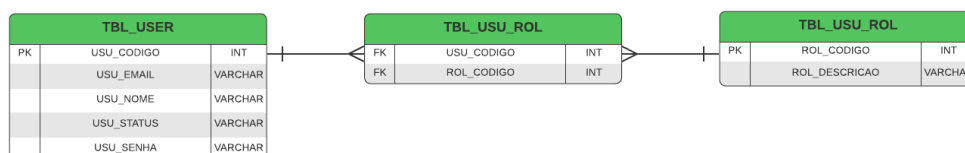
Figura 9 - Modelo físico



Fonte: De autoria própria (2022).

A figura 10 exibe a representação do modelo físico do banco de dados utilizado na construção da aplicação de administração.

Figura 10 - Modelo físico - Aplicação de administração



Fonte: De autoria própria (2023).

### 3.6 ORGANIZAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

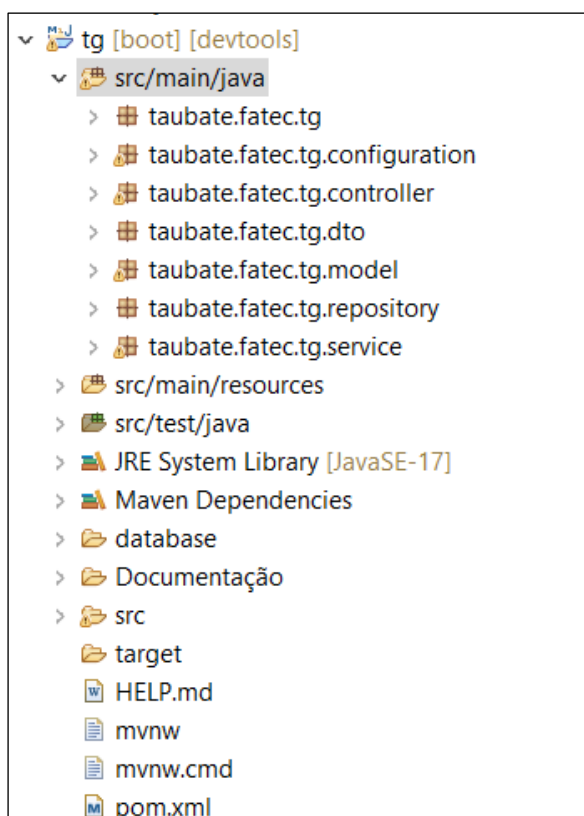
O desenvolvimento do trabalho proposto utilizou-se de ferramentas tecnológicas que necessitaram ser organizadas de maneira adequada ao projeto.

#### 3.6.1 IDE *SPRING TOOLS* E ORGANIZAÇÃO DOS PACOTES

O sistema foi criado utilizando o gerenciador de pacotes Maven. A estrutura de arquivos e pacotes do projeto, realizada na IDE *Spring Tools*, foi feita observando boas práticas de desenvolvimento, organizada nos pacotes: *configuration*, *controller*, *dto*, *model*, *repository* e *service*.

A figura 11 mostra a estrutura do projeto da aplicação integradora desenvolvida na IDE *Spring Tools*.

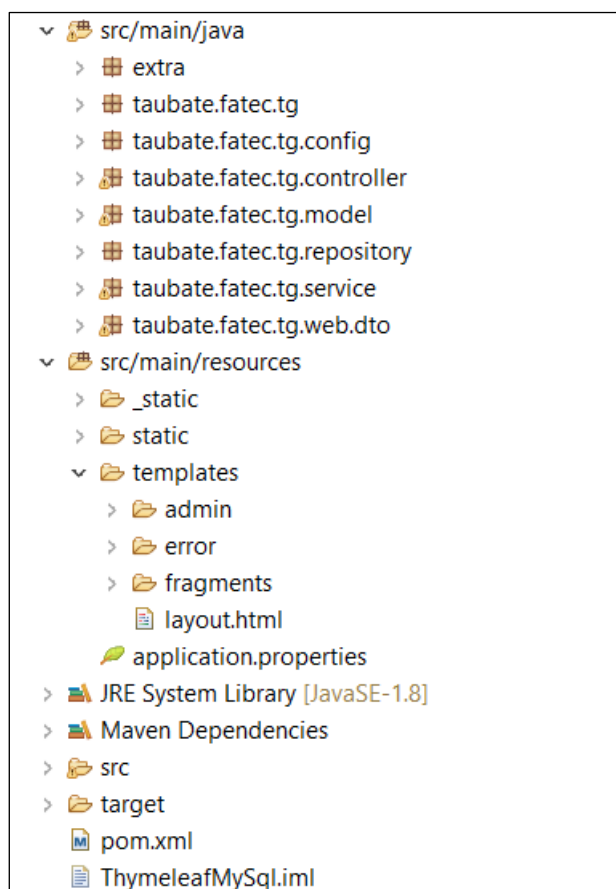
Figura 11 - Estrutura do projeto da aplicação integradora



Fonte: De autoria própria (2023).

A aplicação de administração foi desenvolvida utilizando a arquitetura *Web*, em atendimento ao RNF002. O desenvolvimento da aplicação também se deu utilizando a IDE *Spring Tools*, bem como a organização dos pacotes de maneira similar àquela utilizada na aplicação integradora, conforme exibido na figura 12.

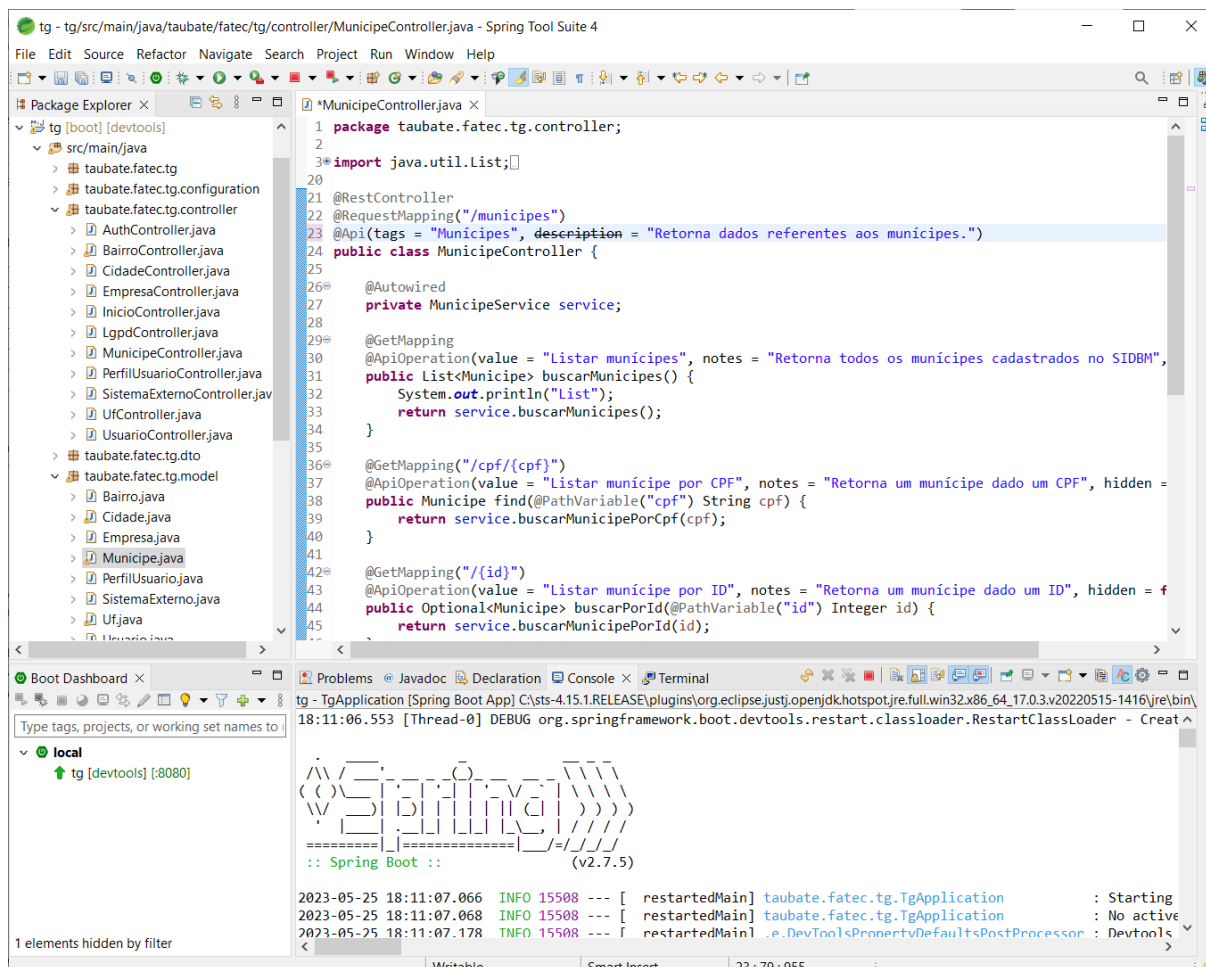
Figura 12 - Organização dos pacotes da aplicação de administração



Fonte: De autoria própria (2023).

A utilização dos *Workspaces* da IDE Spring Tools possibilitou a separação do desenvolvimento das duas aplicações. A figura 13 mostra a tela da IDE durante a construção da aplicação integradora.

Figura 13 - IDE Spring Tools durante o desenvolvimento



Fonte: De autoria própria (2023).

### 3.6.2 ORGANIZAÇÃO DO AMBIENTE DE TESTE

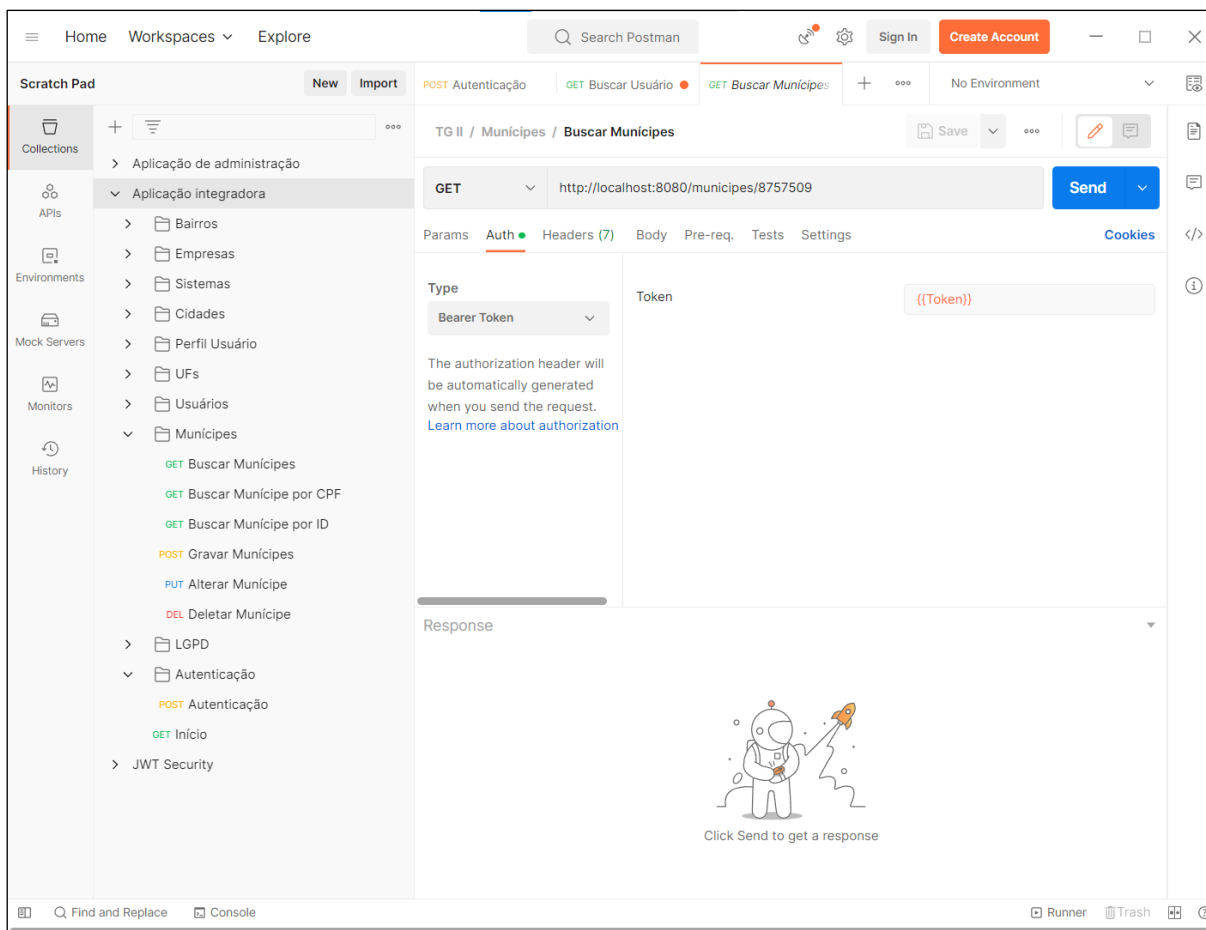
Para a realização de testes rápidos de validação dos *endpoints* criados, foi utilizada a ferramenta Postman. A funcionalidade *Collection* permite a organização em formatos de pastas e a inclusão das requisições de testes nestas pastas. Desta forma, criou-se duas *Collections*, uma para a aplicação integradora e outra para a aplicação de administração. Nas *Collections* foram criadas pastas para os *endpoints* disponibilizados e nas pastas foram criadas as requisições para cada verbo HTTP utilizado naquele *endpoint*.

Já a funcionalidade *Environment* permite a criação de um ambiente com variáveis que podem ser utilizadas em todas as requisições durante os testes. Essa funcionalidade mostra-se útil no momento em que a autenticação passa a ser obrigatória, pois é possível realizar a autenticação uma vez e criar uma variável com o *token* de acesso e então utilizar este *token* de acesso nas diversas requisições



seguintes, eliminando a necessidade de realizar o processo de autenticação inúmeras vezes. A figura 14 exibe o ambiente criado na ferramenta Postman para utilização nos testes durante o desenvolvimento da aplicação integradora.

Figura 14 - Postman



Fonte: De autoria própria (2023).

Foi elaborada uma planilha para acompanhamento e apoio na construção dos *endpoints*, servindo como controle das etapas nas de construção e configuração de cada *endpoint* envolvido na aplicação integradora. A figura 15 mostra a planilha de controle durante a fase de desenvolvimento, quando ainda havia tarefas pendentes.

Figura 15 - Planilha de controle dos endpoints

Endpoint (/inicio)					
Verbo HTTP	URI	Operação	Permissão	Teste	Status
GET	/inicio	Retorna boas vindas	admin, sistema, consulta	OK	OK
Endpoint (/status)					
Verbo HTTP	URI	Operação	Permissão	Teste	Status
GET	/status	Retorna o status do serviço	admin, sistema, consulta	OK	OK
Endpoint (/municipes)					
Verbo HTTP	URI	Operação	Permissão	Teste	Status
GET	/municipes/{cpf}	Pesquisar usuário	admin, sistema, consulta	OK	Corrigir a busca por CPF
POST	/municipes	Salvar novo usuário	admin, sistema	OK	OK
PUT	/municipes/{cpf}	Alterar usuário	admin, sistema	OK	Corrigir a busca por CPF
DELETE	/municipes/{cpf}	Deletar usuário	admin, sistema	OK	Corrigir a busca por CPF
Endpoint (/lgpd)					
Verbo HTTP	URI	Operação	Permissão	Teste	Status
GET	/lgpd	Lista todos os municípios	admin, sistema, consulta	OK	OK
POST	/lgpd/{cpf}	Inclui um município na lista de exclusão	admin, sistema	OK	OK
Endpoint (/sistemas)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/sistemas/{id}	Pesquisar sistema	admin, sistema, consulta	OK	OK
POST	/sistemas	Salvar novo sistema	admin	OK	OK
PUT	/sistemas/{id}	Alterar sistema	admin	-	Verificar
DELETE	/sistemas/{id}	Deletar sistema	admin	-	Validar o ID
Endpoint (/bairros)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/bairros/{id}	Pesquisar bairro	admin, sistema, consulta	OK	OK
POST	/bairros	Salvar novo bairro	admin	OK	OK
PUT	/bairros/{id}	Alterar bairro	admin	OK	OK
DELETE	/bairros/{id}	Deletar bairro	admin	OK	OK
Endpoint (/cidades)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/cidades/{id}	Pesquisar cidade	admin, sistema, consulta	OK	OK
POST	/cidades	Salvar novo cidade	admin	OK	OK
PUT	/cidades/{id}	Alterar cidade	admin	OK	OK
DELETE	/cidades/{id}	Deletar cidade	admin	OK	OK
Endpoint (/empresas)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/empresas/{id}	Pesquisar empresa	admin, sistema, consulta	OK	OK
POST	/empresas	Salvar novo empresa	admin	OK	OK
PUT	/empresas/{id}	Alterar empresa	admin	OK	OK
DELETE	/empresas/{id}	Deletar empresa	admin	OK	OK
Endpoint (/perfisusuuario)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/perfisusuuario/{id}	Pesquisar perfil de usuário	admin, sistema, consulta	OK	OK
POST	/perfisusuuario	Salvar novo empresa	admin	OK	OK
PUT	/perfisusuuario/{id}	Alterar empresa	admin	-	Verificar
DELETE	/perfisusuuario/{id}	Deletar empresa	admin	OK	OK
Endpoint (/ufs)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/ufs/{id}	Pesquisar perfil de usuário	admin, sistema, consulta	OK	OK
POST	/ufs	Salvar novo empresa	admin	OK	OK
PUT	/ufs/{id}	Alterar empresa	admin	OK	Verificar
DELETE	/ufs/{id}	Deletar empresa	admin	OK	OK
Endpoint (/usuarios)					
Verbo HTTP	URI	Operação	Permissão	Testes	Status
GET	/usuarios/{id}	Pesquisar usuário	admin, sistema, consulta	OK	OK
POST	/usuarios	Salvar novo usuário	admin	OK	OK
PUT	/usuarios/{id}	Alterar usuário	admin	-	Verificar
DELETE	/usuarios/{id}	Deletar usuário	admin	OK	OK

Fonte: De autoria própria (2023).

### 3.7 CONFIGURAÇÃO DA DOCUMENTAÇÃO COM SWAGGER

Para o atendimento do Requisito não funcional RNF004, foi implementada na aplicação integradora a utilização do *framework* Swagger em sua versão 2, realizada através da biblioteca SpringFox 3. Os *endpoints* foram identificados com a utilização da anotação *@ApiOperation*, possibilitando a inclusão de uma descrição para cada *endpoint*, bem como a definição de exibição na página pública de documentação.

A possibilidade de manipular individualmente as regras de exibição dos *endpoints*, inclusive nos métodos HTTP permitidos, resulta em um maior controle, dado que alguns *endpoints* são de utilização interna, permitida apenas para a aplicação de administração, assim não há a necessidade de exibição na página de documentação pública.

### 3.8 CONFIGURAÇÃO DA CAMADA DE SEGURANÇA

A camada de segurança adicionada à aplicação integradora utilizou-se do *framework* Spring Security, permitindo o atendimento aos requisitos funcionais RF005 e RF006, bem como os requisitos não funcionais RNF007 e RNF008.

A utilização do *framework*, através da classe de configuração *SecurityFilterChain*, permite a definição de acesso autenticado ao *endpoints* disponibilizados pela aplicação integradora.

Dentre todos os *endpoints* desenvolvidos, apenas os seguintes possuem acesso livre, sem a necessidade de autenticação:

- /inicio com o método *get*;
- /login com o método *post*, e;
- Todos os relacionados ao Swagger para exibição da documentação.

Para melhor gerenciamento destas exceções, foi criada uma lista denominada AUTH\_WHITELIST.

Ainda através das configurações do *SecurityFilterChain* foi possível a definição de uma comunicação sem estado, ou seja, sem o estabelecimento de uma sessão entre a aplicação integradora e os clientes. Este tipo comunicação é possível graças à utilização do *Json Web Token* (JWT) nas requisições.

A validação dos dados de autenticação recebidos na no body da requisição HTTP é feita pela classe *Authentication*, caso as credenciais estejam corretas, o *service TokenService* é responsável por gerar um novo token e retornar ao cliente.

O token é gerado utilizando o método de criptografia Bcrypt com o algoritmo HMAC256, e possui os seguintes campos no *payload*:

- *iss* – Nome do sistema;
- *sub* – Nome do usuário;
- *id* – Código do usuário;
- *exp* – Validade do token.

A validação do token recebido na requisição HTTP no campo definido como “*Authorization*”, inicialmente é feita pelo *SecurityFilterChain*, com auxílio do método *doFilterToken*, que extrai as informações do token e verifica seu formato. Após isso o *TokenService* é responsável por validar os dados de autenticação e então retorna se estão corretos. Com esse retorno, *SecurityFilterChain* finalmente autoriza ou não o acesso através das políticas de permissão estabelecidas.

Já na aplicação de administração, a camada de segurança foi desenvolvida utilizando também o framework Spring Security, porém através da classe *WebSecurityConfigurerAdapter* em conjunto, que permite estabelecer a comunicação segura com o cliente através de uma sessão. Para realizar a autenticação dos usuários, foi utilizado o método *authenticationProvider*. Os dados de autenticação dos usuários da aplicação de administração são armazenados no banco de dados da própria aplicação, sem relação direta com os dados da aplicação integradora.

### 3.9 INTERFACE DO USUÁRIO

A aplicação de administração, devido sua natureza, bem como o atendimento das RF001 à RF006 e a RNF002, necessita do desenvolvimento de uma interface gráfica para utilização dos usuários que realizarão as tarefas administrativas relacionadas à aplicação integradora.

Para a construção da interface de usuário da aplicação de administração, foi utilizado o *template* AdminLTE em conjunto com o *template engine* Thymeleaf.

O acesso à aplicação é realizado a partir de uma tela de autenticação, na URI */login*. Após a realização da autenticação, o usuário é direcionado para a página de boas-vindas, onde possui acesso a todas as funcionalidades da aplicação em uma única tela a partir do menu lateral.

## **4 RESULTADOS OBTIDOS**

Neste capítulo serão apresentados os resultados obtidos na fase de desenvolvimento. Os resultados serão abordados da ótica da aplicação integradora e da aplicação de administração.

### **4.1 APLICAÇÃO INTEGRADORA**

Para a realização dos testes, foi criada uma base de dados com informações fictícias de municípios, empresas e sistemas. Para os dados de cidades e unidades federativas, foram utilizadas informações provenientes do Instituto Brasileiro de Estatística e Geografia. Por fim, para os dados dos bairros, foi utilizada base disponibilizada pela Empresa Brasileira de Correios e Telégrafos.

#### **4.1.1 SWAGGER**

A utilização do framework Swagger aliado ao Spring Boot resultou na disponibilização de uma página dedicada à documentação da API criada, conforme é observado na figura 16.

Figura 16 – Tela de documentação da API - Swagger



Fonte: De autoria própria (2023).

A partir da implementação do Swagger, foi possível extrair dados da documentação da API em formato Json, e utilizando a ferramenta SwDoc foi possível a geração de toda documentação em formato PDF.

A disponibilização da documentação atende ao requisito não funcional RNF004.

#### 4.1.2 REPOSTAS DAS REQUISIÇÕES DA API

A figura 17 apresenta o resultado de uma requisição GET realizada através do *endpoint* `"/municipes/{cpf}"` passando como parâmetro um CPF cadastrado. Foi utilizada a ferramenta Postman para o envio da requisição.

Figura 17 - Resposta da requisição GET

GET ⌵ http://localhost:8080/municipes/61309669843

Params ● Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input type="checkbox"/>		
	Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ⌵ ≡

```

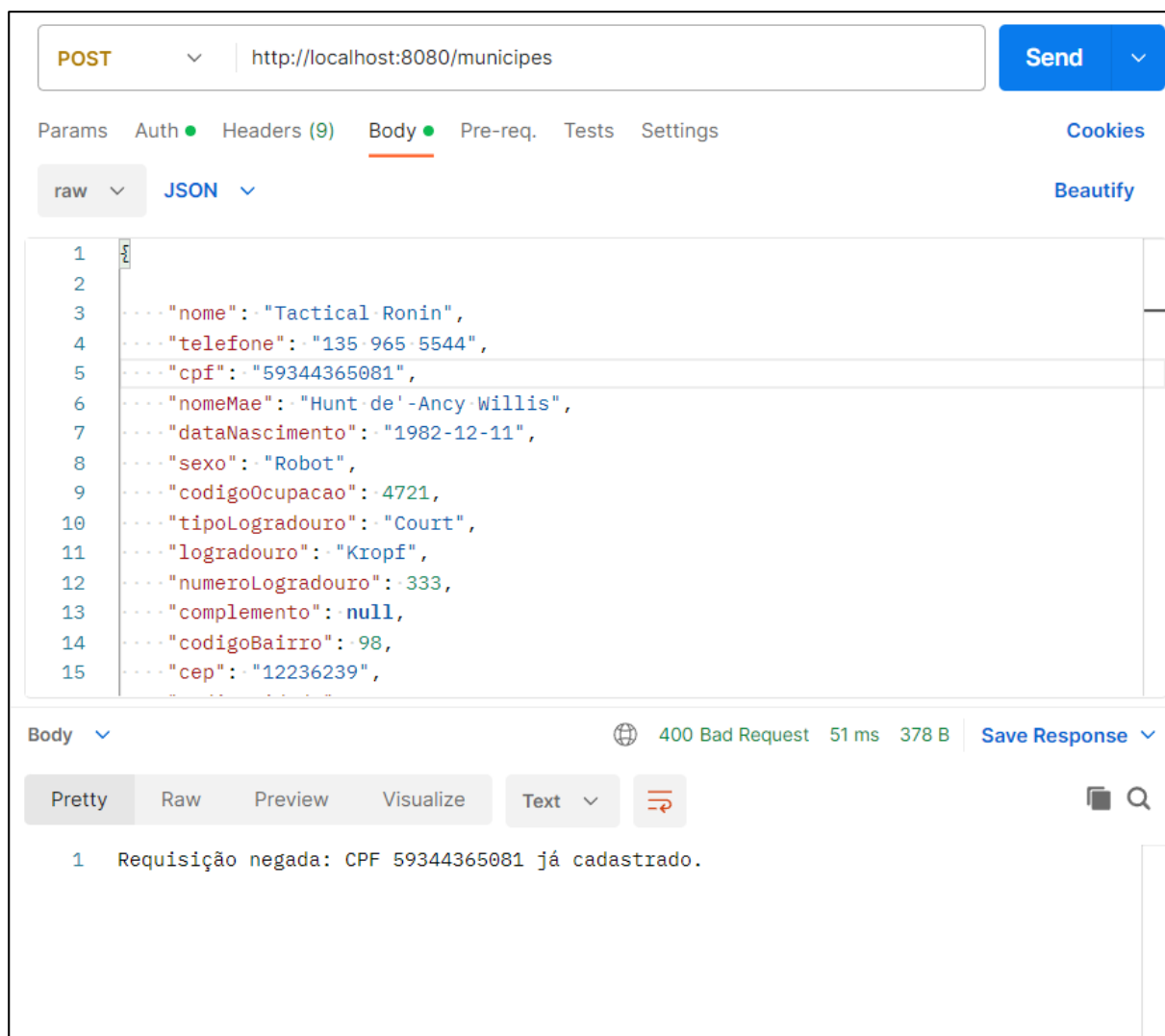
1  {
2    "codigo": 8757512,
3    "nome": "Gram Wilfinger",
4    "telefone": "135 965 1494",
5    "nomeSocial": null,
6    "cpf": "61309669843",
7    "nomeMae": "Hunt de'-Ancy Willis",
8    "dataNascimento": "1982-12-13",
9    "sexo": "Male",
10   "codigoOcupacao": 4721,
11   "tipoLogradouro": "Court",
12   "logradouro": "Kropf",
13   "numeroLogradouro": 333,
14   "complemento": null,
15   "codigoBairro": 98,
16   "cep": "12236239",
17   "codigoCidade": 355410,
18   "anoObito": null,
19   "nacionalidade": "Brasileira",
20   "cidadeNascimento": "355410"

```

Fonte: De autoria própria (2023).

Já a figura 18 mostra o resultado da tentativa de inserção um munícipe que já possui cadastro. A requisição é retornada com o código 400 *bad request* e uma mensagem de aviso.

Figura 18 - Resposta da requisição POST após validação

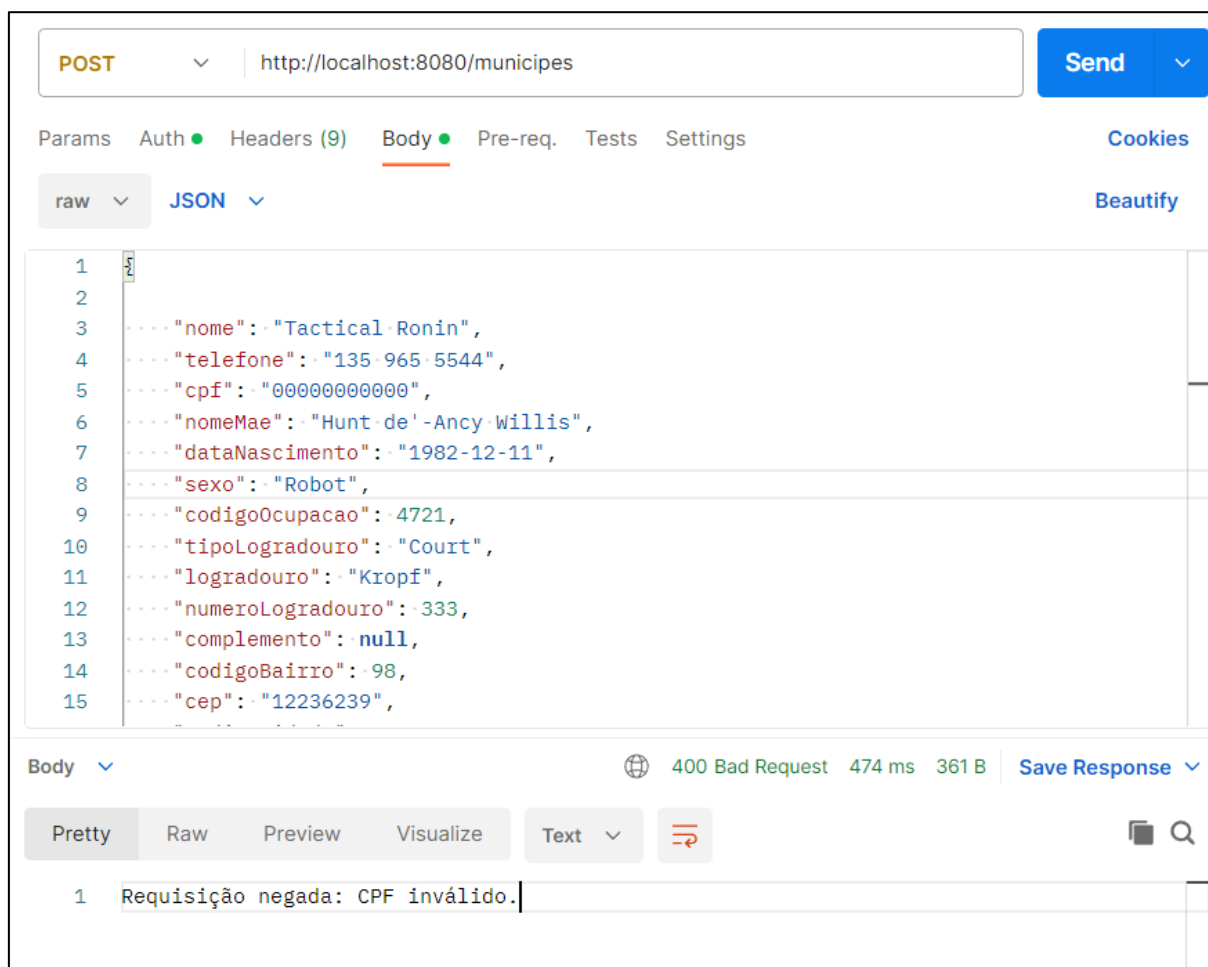


Fonte: De autoria própria (2023).

Já a figura 19 exibe o retorno de uma requisição POST na URI /municipes para a inserção de um novo munícipe, porém foi utilizando um número de CPF inválido, o que resultou em uma mensagem de erro 400.



Figura 19 - Resposta da requisição POST após validação



Fonte: De autoria própria (2023).

#### 4.1.3 APLICAÇÃO SEGURA UTILIZANDO JWT

Para assegurar os requisitos de segurança, foi implementada a autenticação via JWT.

A figura 20 mostra uma tentativa de acesso ao *endpoint* /usuários sem a devida autenticação, que retorna a requisição com o *status* 403 *Forbidden*, exibido no destaque.

Figura 20 - Resposta da requisição GET

The screenshot shows a web browser's developer tools interface. At the top, a GET request to `http://localhost:8080/usuarios` is shown. Below the request bar, the 'Query Params' section is empty. The 'Headers' section is expanded, showing a table of response headers. The status bar indicates a '403 Forbidden' error. The 'Test Results' section is also visible.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

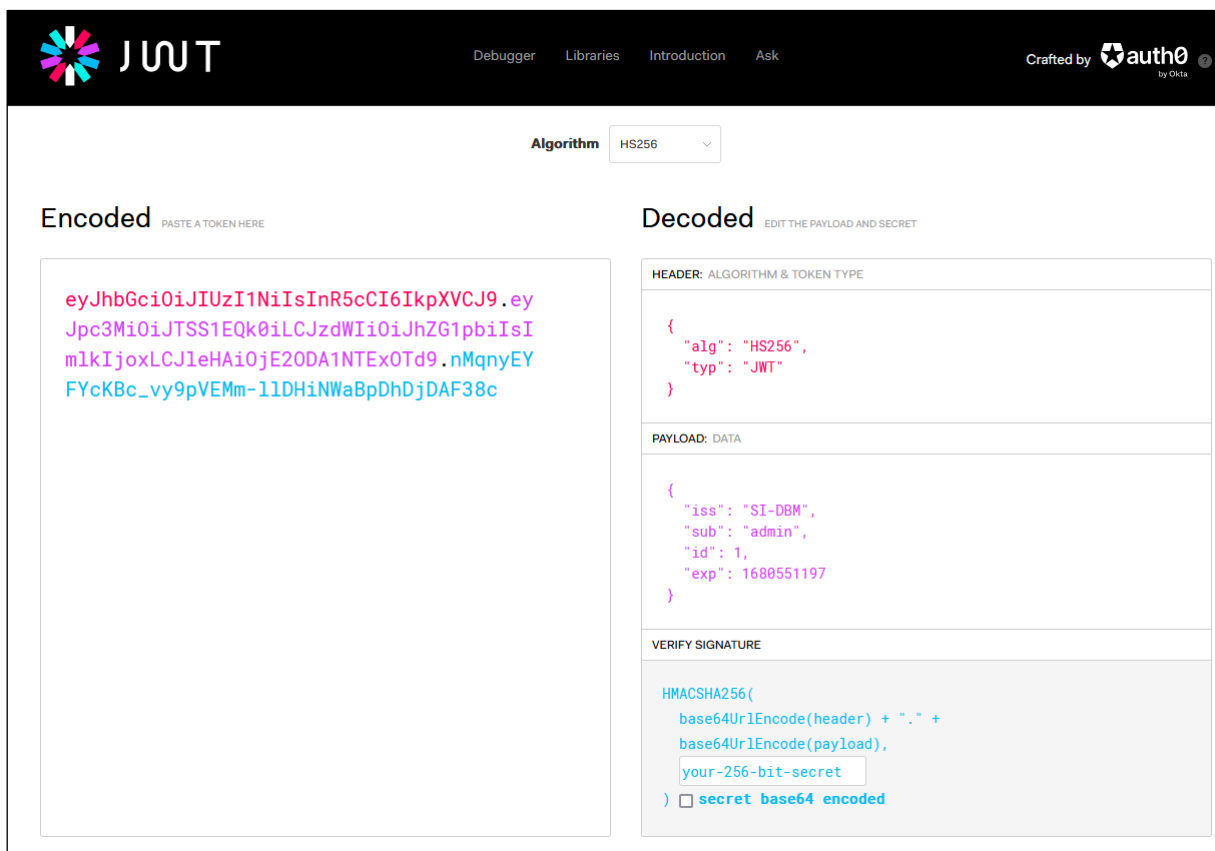
  

KEY	VALUE
X-Content-Type-Options ⓘ	nosniff
X-XSS-Protection ⓘ	1; mode=block
Cache-Control ⓘ	no-cache, no-store, max-age=0, must-revalidate
Pragma ⓘ	no-cache
Expires ⓘ	0
X-Frame-Options ⓘ	DENY

Fonte: De autoria própria (2023).

A utilização das bibliotecas do Spring Security permitiu a geração do JWT e entrega ao cliente autenticado. A figura 21 exibe informações sobre um JWT gerado e enviado ao cliente. No lado esquerdo, no campo identificado na figura como “*Encoded*” é exibido o token como é entregue. No lado direito, é mostrada as partes do JWT: *Header*, *Payload* e *Signature*.

Figura 21 - Partes e conteúdo do JWT



The screenshot shows the JWT.io interface. At the top, there's a navigation bar with links: Debugger, Libraries, Introduction, Ask, and a 'Crafted by auth0 by Okta' logo. Below the navigation bar, there's a dropdown menu for 'Algorithm' set to 'HS256'.

The main content is divided into two sections:

- Encoded:** Labeled 'PASTE A TOKEN HERE'. It contains a long, colorful string representing a JWT token: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJTSS1EQk0iLCJzZWUiOiJhZG1pbSImlkIjoxCjleHAiOiJlbnR5EYFYcKBC_vy9pVEMm-1lDHiNWaBpDhDjDAF38c`.
- Decoded:** Labeled 'EDIT THE PAYLOAD AND SECRET'. It shows the decoded structure of the token:
  - HEADER: ALGORITHM & TOKEN TYPE:**

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```
  - PAYLOAD: DATA:**

```
{
  "iss": "SI-DBM",
  "sub": "admin",
  "id": 1,
  "exp": 1680551197
}
```
  - VERIFY SIGNATURE:**

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

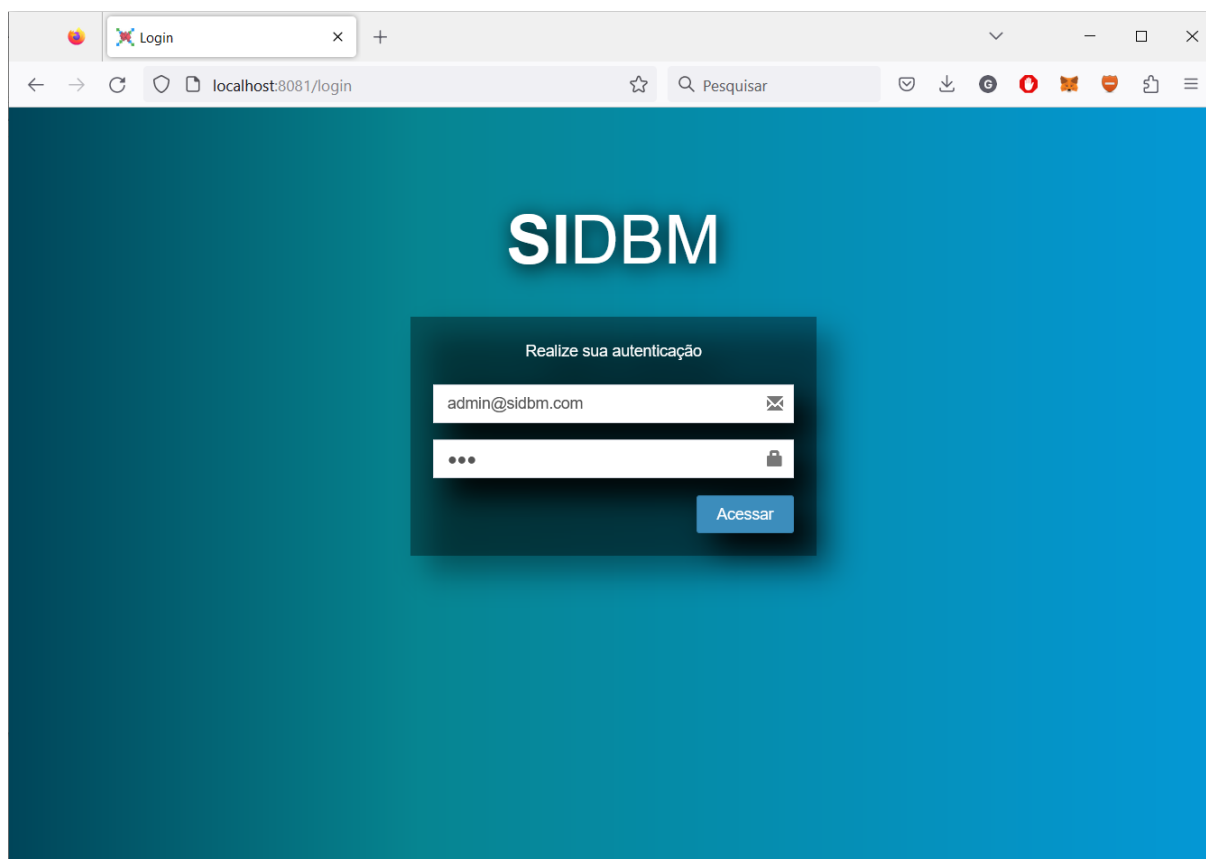
Fonte: De autoria própria (2023).

## 4.2 APLICAÇÃO DE ADMINISTRAÇÃO

A utilização do framework Spring com o Thymeleaf e ainda utilizando como modelo base o AdminLTE resultou em um ambiente dentro do especificado inicialmente.

O acesso à aplicação se dá através de navegador web, onde inicialmente é apresentada a página de autenticação, conforme é exibido na figura 22.

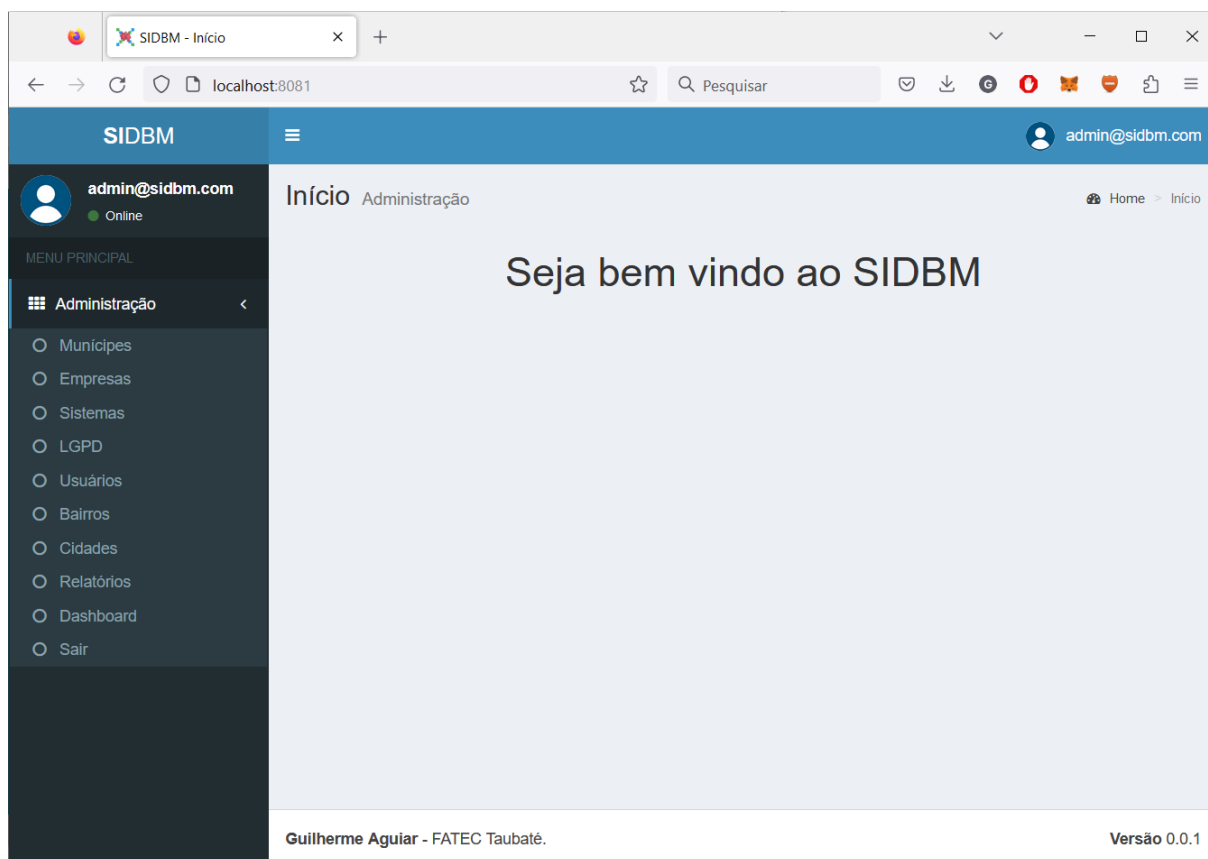
Figura 22 - Tela de login



Fonte: De autoria própria (2023).

A figura 23 mostra o resultado obtido com a implementação da solução proposta utilizando o *template* AdminLTE.

Figura 23 - Tela inicial



Fonte: De autoria própria (2023).

O menu lateral oferece acesso a todas as funcionalidades permitidas pelo sistema, de acordo com o perfil de acesso. Para fins de demonstração, os resultados aqui apresentados foram gerados a com acesso de um perfil de administrador.

A figura 24 exibe a lista de municípios cadastrados. É possível realizar a busca de um município em particular através do campo de busca, localizado logo acima da lista.

Figura 24 - Tela com a lista de munícipes

The screenshot displays the 'SIDBM - Municípios' web application. The browser address bar shows 'localhost:8081/municipe/list'. The application has a dark sidebar menu on the left with the 'Administração' section expanded, showing options like 'Municípios', 'Empresas', 'Sistemas', 'LGPD', 'Usuários', 'Bairros', 'Cidades', 'Relatórios', 'Dashboard', and 'Sair'. The main content area is titled 'Lista de munícipes' and includes a search bar and a '+ Novo cadastro' button. Below these is a table with the following data:

ID	CPF	Nome	Data de nascimento	Sexo	Ações
8757509	34307115608	Alec Jeans	1968-10-14	Male	<a href="#">Edit</a> <a href="#">Delete</a>
8757510	57537057353	Marji Cardno	1954-03-30	Female	<a href="#">Edit</a> <a href="#">Delete</a>
8757511	73663282393	Jami McAlarney	1980-04-11	Female	<a href="#">Edit</a> <a href="#">Delete</a>
8757512	61309669843	Gram Wilfinger	1982-12-13	Male	<a href="#">Edit</a> <a href="#">Delete</a>
8757513	45465892180	Adair Kabisch	2001-09-13	Agender	<a href="#">Edit</a> <a href="#">Delete</a>
8757514	78534907044	Vina Hamelyn	1999-07-05	Female	<a href="#">Edit</a> <a href="#">Delete</a>
8757515	45729171077	Sheff Maric	1948-08-24	Male	<a href="#">Edit</a> <a href="#">Delete</a>
8757516	82526629802	Ruth Wilacot	1952-04-08	Genderque	<a href="#">Edit</a> <a href="#">Delete</a>
8757517	80160887253	Merrile Peltzer	1952-03-23	Female	<a href="#">Edit</a> <a href="#">Delete</a>

At the bottom of the table area are navigation buttons: 'Anterior', 'Página 1', and 'Próxima'. The footer of the application shows 'Guilherme Aguiar - FATEC Taubaté.' on the left and 'Versão 0.0.1' on the right.

Fonte: De autoria própria (2023).

A partir desta lista, também é possível adicionar um novo cadastro clicando no botão “Novo Cadastro” na parte superior direita. Também é possível acessar a página de edição do munícipe e realizar exclusão de munícipes nos casos previstos pelo administrador do sistema integrador.

Na figura 25 é exibida a tela de cadastro de um novo sistema.

Figura 25 - Tela de cadastro de novo sistema externo

The screenshot displays the 'Detalhes do Sistema Externo' (External System Details) page in the SIDBM application. The page is accessed via a web browser at the URL `localhost:8081/sistemaExterno/add`. The user is logged in as `admin@sidbm.com` and is in the 'Administração' (Administration) section.

The form contains the following fields:

- Código:** A text input field with the placeholder 'Código do Sistema Externo'.
- Descrição:** A text input field.
- Empresa:** A dropdown menu with the placeholder 'Selecione a Empresa'.
- E-mail:** A text input field.
- Preposto:** A text input field.
- Status:** A dropdown menu with the selected value 'Ativo'.

At the bottom of the form, there are two buttons: 'Cancelar' (Cancel) and 'Salvar' (Save).

The sidebar menu on the left includes the following items:

- Administração (selected)
- Municípios
- Empresas
- Sistemas
- LGPD
- Usuários - SIDBM
- Bairros
- Cidades
- Dashboard
- Sair

The footer of the page displays the text 'Guilherme Aguiar - FATEC Taubate.' on the left and 'Versao 0.0.1' on the right.

Fonte: De autoria própria (2023).

Já a figura 26 mostra a tela a alteração de um cadastro de munícipe já existente.

Figura 26 - Tela de alteração de munícipe

The screenshot displays a web browser window with the URL `localhost:8081/munícipe/edit/8757510`. The application interface includes a top navigation bar with the SIDBM logo and a user profile for `admin@sidbm.com`. A left sidebar menu lists various administrative options, with 'Administração' currently selected. The main content area is titled 'Detalhes do Múncipe' and contains a form with the following fields:

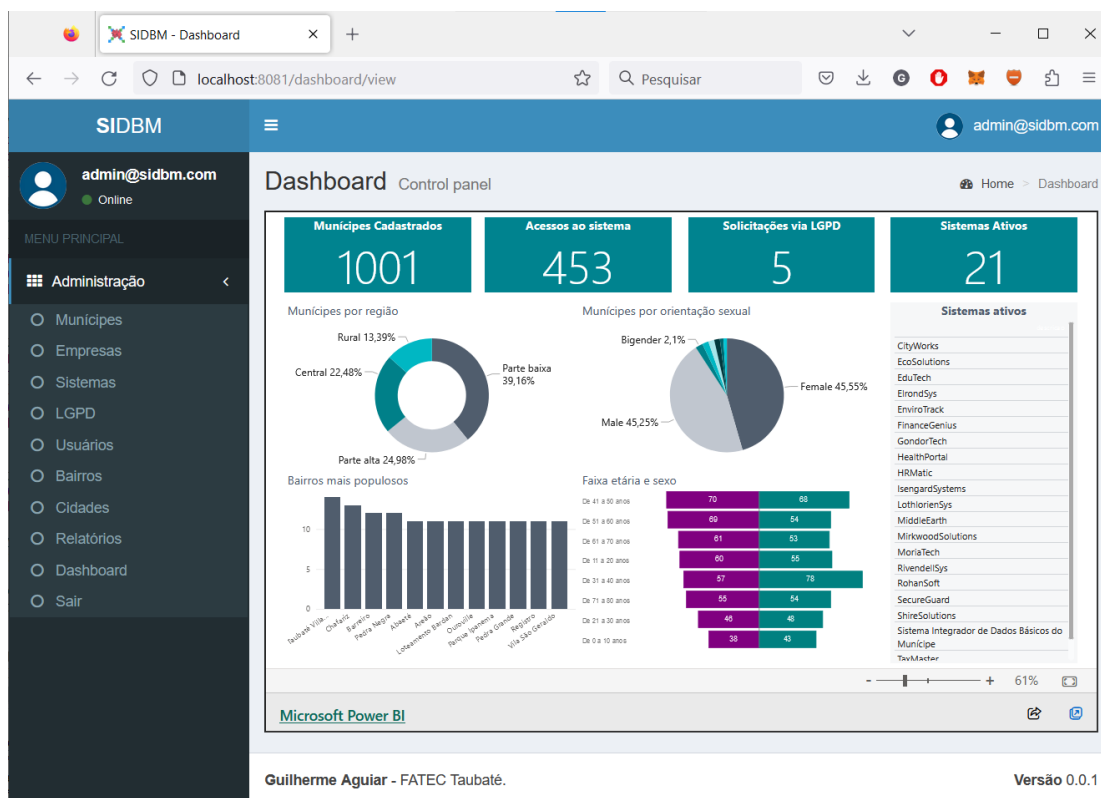
Field Label	Value
Código	8757510
Nome completo	Marji Cardno
Nome social	
Telefone	1232523615
CPF	575.370.573-53
Nome da Mãe	Penny Burchatt
Data de Nascimento	30 / 03 / 1954
Sexo	Female
Tipo de Logradouro	Lane
Logradouro	Rua Anizio Ortiz Monteiro
Número do Logradouro	46

Fonte: De autoria própria (2023).

As figuras 27 e 28 exibem a tela de dashboard, em atendimento ao RF004. O painel interativo apresentado possui duas telas, a primeira com um resumo dos dados proveniente da aplicação integradora e a segunda com um mapa de calor preenchido com quantidade de munícipes de acordo com o endereço de residência. Os filtros laterais permitem alterar a visualização de acordo com a faixa etária fixada, bem como a separação por sexo.

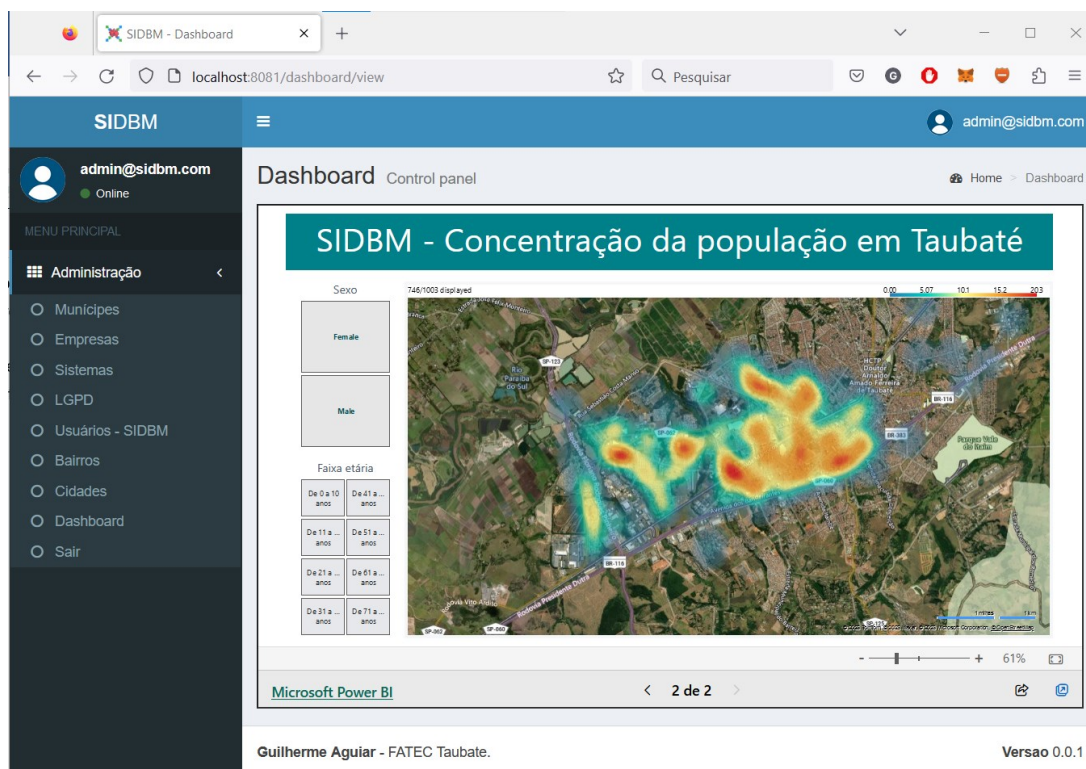


Figura 27 - Tela do dashboard 1



Fonte: De autoria própria (2023).

Figura 28 - Tela do dashboard 2



Fonte: De autoria própria (2023).

## 5 CONCLUSÃO

O cenário atual da informatização da administração pública demonstra uma necessidade de aprimorar a gestão de dados e aproveitar melhor as vantagens possibilitadas por uma política orientada a dados. A diversidade de serviços prestados à população, em muitos casos, traz uma diversidade na contratação de soluções tecnológicas. A proposta de criar um sistema capaz de integrar e padronizar os dados básicos dos municípios apresentou perspectivas de solucionar os problemas gerados pela heterogeneidade de soluções tecnológicas contratadas pelas administrações municipais.

A escolha do desenvolvimento da proposta utilizando ferramentas de grande aceitação no mercado e bem consolidadas, se provou correta ao longo do processo, tanto na busca de soluções para os desafios encontrados quanto na estabilidade da plataforma.

Os resultados obtidos ficaram dentro esperado, possibilitando uma comunicação eficaz entre sistemas externo e a aplicação integradora. A escolha do desenvolvimento da aplicação de administração de forma separada da aplicação integradora possibilitou colocar à prova sua funcionalidade ao utilizar-se da mesma API que será disponibilizada para as empresas que prestam serviços à administração pública municipal.

Sugere-se como trabalhos futuros a implementação de funcionalidades que não fizeram parte do escopo ou ainda que não puderam ser atingidas durante o desenvolvimento deste trabalho, como melhoria na validação de dados de entrada na aplicação de administração, aprimoramento do registro de informações dos usuários da aplicação integradora para rastreabilidade e por fim, possibilitar e auditoria e leitura de *logs* através da aplicação de administração.

Diante disso, a proposta de um sistema integrador de dados básicos do município, nos moldes aqui apresentados, se mostrou eficaz para solucionar os problemas apontados neste trabalho.

## REFERÊNCIAS

ABEP-TIC. **Sobre os aprimoramentos do Índice para 2022**: comparativo 2021/2022. COMPARATIVO 2021/2022. 2022. Disponível em: <https://abep-tic.org.br/indice-abep-2022/>. Acesso em: 15 abr. 2023.

ADMINLTE. **AdminLTE Bootstrap Admin Dashboard Template**. Disponível em: <https://adminlte.io>. Acesso em: 22 abr. 2023.

AGRAWAL, Shubhangi Raj. **The 7 RESTful routes!** 2018. Disponível em: <https://medium.com/@shubhangirajagrawal/the-7-restful-routes-a8e84201f206>. Acesso em: 22 maio 2022.

APACHE SOFTWARE FOUNDATION. **Trilha do Aprendizado do Java EE e Java Web**. Disponível em: [https://netbeans.apache.org/kb/docs/java-ee\\_pt\\_BR.html](https://netbeans.apache.org/kb/docs/java-ee_pt_BR.html). Acesso em: 22 maio 2022.

BAELDUNG. **Generate PDF from Swagger API Documentation**. 2023. Disponível em: <https://www.baeldung.com/swagger-generate-pdf>. Acesso em: 21 abr. 2023.

BARBIERI, Carlos. **Governança de Dados**: práticas, conceitos e novos caminhos. Rio de Janeiro: Alta Books, 2020. 288 p.

DEVMEDIA. **Introdução a web services RESTful**. 2016. Disponível em: <https://www.devmedia.com.br/introducao-a-web-services-restful/37387>. Acesso em: 26 maio 2022.

DIAS NETO, Arilo Cláudio. **Modelagem de Dados Tutorial**. 2020. Disponível em: <https://www.devmedia.com.br/modelagem-de-dados-tutorial/20398>. Acesso em: 21 abr. 2023.

ECLIPSE. **About the Eclipse Foundation**. 2022. Disponível em: <https://www.eclipse.org/org/>. Acesso em: 06 set. 2022.

FERNIGRINI, Lisandro. **What Are Conceptual, Logical, and Physical Data Models?** 2021. Disponível em: <https://vertabelo.com/blog/conceptual-logical-physical-data-model/>. Acesso em: 14 nov. 2022.

FERREIRA, Rodrigo. **REST: princípios e boas práticas.** Princípios e boas práticas. 2017. Disponível em: <https://www.alura.com.br/artigos/rest-principios-e-boas-praticas>. Acesso em: 29 maio 2022.

GOV.BR (org.). **Cadastro Base do Cidadão:** governança de dados, cadastro base do cidadão - cbc. Governança de Dados, Cadastro Base do Cidadão - CBC. 2023. Disponível em: <https://www.gov.br/governodigital/pt-br/governanca-de-dados/cadastro-base-do-cidadao-cbc>. Acesso em: 21 abr. 2023a.

\_\_\_\_\_. **Cadastro Base do Cidadão (CBC - CPF):** ministério da economia (me). Ministério da Economia (ME). 2022. Disponível em: <https://www.gov.br/conecta/catalogo/apis/cadastro-base-do-cidadao-cbc-cpf>. Acesso em: 21 abr. 2023b.

HORWITZ, Lauren; SCARDINA, Jesse. **Microsoft Power BI.** 2022. Disponível em: <https://www.techtarget.com/searchcontentmanagement/definition/Microsoft-Power-BI>. Acesso em: 20 abr. 2023.

KAINULAINEN, Petri. **Spring Data.** Birmingham: Packt Publishing, 2012. 160 p.

KHOUMBATI, Khalil; KALHORO, M.S; BUKHARI, Asadulla Shah. **EVALUATING THE SYSTEMS INTEGRATION TECHNOLOGIES IN THE PUBLIC SECTOR ORGANISATIONS.** 2008. Disponível em: [https://www.academia.edu/2688343/EVALUATING\\_THE\\_SYSTEMS\\_INTEGRATION\\_TECHNOLOGIES\\_IN\\_THE\\_PUBLIC\\_SECTOR\\_ORGANISATIONS](https://www.academia.edu/2688343/EVALUATING_THE_SYSTEMS_INTEGRATION_TECHNOLOGIES_IN_THE_PUBLIC_SECTOR_ORGANISATIONS). Acesso em: 20 maio 2023.

LECHETA, Ricardo. **Web services RESTful:** aprenda a criar web services restful em java na nuvem do google. São Paulo: Novatec Editora, 2015. 432 p.

LUCA, Cristina de; BASSI, Silvia. **Potencializando o uso de Big Data para cidades inteligentes: Um guia estratégico para gestores.** Um guia estratégico para gestores. 2023. Disponível em: <https://publications.iadb.org/publications/portuguese/viewer/Potencializando-o-uso-de-big-data-para-cidades-inteligentes-um-guia-estrategico-para-gestores.pdf>. Acesso em: 11 maio 2023.

MARIADB. **About MariaDB Server.** 2022. Disponível em: <https://mariadb.org/about/>. Acesso em: 14 nov. 2022.

MENDES, Douglas Rocha. **Programação Java com Ênfase em Orientação a Objetos.** São Paulo: Novatec Editora, 2009.

MENDES, Rafael Vieira. **Desenvolvimento de uma ferramenta para organização e gerenciamento de atividades de docentes.** Orientador: Prof. Dr. André Ricardo Backes. 2018. 31 p. Trabalho de conclusão de curso (Bacharel em Sistemas da Informação) - Universidade Federal de Uberlândia, Uberlândia, 2018. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/22185/1/DesenvolvimentoFerramentaOrganizacao.pdf>. Acesso em: 22 abr. 2023.

MICROSOFT. **O que é Power BI?** 2023. Disponível em: <https://learn.microsoft.com/pt-br/power-bi/fundamentals/power-bi-overview>. Acesso em: 20 abr. 2023.

MULARIEN, Peter. **Spring Security 3: Secure your web applications against malicious intruders with this easy to follow practical guide.** Birmingham: Packt Publishing, 2010. 397 p.

NILCAIN, Mário. **Como realizar o Levantamento de Requisitos no desenvolvimento de software.** 2018. Disponível em: <https://blog.cedrotech.com/levantamento-de-requisitos-e-desenvolvimento-de-sofware>. Acesso em: 14 nov. 2022.

ORACLE. **What is Java technology?** 2022. Disponível em:

[https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html). Acesso em: 14 nov. 2022.

PHALTANKAR, Amit. **What is JPA, Spring Data and Spring Data JPA**. Disponível em: <https://www.amitph.com/jpa-and-spring-data-jpa/>. Acesso em: 14 maio 2022.

POSTMAN. **About Postman**. Disponível em: <https://www.postman.com/about/>. Acesso em: 03 abr. 2023.

PRESSMAN, Roger S.. **Engenharia de Software**: uma abordagem profissional. 7. ed. Porto Alegre: Amgh Editora Ltda., 2011. 780 p.

PROMPT SOFTTECH. **How to Use Swagger Tool for API Documentation?** 2019. Disponível em: <https://www.promptsofttech.com/blog/how-to-use-swagger-tool-for-api-documentation/>. Acesso em: 14 nov. 2022.

RÊGO, Bergson Lopes. **Gestão e Governança de Dados**: promovendo dados como ativo de valor nas empresas. Rio de Janeiro: Brasport, 2013. 312 p.

SANTANA, Eduardo Felipe Zambom. **Back-end Java**: microserviços, spring boot e kubernetes. São Paulo: Casa do Código, 2021. 155 p.

SPRING. **2. Introduction to the Spring Framework**: part i. overview of spring framework. Part I. Overview of Spring Framework. Disponível em: <https://docs.spring.io/spring-framework/docs/4.2.1.RELEASE/spring-framework-reference/html/overview.html>. Acesso em: 29 maio 2022

\_\_\_\_\_. **Spring Data**. 2021. Disponível em: <https://spring.io/projects/spring-data>. Acesso em: 26 maio 2022.

SWAGGER. **About Swagger**: the history behind swagger. The History Behind Swagger. 2022. Disponível em: <https://swagger.io/about/>. Acesso em: 14 nov. 2022.

TRIBUNAL DE CONTAS DO ESTADO DE SÃO PAULO. Tcesp. Tribunal de Contas do Estado de São Paulo. **Índice de Efetividade da Gestão Municipal (IEG-M)**. 2022.

Disponível

em:

[https://painel.tce.sp.gov.br/pentaho/api/repos/%3Apublic%3Aieg\\_m%3Aiegm.wcdf/generatedContent](https://painel.tce.sp.gov.br/pentaho/api/repos/%3Apublic%3Aieg_m%3Aiegm.wcdf/generatedContent). Acesso em: 02 abr. 2023.

WITTIG, Andreas; WITTIG, Michael. **Amazon Web Services em ação**. São Paulo: Novatec Editora, 2015. 512 p.

ZHU, Wei-Dong; DANIEL, Andrew J; A DAS, Andrew; DICKERSON, Simon; FALKL, John; SANDERS, Katherine; SHETTY, Dinesh G; WOOD, Chris. **Exposing and Managing Enterprise Services with IBM API Management**. Lindon: Vervanté, 2014. 266 p.

## **APÊNDICE A – VÍDEO DE DEMONSTRAÇÃO DOS RESULTADOS OBTIDOS**

Como demonstração prática dos resultados obtidos após o desenvolvimento da aplicação integradora e da aplicação administradora, elaborou-se vídeo com a apresentação das principais funcionalidades disponibilizadas. É demonstrada a documentação gerada utilizando o Swagger, após é apresentado envio de requisições para autenticação e inclusão de um novo munícipe utilizando o Postman. A seguir é evidenciada a validação de dados ao realizar o envio de dados contendo o campo CPF com numeração inválida. Por fim é apresentado o resultado obtido com o desenvolvimento da aplicação administradora, demonstrando todas as interfaces criadas, inclusive com a exibição do munícipe inserido anteriormente através da requisição realizado através do Postman.

Hospedou-se o vídeo demonstrativo na plataforma de vídeos YouTube, acessado pela URL <https://youtu.be/HbH9AV9rUdc>.