INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA

WILLIAN SILVA DOS SANTOS

DESENVOLVIMENTO DE UMA API PARA SOLICITAÇÃO DE REPARO EM ILUMINAÇÃO DE VIAS PÚBLICAS

VILHENA/RO

2022

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA

WILLIAN SILVA DOS SANTOS

DESENVOLVIMENTO DE UMA API PARA SOLICITAÇÃO DE REPARO EM ILUMINAÇÃO DE VIAS PÚBLICAS

Trabalho de Conclusão de Curso apresentado à Banca Examinadora do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de Rondônia, *campus* Vilhena, realizado em cumprimento de requisito para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Msc. Roberto Simplício Guimarães

VILHENA/RO

2022

FICHA CATALOGRÁFICA

Biblioteca IFRO – Campus Vilhena

S237d

SANTOS, Willian Silva dos

Desenvolvimento de uma API para solicitação de reparo em iluminação de vias públicas / Willian Silva dos Santos – Vilhena, Rondônia, 2022.

48f. ; il.

Orientador : Prof. Me. Roberto Simplício Guimarães

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO

1. API 2. Scrum 3. Typescript 4. Database 5. UMLI. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – IFRO II. Título

006.7882

Bibliotecária responsável Rosilene Maria do Couto Marques CRB 11/321

Willian Silva dos Santos

Desenvolvimento de uma API para solicitação de reparo em iluminação de vias públicas

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – campus Vilhena, realizado em cumprimento de requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Trabalho aprovado. Vilhena - RO, 12 de dezembro de 2022

Documento assinado digitalmente
gov.br
ROBERTO SIMPLICIO GUIMARAES
Data: 06/02/2023 17:39:00-0300
Verifique em https://verificador.iti.br

Msc. Roberto Simplício Guimarães
Orientador

Documento assinado digitalmente
gov.br
ERICK LEONARDO WEIL
Data: 10/02/2023 16:21:10-0300
Verifique em https://verificador.iti.br

Esp. Erick Leonardo Weil
Convidado 1

Documento assinado digitalmente
gov.br
MARCO ANTONIO AUGUSTO DE ANDRADE
Data: 07/02/2023 10:16:29-0300
Verifique em https://verificador.iti.br

Msc. Marco Antonio Augusto de Andrade
Convidado 2

Documento assinado digitalmente
gov.br
WESLEY JHONNES RAMOS ROLIM
Data: 10/02/2023 16:26:01-0300
Verifique em https://verificador.iti.br

Msc. Wesley Jhonnes Ramos Rolim
Convidado 3

Vilhena - RO
2022

Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida, a minha esposa Jaqueline, as minhas filhas Louise e Laís, ao meu pai João, a minha mãe Lucimar e aos meus irmãos Luiz e Jackeline.

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado a vida, força para enfrentar os desafios da vida e saúde pela qual pude desenvolver esse trabalho.

A minha esposa Jaqueline, por todo apoio que me deu durante toda essa etapa e minhas filhas Louise e Laís que são a razão da minha vida.

Ao corpo de docentes que contribuiram de forma majestosa no tocante ao aprendizado conquistado durante o tempo em que estive presente.

A esta instituição de ensino e ao seu corpo docente, que contribuiram de forma majestosa no tocante ao aprendizado conquistado durante o tempo em que estive presente.

Aos amigos Rodrigo, Gabriel, Íkaro, Evaldo e Wallyson com os quais tive o imenso prazer de divir parte da minha história.

Faça o teu melhor, na condição
que você tem, enquanto você
não tem condições melhores,
para fazer melhor ainda.

Mario Sergio Cortella

Resumo

Gerir um departamento ou uma secretaria responsável pela manuntenção da iluminação de vias públicas do município é um trabalho desafiador visto que a quantidade de vias tende a crescer com o aumento da população. Esse trabalho descreve o processo de desenvolvimento de uma API como parte de uma solução para esse problema, nele serão descritas as problemáticas evidentes do negócio, objetivos para superar os obstáculos elencados na problemática, os diagramas estruturais e comportamentais para nortear a fase de desenvolvimento, aperesentação das tecnologias utilizadas durante o desenvolvimento do produto, a metodologia utilizada durante a programação e a apresentação da aplicação como resultado.

Palavras-chave: API, Scrum, Typescript, Banco de dados, UML, Iluminação pública.

Abstract

Managing a department or secretariat responsible for maintaining the lighting of public roads in the municipality is a challenging job since the number of roads tends to grow with the increase in population. This work describes the process of developing an API as part of a solution to this problem, it will describe the obvious problems of the business objectives, to overcome the obstacles listed in the problem, behavioral and behavioral diagrams to guide the development phase, presentation of the technologies used during product development, the methodology used during programming and the presentation of the application as a result.

Keywords: API, Scrum, Typescript, Database, UML, Street lighting.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface.
CRUD	Acrônimo para Create (criação), Read (consulta), Update (atualização) e Delete (destruição) de dados.
MVP	Minimum Viable Product.
QA	Quality Assurance.
REST	Representational State Transfer.
HTTP	Hypertext Transfer Protocol.
SGBD	Sistema Gerenciador de Banco de Dados.
IDE	Integrated Development Environment.
NPM	Node Package Manager.
GIT	Global Information Tracker.
PIB	Produto Interno Bruto.
WIP	Trabalho em progresso.

Sumário

1 Introdução

1.1 Contexto e problema

Em 4 de setembro de 1882, Thomas Edison ligou pela primeira vez lâmpadas elétricas em Wall Street (CORPORATION, 2022), aquele momento representou o início de um serviço atualmente conhecido por iluminação pública, segundo o artigo publicado pela revista digital AdNormas “A iluminação dos espaços públicos no período noturno por poste de luz é fundamental para qualidade de vida nos centros urbanos modernos” (GRáFICA, 2020).

Com o passar do tempo e os avanços tecnológicos, a demanda por esse serviço cresceu de forma tão expressiva que atualmente não se tem notícias no Brasil de municípios que ainda não possuem iluminação pública pelo menos em parte de suas vias.

O serviço de iluminação pública está regulamentado no artigo 30 inciso V (BRASIL, 2002a) e artigo 149-A (BRASIL, 2002b) da Constituição Federal considerado como um serviço público de interesse local, sendo responsabilidade do município a "a obrigação de organizar e prestar, diretamente ou sob regime de concessão ou permissão, os serviços públicos"Jurídico (2014). Com o crescimento dos municípios e o aumento de vias públicas, a demanda pela iluminação pública tende a acompanhar o ritmo de crescimento e com o passar do tempo, os postes ou os equipamentos à ele ligado, tendem a apresentar problemas necessitando de reparos. Atualmente as solicitações de reparo na iluminação públicas feitas pela população à prefeitura do município de Vilhena são realizadas de forma manual por meio de contato telefônico ou aplicativo de mensagem, os dados das solicitações são armazenados em planilhas e entregues ao responsável técnico pela manutenção. O processo manual tende a ser lento, plausível à erros humanos e em grande parte inviável, informatizar parte do processo com um serviço de integração gera melhora no desempenho do processo? fornece informações essenciais extraídas em grande volume de dados?

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver uma *API REST* para receber solicitações referente a manutenção da iluminação pública nas vias de acesso, enviadas via aplicativo mobile, disponibilizar histórico das solicitações dos usuários, e permitir feedback por parte dos usuários solicitantes.

1.2.2 Objetivos específicos

- Documentar o processo atual usado pela secretaria de obras na manutenção da iluminação pública;
- Levantar de requisitos necessário para o desenvolvimento da solução.
- Elaborar diagrama de entidade e relacionamento do banco de dados.
- Elaborar diagramas UML.
- Desenvolver do produto.
- Realizar testes das funcionalidades desenvolvidas.

1.3 Justificativa

Manter um histórico das solicitações de manutenção que podem ser usados para gerar informações importantes sobre locais com problema recorrentes, permitir uma avaliação dos serviços prestados à população como feedback de atendimento e a inclusão de dados em sistema legados de gestão, desde que sejam respeitados os requisitos de tecnologia do serviço.

2 Fundamentação teórica

Neste capítulo será apresentado os princípios utilizados como base para o desenvolvimento deste trabalho.

2.1 Trabalhos similares

Durante a pesquisa e desenvolvimento desse trabalho não foram encontradas soluções de código aberto com a mesma proposta apresentada por esse trabalho, sendo encontrado apenas softwares proprietários, são eles:

- GIP Presidente Dutra[1]: Plataforma para gestão de iluminação pública da rodovia Presidente Dutra.
- GisWorks[2]: Plataforma de gerenciamento de serviços de iluminação pública.

2.2 HTTP - *Hypertext Transfer Protocol*

O *Hypertext Transfer Protocol* (HTTP) foi publicado em uma RCF(*Request for Comments*) resultado de um trabalho em conjunto entre Tim Berners-Lee, Roy Fielding e Henrik Frystyk Nielsen. (FIELDING et al., 1999) descreve o HTTP como um protocolo a nível de aplicação para sistemas de informação distribuídos, colaborativos e hipermedia. É o protocolo utilizado pela *Word Wide Web* desde 1990 para acesso de websites, aplicações web e desenvolvimento de arquiteturas para troca de informações entre diferentes *softwares* (FIELDING et al., 1999).

2.3 *WebServices*

Durante algum tempo, os sistemas com características monolíticas dominavam o mercado de software até o momento em que a demanda de mercado do mundo moderno começou a exigir troca de informações entre sistemas. Inicialmente a solução encontrada para resolver esse problema foi a troca de arquivos que ainda são utilizadas, como o CNAB(Centro Nacional de Automação Bancária)[3]. Porém a troca de arquivos apresenta alguns problemas, como por exemplo, as informações devem seguir uma sequencia determinada no arquivo, caso essa sequência seja quebrada, o sistema que irá consumir o arquivo

[1] Disponível em https://presidentedutra.aegis.app.br/
[2] Disponível em https://gestoriluminacaopublica.com.br/pt
[3] Disponível em: https://portal.febraban.org.br/pagina/3053/33/pt-br/layout-240

corre o risco de produzir informações inconsistentes. Como alternativa para esse problema, surgiu o *WebService* segundo (ALONSO et al., 2004) webservices "aplicações empresariais modulares e auto-contidas que têm interfaces abertas, orientadas para a Internet, baseadas em normas". WebService ainda utiliza arquivos para troca de informações entre as duas extremidades, porém são arquivos estruturados que permitem que as validações chequem não apenas informações obrigatórias mas também o tipo de dado enviado.

2.4 REST - *Representational State Transfer*

O REST segundo Saudate (2014) é um estilo de desenvolvimento de webservice. No REST todo o conjunto de dados trafegado pelo protocolo são denominados de recursos. Saudate (2014) Afirma que os recursos "são entidades bem definidas em sistemas, que possuem identificadores e endereços (URL' s) próprios". No REST os recursos são criados, recuperados, atualizados ou removidos usando os verbos *POST GET*, *PUT*, *DELETE* do protocolo *HTTP*

2.5 JSON - *JavaScript Object Notation*

Segundo Marrs (2017) JSON é um formato de dados que possibilita a troca de informações entre aplicativos em uma rede, normalmente por meio de API RESTful. É baseado em um subconjunto do *JavaScript Programming Language Standard ECMA-262 3rd Edition*, segundo Org (2022) as vantagens dessa estrutura se deve ao fato da facilidade de leitura para humanos e ser leve para que as máquinas possam analisar e gerar.

2.6 Método *Kanban*

Ao final do século XIX e início do século XX, período em que ocorreu a segunda revolução industrial, a indústria foi beneficiada pelas inovações que marcaram esse período, um exemplo clássico é a produção em massa provocadas pelo advento da energia elétrica e da linha de produção Schwab (2019). Porém essas mudanças geraram alguns problemas de gestão como o controle de estoque Peinado e Aguiar (2007), as pessoas que controlavam o estoque não tabalhavam diretamente na linha de produção, essa forma de abastecimento dificultava o trabalho entre o pessoal da linha de produção e do controle de estoque, a falta de comunicação entre as equipes tinham como consequências dois problemas comuns, sobravam muitas peças próximo ao montador da linha de produção ou faltavam peças e a produção parava Peinado e Aguiar (2007). Em 1953 Taiichi Ohno, executivo da área industrial da Toyta do Japão, inspirado pelo sistema de abastecimento das prateleiras de um supermercado norte-americano onde, os produtos eram distribuídos em prateleiras, retirados pelo próprio consumidor, as informações sobre os produtos estavam

escritos em pequenos cartões e a reposição era feita à medida que os produtos eram vendidos, desenvolveu o método que chamou inicialmente de "sistema de supermecado de abastecimento"onde os montadores que trabalhavam na linha de produção passaram a desempenhar o papel de "repositores"e a linha de produção era abastecida conforme as peças e matérias-primas eram utilizadas Peinado e Aguiar (2007). Com receio de que o sistema fosse copiado pela concorrência, o japoneses decidiram mudar o nome para "sistema *Kanban* de abastecimento", *Kanban* em japonês significa cartão Peinado e Aguiar (2007).

O método *Kanban* é inspirado no sistema *Kanban* de abastecimento, foi nomeado em 2007 por David J. Anderson e Corbis Anderson e Carmichael (2016) que define *Kanban* como "um método para definir, gerenciar e melhorar serviços que entregam trabalho de conhecimento, tais como serviços profissionais, atividade criativas e o designer de produtos físicos e de software"Anderson e Carmichael (2016). Segundo Anderson e Carmichael (2016) Kanban é usado para determinar, gerenciar e melhorar sistemas que fornecem serviços de valor para o cliente.

Figura 1 – Quadro *Kanban*

Fonte: Anderson e Carmichael (2016)

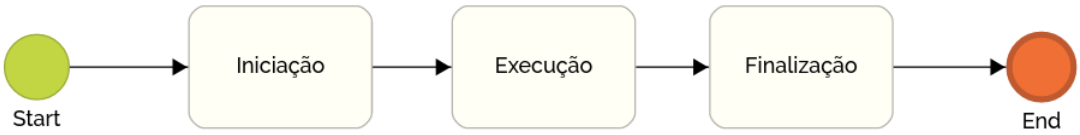
Conforme demonstrado na figura 1, o quadro *kanban* é o meio físico que torna o possível visualizar o trabalho em progresso (WIP), as atividades são descritas em cartões e distribuídas em colunas, que representam passos em um processo, segundo Anderson e Carmichael (2016) o designer do quadro não se restringe a um padrão específico.

3 Materiais e métodos

3.1 Ciclo de vida e processos

Ciclo de vida e processo utilizado no desenvolviment deste projeto é representado pela na figura 2 dividido em três etapas distintas, sendo elas:

Figura 2 – Ciclo de vida

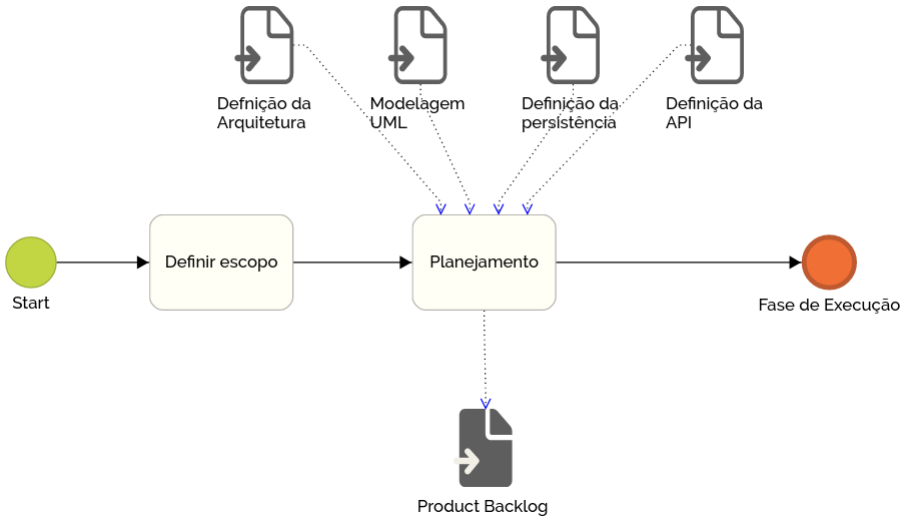


Fonte: elaborado pelo autor

3.1.1 Iniciação

Conforme demonstrado na figura 3, nessa etapa são coletados os requisitos funcionais e não funcionais para o desenvolvimento de um MVP (*Minimum Viable Product*). Em seguida os requisitos são particionados em tarefas em colocado em uma lista com ordem de prioridade ascendente denomidada *Product BackLog*

Figura 3 – Fase de iniciação

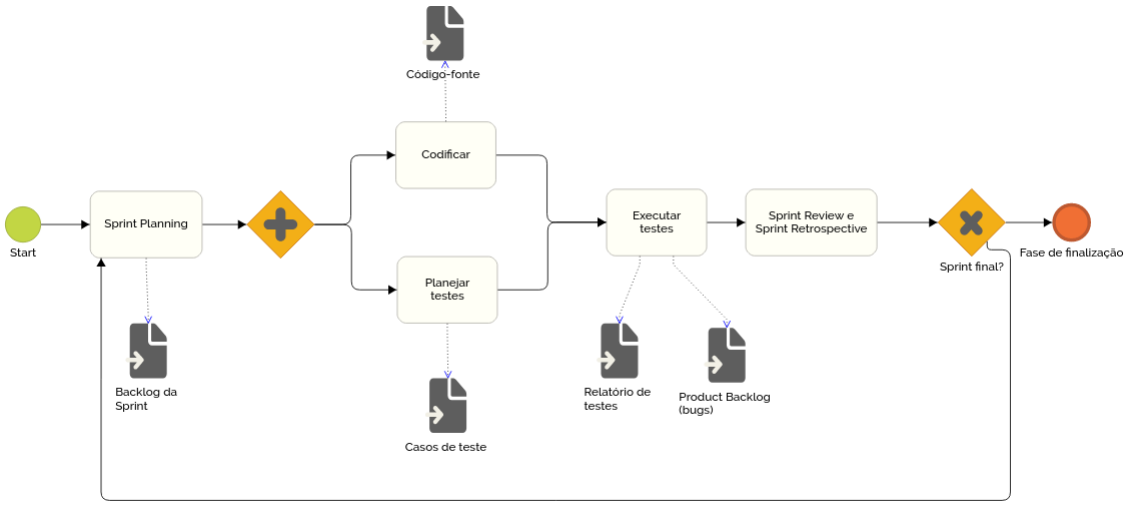


Fonte: elaborado pelo autor

3.1.2 Execução

Nessa etapa, conforme descrito na figura 4 e de acordo com o guia do scrum proposto por Schwaber (2020), acontece o planejamento da *sprint*, as atividade do *backlog* são selecionadas de acordo com a prioridade e nível de complexidade de desenvolvimento, o time de desenvolvimento se compromete com o desenvolvimento das tarefas selecionadas da *sprint*. Na sequencia as tarefas e os testes são codificados, todo o código fonte desenvolvido é testado, nesse momento, alguns *bugs* podem ser detectados e resolvidos antes do *deploy* da aplicação. O guia do scrum (SCHWABER, 2020) orienta que ao final de cada *sprint* seja realizada uma revisão da *sprint* para que seja possível inspecionar o resultado e determinar futuras adaptações quando necessárias.

Figura 4 – Fase de execução

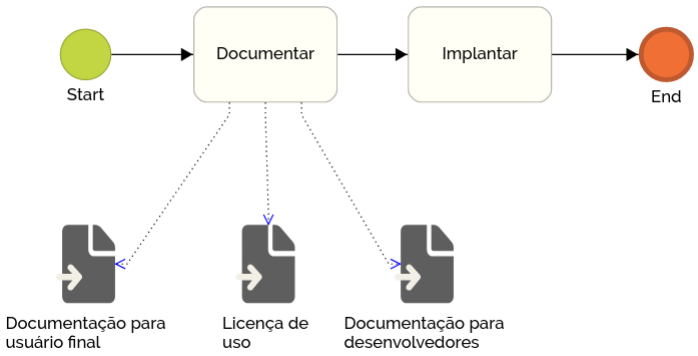


Fonte: elaborado pelo autor

3.1.3 Finalização

Nessa etapa, acontece o desenvolvimento da documentação do software, tanto para o usuário final quanto para desenvolvedores.

Figura 5 – Fase de finalização



Fonte: elaborado pelo autor

3.2 Ferramentas e tecnologias utilizadas

3.2.1 NodeJS

O NodeJs [1] é definido por Pereira (2016) como "uma plataforma esclável e de baixo nível", Pereira (2016) elencou uma lista com as vantagens sobre o uso dessa tecnologia, entre elas estão, o uso do *JavaScript* como linguagem de programação *server-side* muito difundida do mercado de desenvolvimento de software, comunidade ativa e presente no mundo inteiro, *Ready for realtime* através do protocolo *websocket* é possível realizar comunicação bi-direcional entre o cliente e servidor e o uso por *big players* como LinkedIn, Wallmart,Groupon, Microsoft, Netflix, Uber e Paypal entre outros.

3.2.2 ExpressJS

Segundo Mardan (2014) o Express[2] é um framework web baseado no modulo *http core* do NodeJs e nos componentes de conexão. É um framework muito popular, segundo dados divulgados no NPM até o momento da redação deste artigo (outubro de 2022) o Express conta com uma taxa superior a 25 milhões de *downloads* semanais.

3.2.3 NestJS

O NestJS[3] é um framework progressivo do NodeJS utilizado para criar aplicativos eficientes e escaláveis do lado do servidor. Conforme descrito na filosofia em sua documentação o NestJS fornece uma arquitetura de aplicativo pronta para uso que permite que desenvolvedores e equipes criem aplicativos altamente testáveis, escaláveis, pouco

[1] https://nodejs.org/en/
[2] https://expressjs.com/
[3] https://docs.nestjs.com/

acoplados e de fácil manutenção. A arquitetura é fortemente inspirada no Angular. É um framework muito utilizado, segundo no NPM até o momento da redação deste artigo (outubro de 2022) a taxa de download semanal é superior a um milhões e seicentos mil *downloads*.

3.2.4 TypeOrm

O TypeOrm [4] segundo a propria documentação é um ORM(*Object-Relational Mapping*) compatível com alguns *frameworks web* inclusive com NodeJS e com TypeScript(linguagem de programação utilizada pelo NestJS). Ainda segundo a propria documentação, a vantagens de se utilizar o *TypeOrm* em relação a outros está no suporte aos padrões Active Record[5] abordagem utilizada para acessar o banco de dados a partir de um modelo e Data Mapper[6] abordagem utilizada para acessar o banco de dados através de repositório e não do modelo.

3.2.5 PostgresSql

O PostgresSql [7] segundo (MILANI, 2008) é um SGBD (Sistema gerenciador de banco de dados) relacional utilizado para o armazenamento de dados. As vantagens do PostgresSQL segundo (MILANI, 2008) está relacionado a suas aplicações que podem ser utilizadas por pequenas, grandes e médias empresas, além de possuir os mesmos recursos de outros SGBD pagos no mercado. O PostgresSql não tem limite de tamanho para seu banco de dados, é *cross plataform* compatível com os seguintes sistemas operacionais: Linux(Fedora Core, Debian, SuSe,RedHat), Unix (BSD, Solaris, HP-UX, AIX), MacOS, Windows.

3.2.6 Visual Studio Code

O Visual Studio Code [8] é um IDE muito popular no mundo de desenvolvimento de software, entre as principais vantagens estão: *IntelliSense* que auxilia o desenvolvedor fornecendo sugestões para preenchimento automático de código fonte específico para cada linguagem, Integração ao GIT possibilitando o versionamento do código fonte do projeto, suporta as principais linguagens de programação e marcação do mercado entre elas: JavaSript, TypeScript, CSharp, GO, HTML, XML, Java, C, C++, Phyton, uso de extensões desenvolvidas por terceiros que abrem um leque de possibilidades que contribuem para produtividade no desempenho do programador, sendo possível a comparação entre

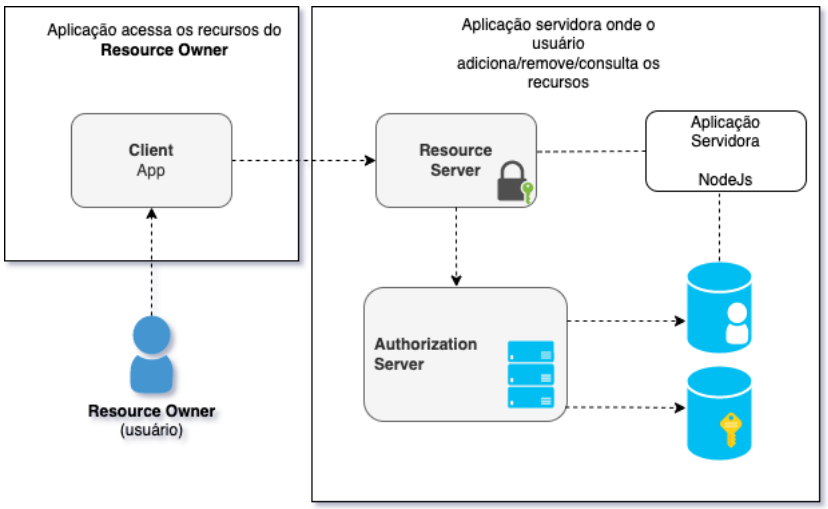
[4] https://typeorm.io/
[5] https://typeorm.io/active-record-data-mapper#what-is-the-active-record-pattern
[6] https://typeorm.io/active-record-data-mapper#what-is-the-active-record-pattern
[7] https://www.postgresql.org/
[8] https://code.visualstudio.com/

textos extensos, formatação de código fonte de acordo com a linguagem, interação com SGBD's que não fornecem uma interface amigável.

3.2.7 Oauth 2.0

A RFC 6749 define Oauth 2.0 como uma estrutura de autorização OAuth 2.0 permite que um terceiro aplicativo para obter acesso limitado a um serviço HTTP, seja em nome de um proprietário de recurso orquestrando uma interação de aprovação entre o proprietário do recurso e o serviço HTTP, ou permitindo que o aplicativo de terceiros para obter acesso em seu próprio nome (HARDT, 2012). Um ponto a ser observado é que a interação entre o usuário e API não acontecem diretamente, é necessário uma aplicação de terceiro para que o usuário possa interagir com API. Para que as informações possam fluir entre as aplicações com segurança algumas medidas necessárias são: não transitar os dados de usuário e senha em todas as requisições, garantir que somente usuários autorizados tenham acesso a recursos restritos, garantir que as informações do usuários que cheguem a ponta não foram alteradas no "meio". O protocolo Oauth 2.0 atende os padrões de segurança para troca de informações entre diferentes aplicações, uma demanda necessária para o desenvolvimento deste projeto é autenticar os usuários e fornecer recursos para cada usuário de acordo com a permissão de acesso definida. Segundo Eloy (2017) o protocolo Oauth 2.0 possuem quatro papeis definidos pela norma RFC 6749 que são: **Resource Owner**, **Cliente**, **Resource Server** e **Authorization Server**, a próxima figura representa os quatro componentes citados e o papel de cada um deles dentro do protocolo Oauth2.0.

Figura 6 – Papeis do Oauth 2.0

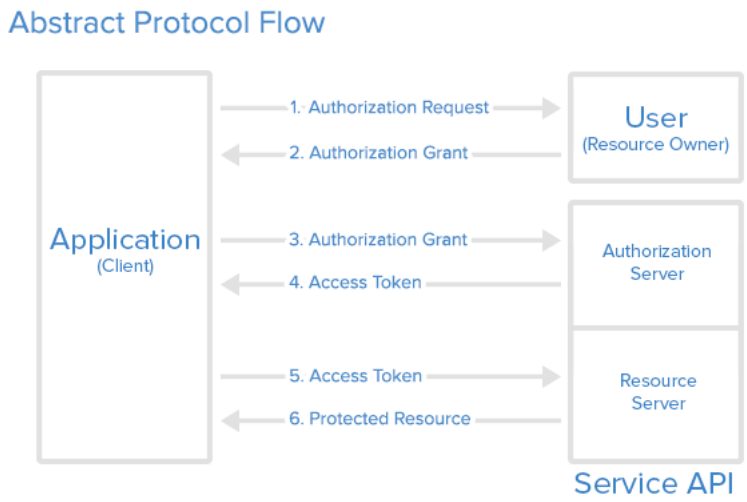


Fonte: Adaptado pelo autor (ELOY, 2017)

A figura 6 representa a comunicação entre os componentes do protocolo, o *Resource*

Owner é representado pelo usuário que utiliza a aplicação *Client* para interagir com a aplicação servidora. O componente *Resource Server* é responsável por validar se os recursos de usuário que o *Client* está tentando acessar possuem restrições e se podem ser acessados através do token informado pelo *Client*. O *Authorization Server* é responsável por registrar o *Client* e fornecer tokens de autorizações para o mesmo.

Figura 7 – Fluxo básico do Oauth 2.0



Fonte: (OCEAN, 2022)

Já a figura 7 representa a fluxo básico do protocolo. É importante observar que a aplicação *Client* precisa estar devidamente registrado na API. O fluxo é composto por três fases: autorização, solicitação do token de acesso e a utilização do token. Na fase de autorização o usuário definitivamente autoriza a aplicação *Client* a realizar determinadas operações em seu nome. Na segunda fase o *Client* solicita o token de acesso, nesse momento o *Client* deve enviar o *authorization code* recebido na fase de autorização. Por fim, a ultima etapa do fluxo utiliza o token de acesso para realizar operações no *Resource Server* em nome do *Resource Owner*.

3.2.8 JWT

Json Web Token [9] é um padrão aberto RFC 5519[10] que define uma forma compacta e autocontida para a transmissão segura de informações entre as partes como um objeto JSON (AUTH, 2022). A normativa RFC 5519 define o JWT como um meio compacto e seguro para URL de representar reivindicações a serem transferidas entre duas partes. As reivindicações em um JWT são codificados como um objeto JSON que é usado como

[9] https://jwt.io/
[10] https://www.rfc-editor.org/rfc/rfc7519

carga útil de um JSON Estrutura de assinatura da Web (JWS) ou como o texto simples de uma Web JSON Estrutura de criptografia (JWE).

3.3 Requisitos

O levantamento de requisito foi realizado por meio de entrevista com especialista de domínio[11] onde foi possível compreender o modelo de negócio. Como resultado da entrevista, foi possível determinar a lista de requisitos funcionais (tabela 2) e não funcionais que estão listados na (tabela 1)

Tabela 1 – Requisitos Funcionais

Requisitos Funcionais		
Codigo	Identificação	Descrição
RF01	Cadastros	Cadastrar solicitante
RF02	Cadastros	Cadastrar tecnico de campo
RF03	Cadastros	Cadastrar gestor
RF04	Segurança	Autenticação específica para usuarios solicitante
RF05	Segurança	Autenticação específica para demais usuários
RF06	Regra de negócio	Cadastrar tickets de atendimentos para novas demandas
RF07	Regra de negócio	Atribuir técnico de campo aos tickets e mudar o status "pendente"para o status "em atendimento"
RF08	Regra de negócio	Retornar status "em atendimento"para "pendente"os ticket que não puderam ser finalizados.
RF09	Regra de negócio	Fechar os tickets que foram concluídos e mudar o status de "em atendimento"para "concluído".
RF10	Regra de negócio	Permitir que o usuário solicitante avalie o atendimento atribuindo uma nota de 1 a 5.

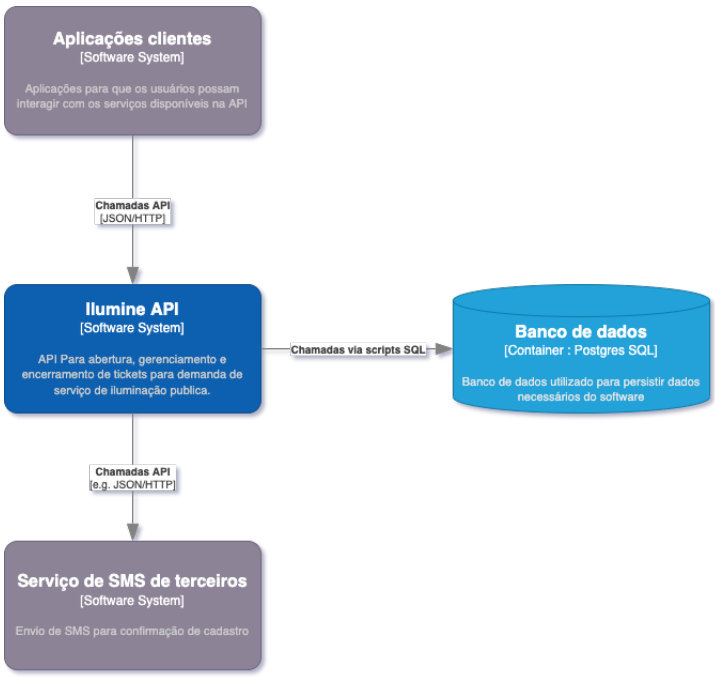
Tabela 2 – Requisitos Não Funcionais

Requisitos Não Funcionais	
Codigo	Descrição
RNF01	Possibilitar a extração de dados para que possam ser utilizados em plataformas de *Business Intelligence*
RNF02	Cadastrar e conceder ou revogar acesso de *client apps*

[11] Um especialista do domínio é uma pessoa que tem familiaridade com o domínio do negócio, mas não necessariamente com o desenvolvimento de sistemas de software. Freqüentemente, esses especialistas são os futuros usuários do sistema de software em desenvolvimento.

3.4 Arquitetura do software

Figura 8 – Arquitetura do Software



Fonte: elaborado pelo autor

Conforme pode ser observado na figura 8 a arquiterura do software está dividida em três componentes:

- Aplicações clientes: são aplicações que possuem interfaces amigáveis aos usuários. Elas são responsáveis por permitir que os serviços disponíveis na *Api* sejam acessíveis ao usuários por meio da suas funcionalidades. Podem ser desenvolvidas em plataforma Web, Mobile ou Desktop, idenpendente da liguagem de programação adotada, desde que a tecnologia escolhida permita a implementação do protocolo HTTP de acordo com a RFC7231[12]
- Ilumine API: Aplicação desenvolvida sobre o estilo de arquitetura REST. É responsável por processar os dados advindos das aplicações clientes por meio do protocolo HTTP estruturados em formato JSON[13] (JavaScript Object Notation). A API processa os dados de acordo com as regras de negócio implementada e persiste os dados válidos no banco de dados quando necessário.
- Serviço de SMS de terceiros: É um aplicação de terceiros desenvolvida em um modelo de API para fornecer o recurso de envio de SMS quando necessário.

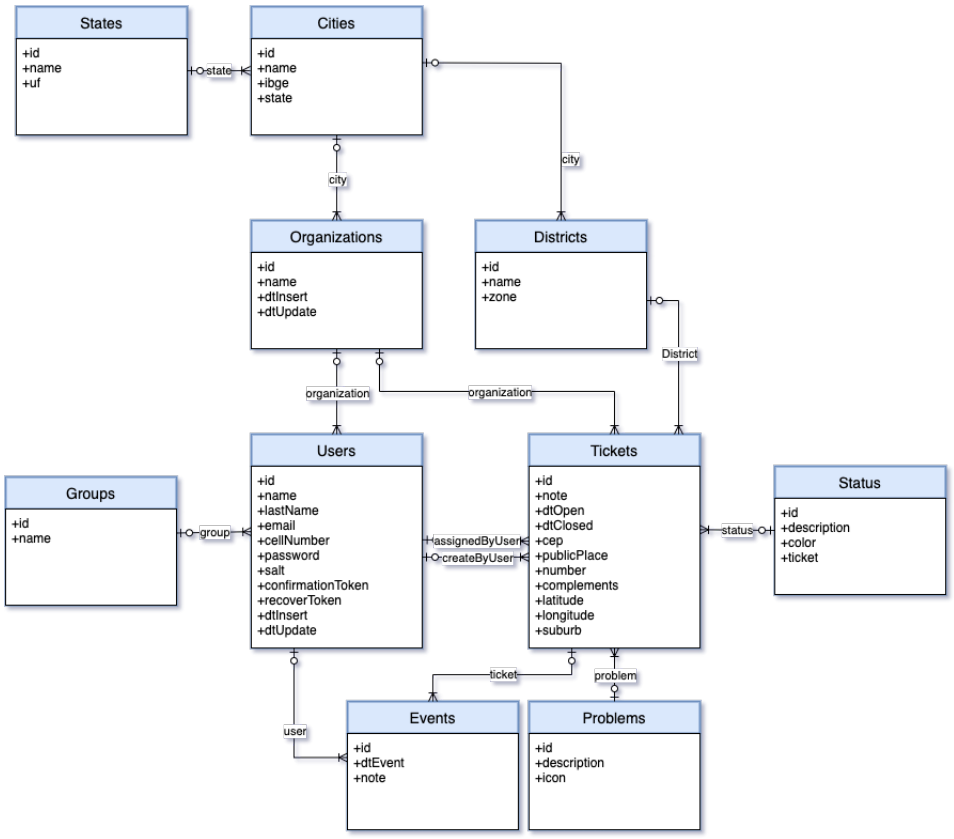
[12] Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - (https://datatracker.ietf.org/doc/html/rfc7231)
[13] https://www.json.org/json-en.html

3.5 Modelagem

3.5.1 DER - Diagrama de entidade e relacionamento

O modelo conceitual do SGBD foi concebido pelo uso da abordagem de Entidade e Relacionamento apresentado em diagrama na Figura 9 conforme descrito por (HEUSER, 1998)

Figura 9 – Diagrama de entidade e relacionamento

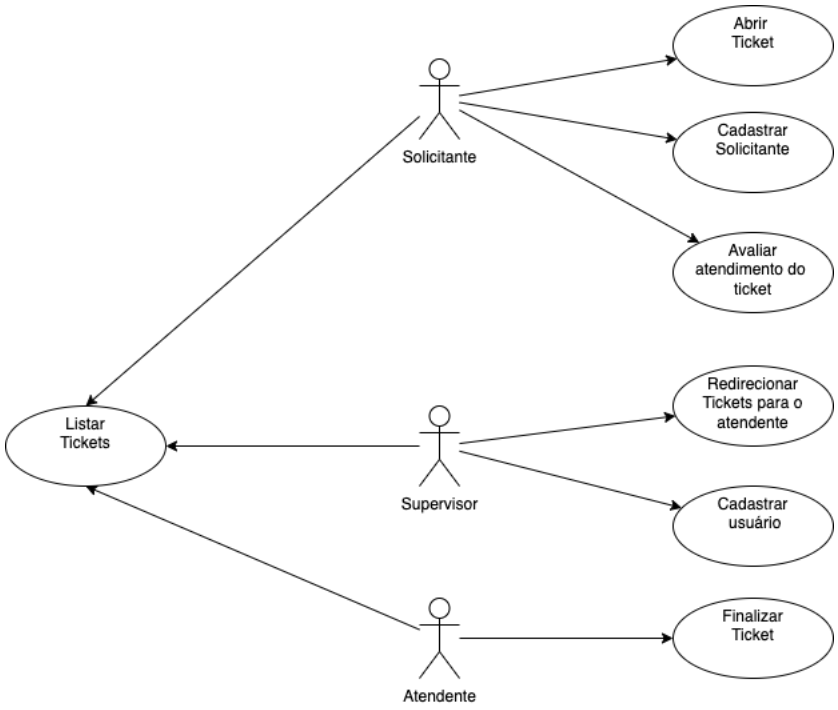


Fonte: elaborado pelo autor

3.5.2 DCU - Diagrama de caso de uso

Conforme (BEZERRA, 2007) o diagrama de caso de uso é utilizado para ilustrar em um alto nível de abstração os elementos externos que interagem com que funcionalidade do sistema. A figura 10 é o diagrama de caso de uso elaborado para o desenvolvimento da aplicação proposta, nela podemos identificar os atores, as funçionalidade da API e a relação entre eles.

Figura 10 – Diagrama de Caso de Uso

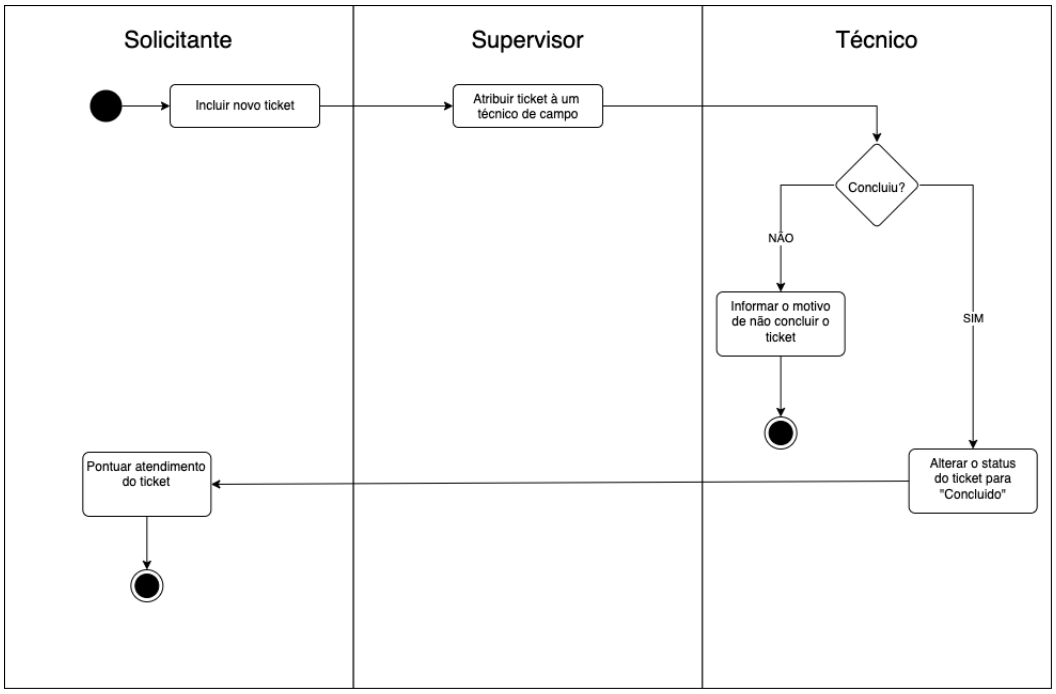


Fonte: elaborado pelo autor

3.5.3 Diagrama de atividade

Segundo (BEZERRA, 2007) o diagrama de atividade é um tipo especial de diagrama de estado, onde são representada o estado da atividade e não o estado do objeto. (BEZERRA, 2007) apresenta uma diferença importante entre diagrama de atividade e os demais diagramas de estado: o diagrama de atividade é orientado a fluxo de controle, já os diagramas de estados são orientados por eventos. A figura 11 ilustra o diagrama de atividades desenvolvido para esse projeto, nele pode-se observar todas as atividades envolvidas desde o cadastro até o fechamento de um ticket.

Figura 11 – Diagrama de atividade



Fonte: elaborado pelo autor

3.6 Licença de uso

Esse projeto é código aberto e é licenciado pelo MIT[14].

[14] Licença pode ser verificada no anexo.

4 Resultados e discussões

Neste capítulo são apresentados os resultados e discussões acerca da aplicação proposta.

4.1 Gerenciamento de configuração e mudanças

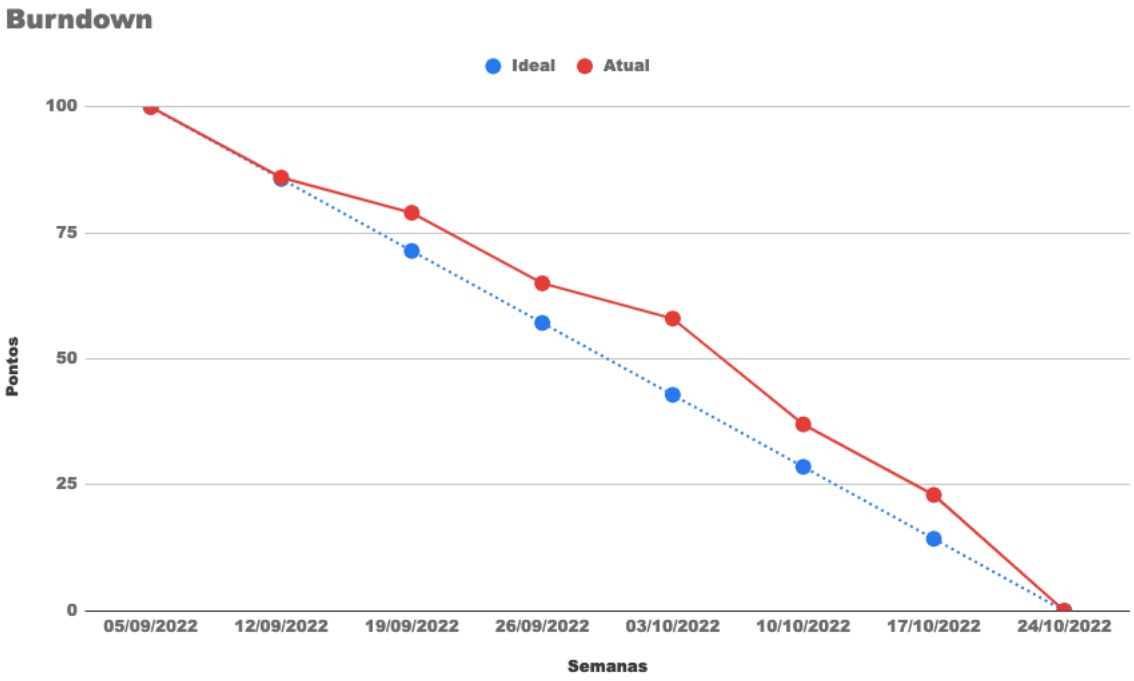
O gerenciador de repositório escolhido para hospedar o projeto foi o Gitlab[1].

4.2 Monitoramento de progresso

De acordo com o Guia do Scrum (SCHWABER, 2020) o monitoramento do progresso deve ser acompanhado pelo *Product Owner* a cada revisão da sprint e deve comparar o valor do trabalho apresentado com o valor do trabalho restante da *sprint review* para avaliar o progresso do projeto. A ferramenta visual utilizada para acompanhar o andamento das atividades desenvolvidas nesse trabalho e sugerida pelo Guia do Scrum (SCHWABER, 2020) é o gráfico *burndown* ilustrado na figura 12. A imagem representa o progresso das atividades realizadas semanalmente, de acordo com o gráfico, as atividades realizadas até o dia doze de setembro foram realizadas dentro do prazo, nas semanas subsequentes ocorreram atrasos que acumularam atividades até o dia três de outubro, nas semanas que seguiram, foram necessário medidas para aumentar a velocidade de entrega como dispor de um tempo maior de programação para tornar possível as entregas acordadas no prazo planejado.

[1] https://gitlab.com/willian.santos/ilumine-backend

Figura 12 – Gráfico *Burndown*

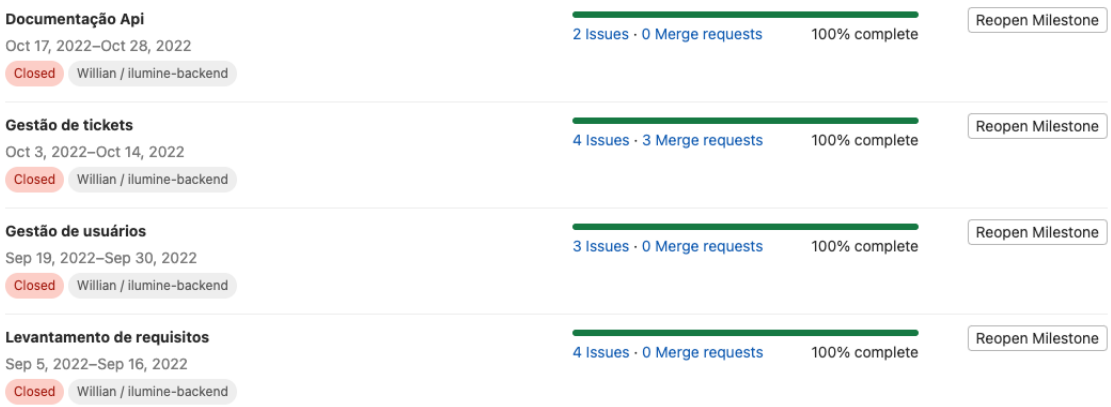


Fonte: elaborado pelo autor

4.3 Processo de desenvolvimento

O projeto foi desenvolvido utilizando a metodologia Scrum adaptado pelo fato de que alguns dos papéis do scrum como Scrum Master, Product Owner e o Time de Desenvolvimento foram exercídos pelo autor deste trabalho. O desenvolvimento do projeto foram divididos em quatro sprint de duas semanas conforme a figura 13.

Figura 13 – Sprints do projeto



Fonte: elaborado pelo autor

4.3.1 Sprint Levantamento de requisitos

Conforme figura 14 possui as tarefas essenciais para o inicio do desenvolvimento do projeto. Os diagramas desenvolvidos na sprint em questão foram utilizados em todas as sprints do projeto pelo time de desenvolvimeto como referência na retrospectiva de cada sprint para mensurar se todas as atividades desevolvidas até o momento estavam atendendo os objetivos do projeto.

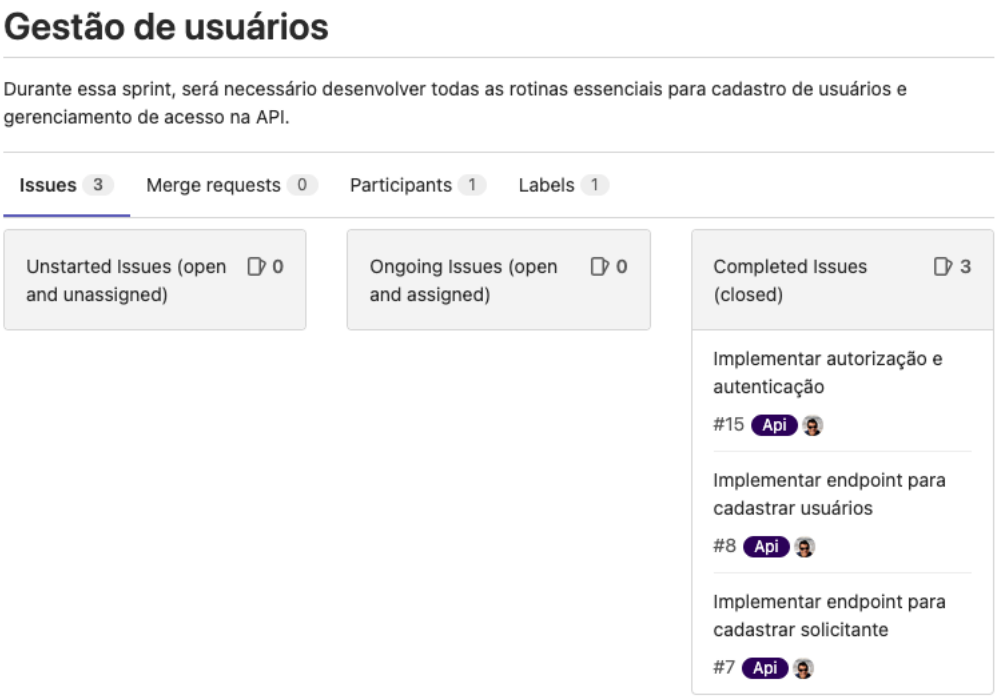
Figura 14 – Levantamento de requisitos

Fonte: elaborado pelo autor

4.3.2 Sprint Gestão de Usuários

A sprint Gestão de usuário marcou o início do desenvolvimento do código fonte do projeto, conforme a figura 15 nesse momento foram implementados as rotinas necessárias para cadastro de usuários e as rotinas de autenticação e regras de autorização necessárias para controlar o acesso por perfil de usuário para as rotinas que seriam implementadas posteriormente.

Figura 15 – Gestão de usuários



Fonte: elaborado pelo autor

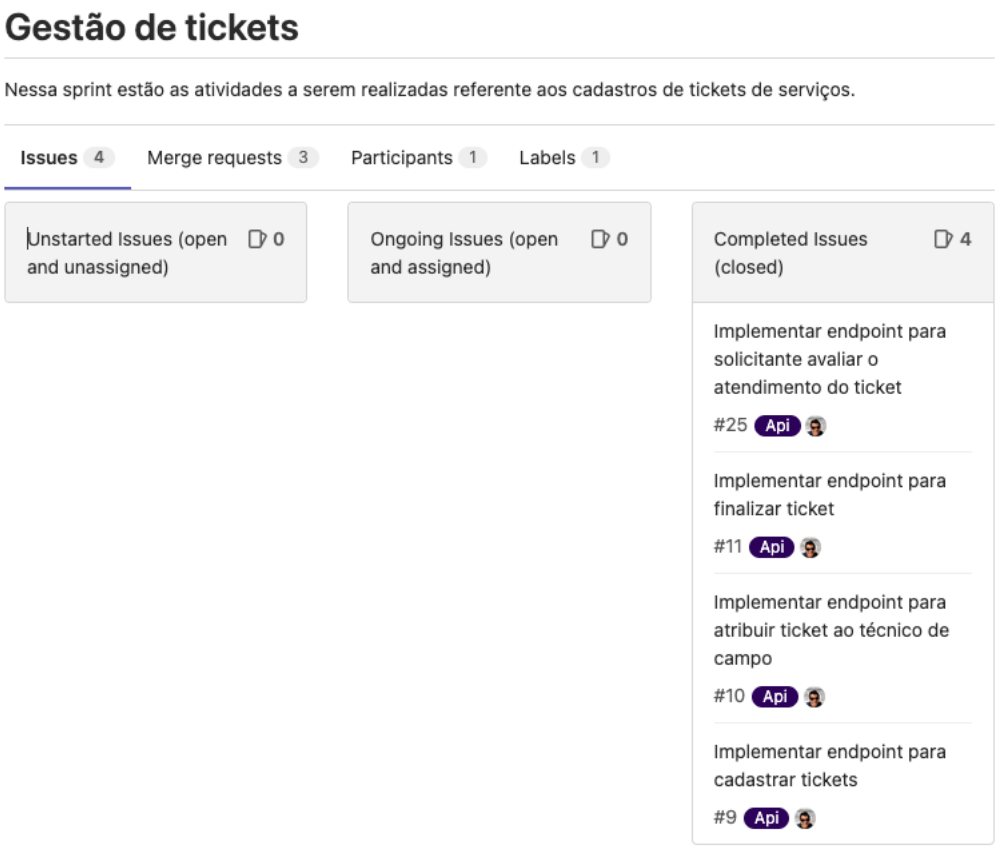
4.3.3 Sprint Gestão de tickets

Nessa fase do projeto conforme documentda na figura 16 foram implementadas as funcionalidade para atender o fluxo principal do projeto sendo abertura do ticket pelo solicitante, atribuição do ticket ao um técnico de campo, a finalização do ticket e a avaliação do ticket pelo solicitante.

4.3.4 Sprint Documentação API

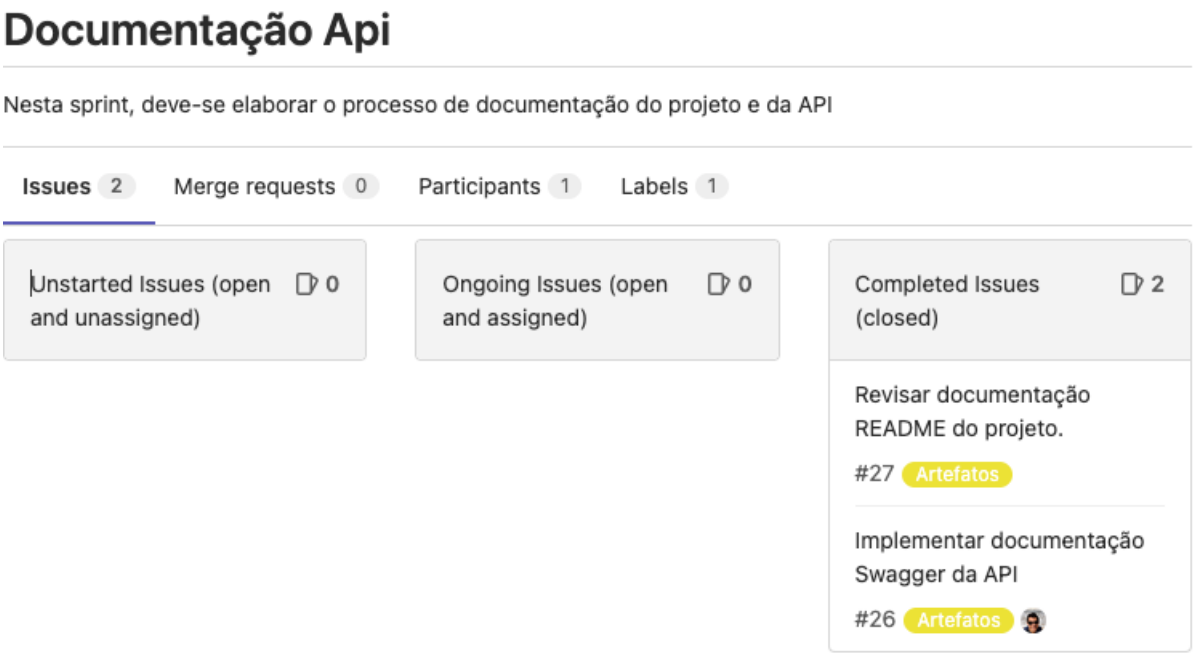
Após o desenvovimento das funcionalidades essencias do MVP, conforme demonstra a figura 17, nessa sprint foram elaborados os documentos da API e a documentação do projeto para facilitar o trabalho do desenvolvedor, tanto para o desenvolvimento do client para API quanto a documentação auxiliar para montar um ambiente de desenvolvimento funcional para futuras atualizações desse projeto.

Figura 16 – Gestão de tickets



Fonte: elaborado pelo autor

Figura 17 – Sprint Documentação API



Fonte: elaborado pelo autor

4.4 Testes automatizados

Segundo Paul e Jeff (2008) um dos objetivos do teste de software é automatizar o máximo possível, reduzindo significativamente seu custo, minimizando o erro humano. Com o uso do teste automatizado foi possível validar a integridade da aplicação após a implementação de novas funcionalidades. Para o desenvolvimento de teste automatizado desse trabalho foi utilizado a *framework* Jest [2].

Figura 18 – Teste automatizado

```
const mockUserRepository = () => ({
    createAdminUser: jest.fn(),
    createUser: jest.fn(),
    findUserById: jest.fn(),
    findUserByIdComplete: jest.fn(),
    getById: jest.fn(),
})

describe('UsersService', () =>{
    let userRepository;
    let service;

    beforeEach(async () => {
       const module: TestingModule = await Test.createTestingModule({
        providers: [
            UsersService,{
                provide: getRepositoryToken(UserRepository),
                useFactory: mockUserRepository,
            },{
                provide: getRepositoryToken(OrganizationRepository),
                useFactory: mockUserRepository,
            },{
                provide: getRepositoryToken(GroupsRepository),
                useFactory: mockUserRepository,
            }
        ]
       }).compile();

       userRepository = await module.get<UserRepository>(getRepositoryToken(UserRepository));
       service = await module.get<UsersService>(UsersService);
    });

    it('should be defined', () =>{
        expect(service).toBeDefined();
        expect(userRepository).toBeDefined();
    });
```

Fonte: elaborado pelo autor

[2] Disponível em: https://jestjs.io/

A figura 18 ilustra a implementação de um código fonte de teste automatizado. As funções utilizadas são descritas a seguir:

1. mockUserRepository: É utilizada para emular as chamadas das funcões dentro do objeto requerido pelo módulo UsersService.
2. describe: Função utilizada para agrupar os testes em blocos.
3. beforeEach: Função chamada antes de cada teste ser realizado, nesse caso é utilizada para inicializar os módulos e suas dependências.
4. it: Essa função é utilizada para descrever cada teste que será realizado, nesse caso verifica se os objetos foram instânciados corretamente.

A figura 19 ilustra o resultado da execução do teste. Na primeira linha a palavra *PASS* indica que o teste foi realizado com sucesso, a segunda linha mostra o bloco do teste descrito como *UserSevice*, a terceira linha pode-se verificar o nome do teste que foi executado, nesse caso *should be defined*.

Figura 19 – Teste automatizado

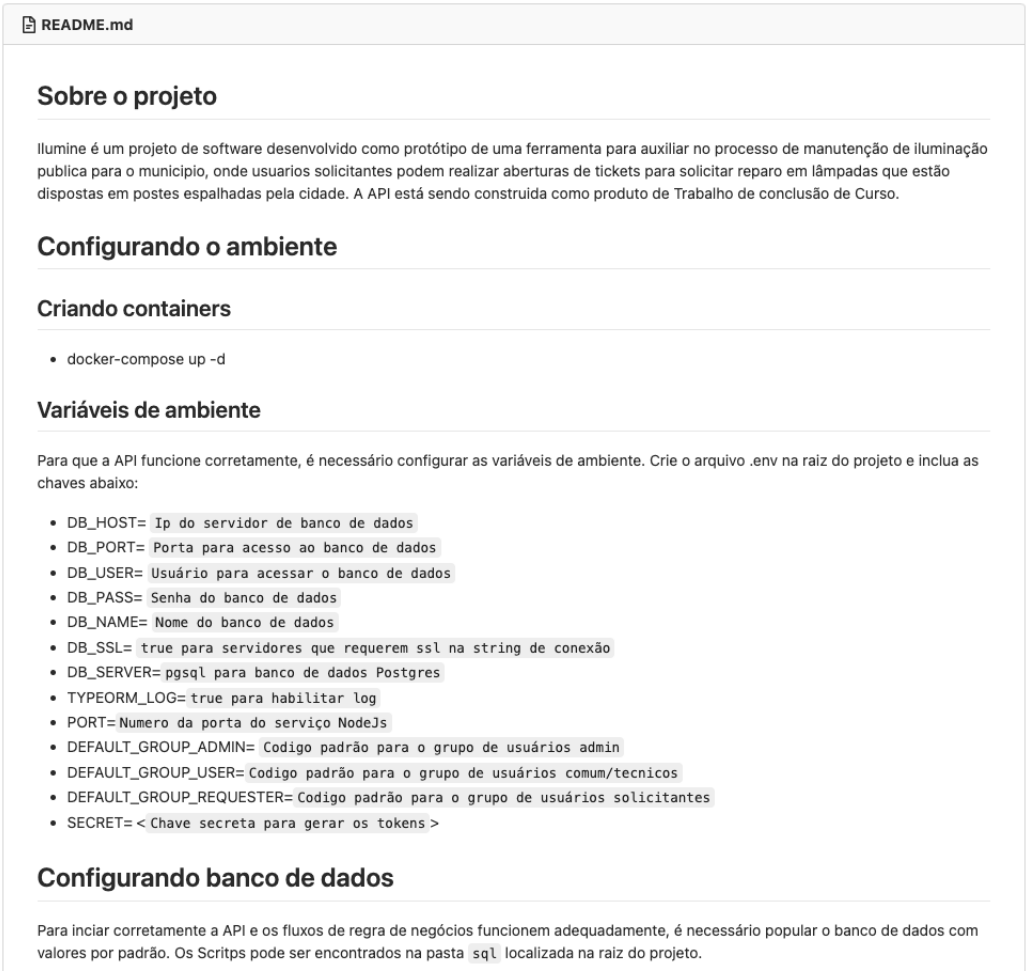
```
PASS src/users/users.service.spec.ts (5.687 s)
  UsersService
    √ should be defined (12 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.762 s
```

Fonte: elaborado pelo autor

4.5 Documentação

A documentação do projeto foi segmentada em dois artefatos, sendo um no README do repositório do projeto onde o desenvolvedor poderá fazer o dowload do código fonte e utilizar a documentação para montar o ambiente de desenvolvimento conforme a figura 20, a outra está documentanda no Swagger conforme a figura 21 que fica disponível durante a execução do projeto.

Figura 20 – Documentação do projeto



README.md

Sobre o projeto

Ilumine é um projeto de software desenvolvido como protótipo de uma ferramenta para auxiliar no processo de manutenção de iluminação publica para o municipio, onde usuarios solicitantes podem realizar aberturas de tickets para solicitar reparo em lâmpadas que estão dispostas em postes espalhadas pela cidade. A API está sendo construida como produto de Trabalho de conclusão de Curso.

Configurando o ambiente

Criando containers

- docker-compose up -d

Variáveis de ambiente

Para que a API funcione corretamente, é necessário configurar as variáveis de ambiente. Crie o arquivo .env na raiz do projeto e inclua as chaves abaixo:

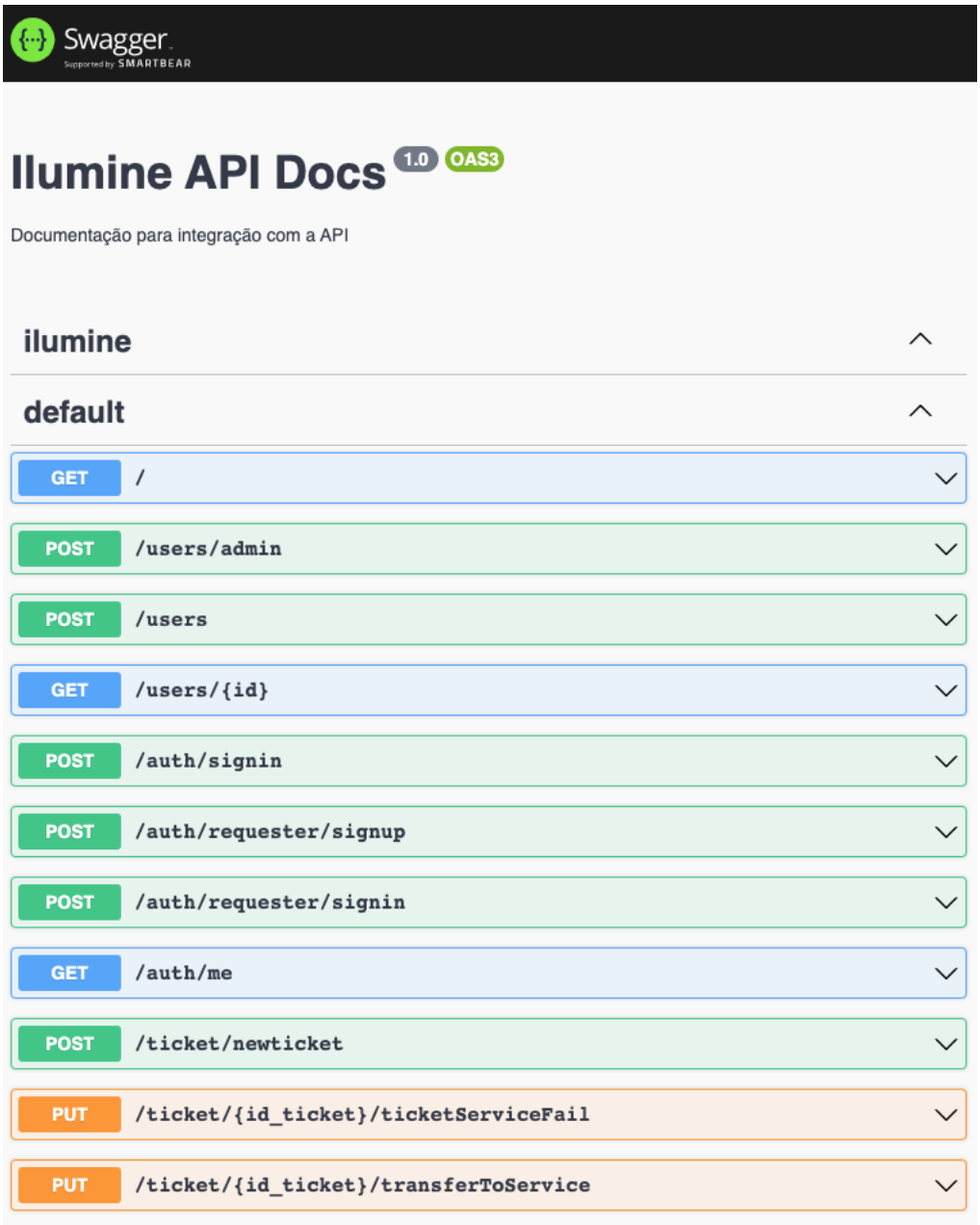
- DB_HOST= `Ip do servidor de banco de dados`
- DB_PORT= `Porta para acesso ao banco de dados`
- DB_USER= `Usuário para acessar o banco de dados`
- DB_PASS= `Senha do banco de dados`
- DB_NAME= `Nome do banco de dados`
- DB_SSL= `true para servidores que requerem ssl na string de conexão`
- DB_SERVER= `pgsql para banco de dados Postgres`
- TYPEORM_LOG= `true para habilitar log`
- PORT= `Numero da porta do serviço NodeJs`
- DEFAULT_GROUP_ADMIN= `Codigo padrão para o grupo de usuários admin`
- DEFAULT_GROUP_USER= `Codigo padrão para o grupo de usuários comum/tecnicos`
- DEFAULT_GROUP_REQUESTER= `Codigo padrão para o grupo de usuários solicitantes`
- SECRET= < `Chave secreta para gerar os tokens` >

Configurando banco de dados

Para inciar corretamente a API e os fluxos de regra de negócios funcionem adequadamente, é necessário popular o banco de dados com valores por padrão. Os Scritps pode ser encontrados na pasta `sql` localizada na raiz do projeto.

Fonte: elaborado pelo autor

Figura 21 – Documentação da API



Fonte: elaborado pelo autor

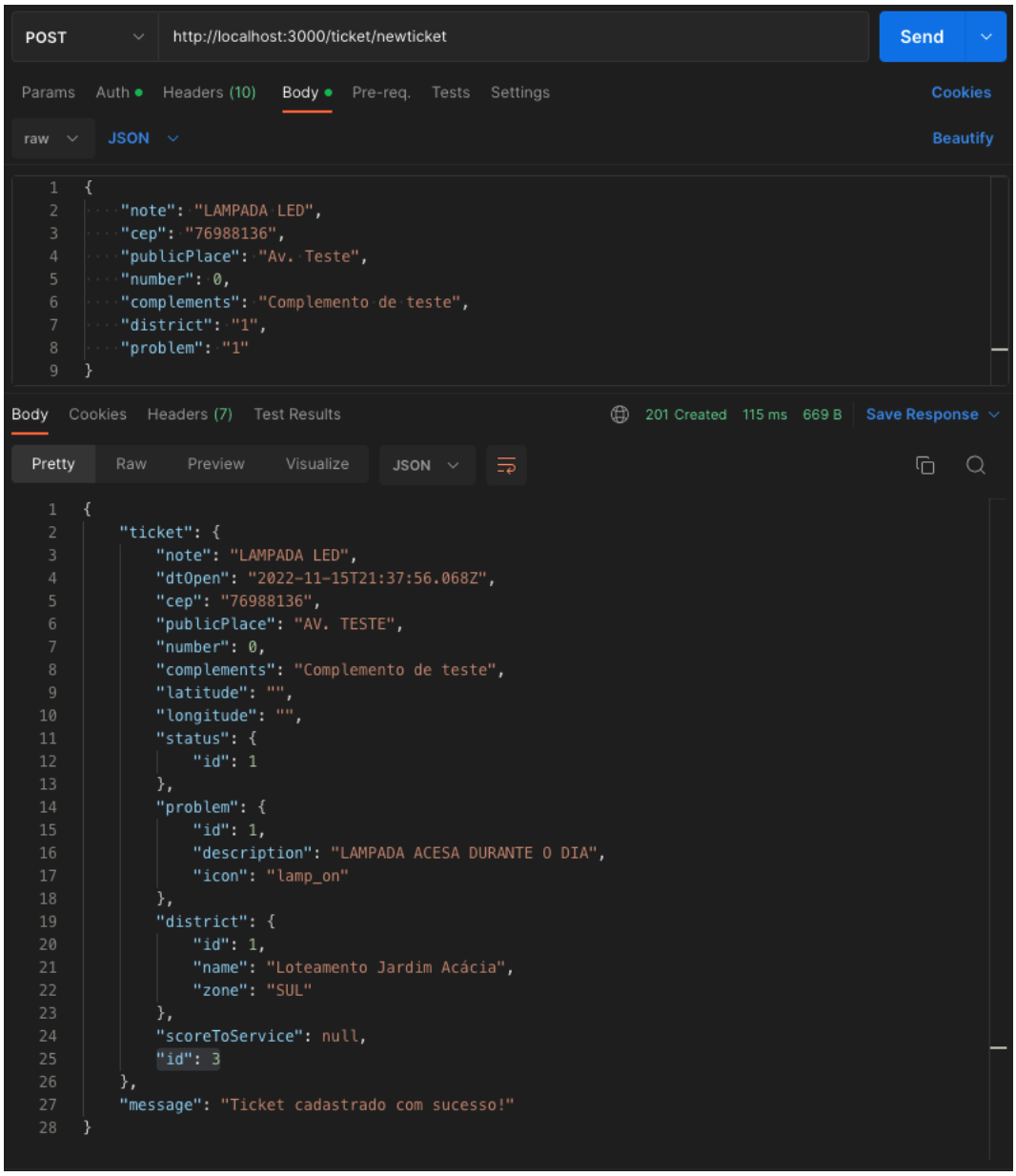
4.6 Demonstração do software

Nesse capítulo será demonstrada as principais funcionalidades da API.

4.6.1 Cadastrar novo ticket

Essa funcionalidade permite ao solicitante incluir uma nova demanda de manutenção na iluminação pública.

Figura 22 – Nova solicitação



Fonte: elaborado pelo autor

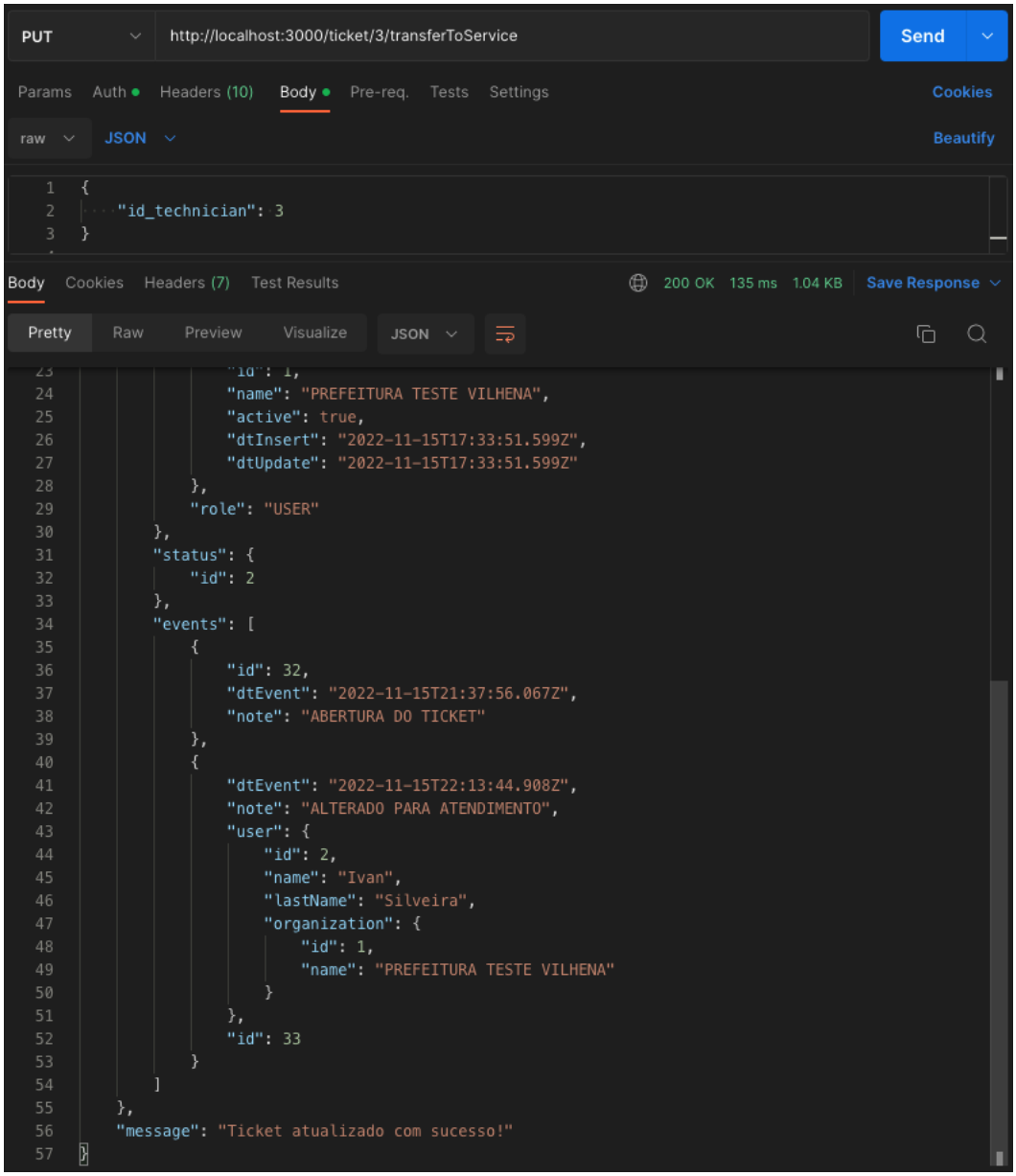
Conforme a figura 22 após o envio da requisição para solicitar a inclusão de uma

nova solicitação na API, a resposta da API (Quando a solicitação é processada com sucesso) é um json com informações detalhadas do novo ticket.

4.6.2 Enviar o ticket para atendimento

Essa funcionalidade permite ao supervisor atribuir um técnico para o ticket e alterar o status para atendimento conforme a figura 23.

Figura 23 – Enviar ticket para atendimento

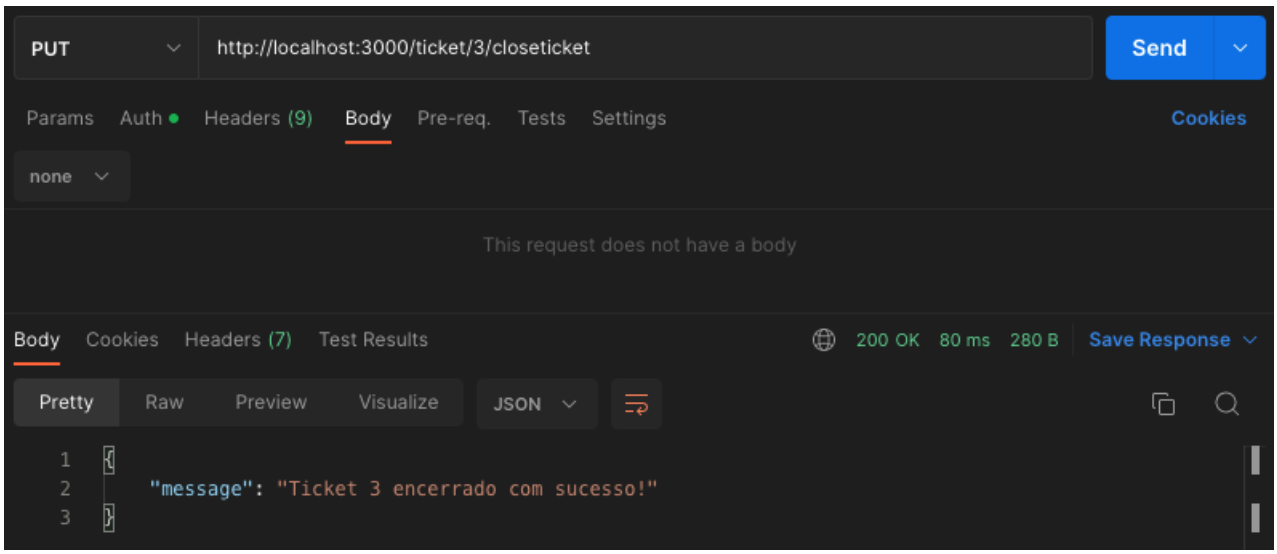


Fonte: elaborado pelo autor

4.6.3 Finalizar atendimento

Conforme a figura 24 essa funcionalidade é utilizada pelo técnico de campo para finalizar o atendimento do ticket e retirar da lista de serviços pendentes.

Figura 24 – Fechar ticket

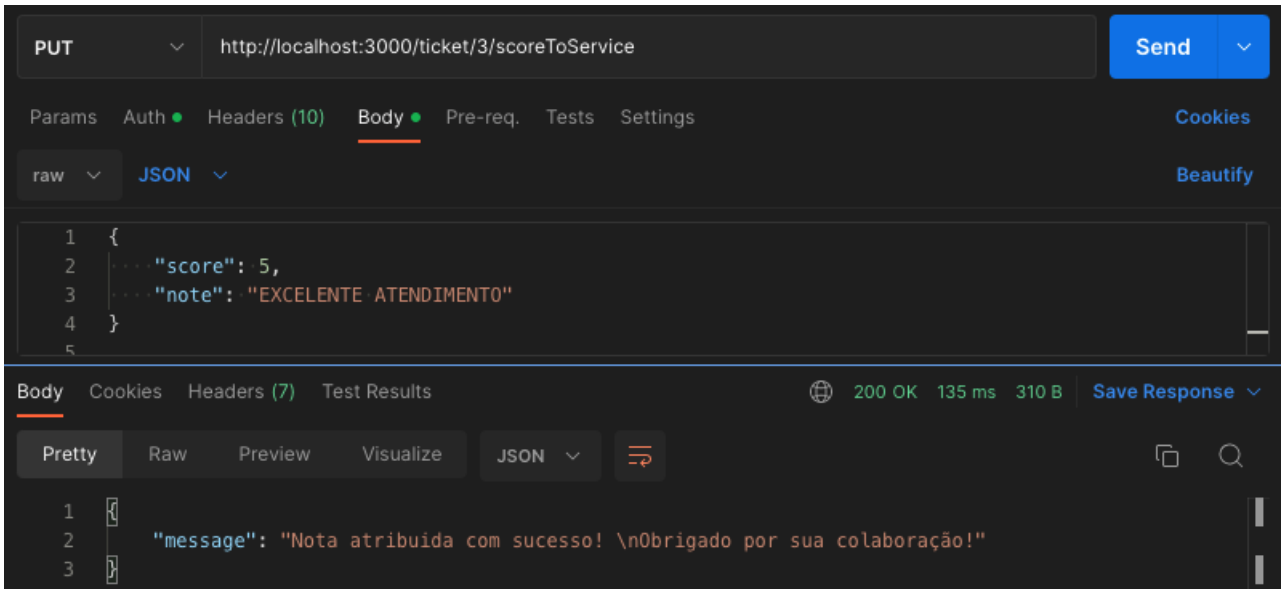


Fonte: elaborado pelo autor

4.6.4 Avaliando atendimento

Essa funcionalidade permite ao solicitante avaliar o atendimento atendimento do ticket, atribuindo uma nota de UM a CINCO para insatisfeito e muito satisfeito respectivamente. A figura 25 ilustra o exemplo de uma avaliação positiva do solicitante.

Figura 25 – Avaliar atendimento



Fonte: elaborado pelo autor

5 Considerações finais

Desenvolver o produto resultante desse trabalho é satisfatório pois foi carregado de desafios comumente encontrado na área de desenvolvimento de sistema onde o conhecimento relacionado a programação por si só não é suficiente, é importante conhecer o negócio, as regras os processos e em alguns casos até ajustar o processo para que a ferramenta seja eficiente colaborando com a performance do negócio. Aprender uma nova linguagem, construir scipts para banco de dados, aprender conceitos de padrões de projetos, testes e metodologias de desenvolvimento de softwares foram desafios superados para o desenvolvimento desse produto.

A API possibilita uma melhora na gestão do serviços de manutenção de iluminação pública, pois o banco de dados mantém históricos de todas as demandas abertas, atendidas, e concluídas, por usuário, por solicitante e por localidade. É possível disponibilizar aos solicitantes a lista de chamados abertos, atendidos e em atendimento, fazer avaliação do atendimento atribuindo uma nota e colocando uma observação.

5.1 Trabalhos futuros

Após o desenvolvimento da solução proposta foram identificados os seguintes trabalhos futuros podem ser realizados:

- Implementação de uma aplicação mobile para cadastro de solicitantes e abertura chamados.
- Implementação de uma aplicação front-end para os gestores.
- Implementação da funcionalidade de roteirização para os técnicos de campo
- Implementação de uma aplicação mobile para que os técnicos possam interagir com os gestores pela plataforma.

Referências

ALONSO, G. et al. Web services. In: *Web services*. [S.l.]: Springer, 2004. p. 123–149. Citado na página 16.

ANDERSON, D. J.; CARMICHAEL, A. *Essential kanban condensed*. [S.l.]: Blue Hole Press, 2016. Citado na página 17.

AUTH, I. Introduction to json web tokens. *HYPERLINK"https://jwt. io/introducti-on/"https://jwt. io/introduction/(visited on 15/10/2022)*, 2022. Citado na página 23.

BEZERRA, E. Princípios de análise e projeto de sistemas com uml. 2ª edição. *Campus-Elsevier. Rio de Janeiro*, 2007. Citado 2 vezes nas páginas 26 e 27.

BRASIL. Ato das disposições constitucionais transitórias. *HYPER-LINK"https://www.planalto.gov.br/ccivil_03/constituicao/Constituicao.htm#art30"(visited on 25/11/2022)*, 2002. Citado na página 13.

BRASIL. Emenda constitucional nº 39, de 19 de dezembro de 2002. *HYPER-LINK"https://www.planalto.gov.br/ccivil_03/constituicao/Constituicao.htm#art149a"(visited on 25/11/2022)*, 2002. Citado na página 13.

CORPORATION, G. E. History first power plant. *HYPER-LINK"http://www.globaledison.com/History_FirstPowerPlant.html"(visited on 15/10/2022)*, 2022. Citado na página 13.

ELOY, A. *OAuth 2.0: Proteja suas aplicações com o Spring Security OAuth2*. Casa do Código, 2017. ISBN 9788594188137. Disponível em: <https://books.google.com.br/books?id=iRM4DwAAQBAJ>. Citado na página 22.

FIELDING, R. et al. *RFC2616: Hypertext Transfer Protocol–HTTP/1.1*. [S.l.]: RFC Editor, 1999. Citado na página 15.

GRáFICA, L. T. E. A qualidade das luminárias para iluminação pública. *HYPERLINK"https://revistaadnormas.com.br/2020/08/11/a-qualidade-das-luminarias-para-iluminacao-publica"(visited on 15/10/2022)*, 2020. Citado na página 13.

HARDT, D. *The OAuth 2.0 Authorization Framework*. [S.l.]: RFC Editor, 2012. Citado na página 22.

HEUSER, C. A. *Projeto de Banco de Dados*. [S.l.]: Sagra Luzzato, 1998. Citado na página 26.

JURíDICO, R. C. Iluminação pública é competência do município, não da união. *HYPERLINK"https://www.conjur.com.br/2014-ago-03/iluminacao-publica-competencia-municipio-nao-uniao"(visited on 25/11/2022)*, 2014. Citado na página 13.

MARDAN, A. *Express. js Guide: The Comprehensive Book on Express. js*. [S.l.]: Azat Mardan, 2014. Citado na página 20.

MARRS, T. *JSON at work: practical data integration for the web*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado na página 16.

MILANI, A. *PostgreSQL-Guia do Programador*. [S.l.]: Novatec Editora, 2008. Citado na página 21.

OCEAN, L. D. An introduction to oauth 2. *HYPERLINK"https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2"https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2 (visited on 15/10/2022)*, 2022. Citado na página 23.

ORG, J. *Introducing JSON*. 2022. Urlhttps://www.json.org/json-en.html. Citado na página 16.

PAUL, A.; JEFF, O. *Introduction to software testing*. [S.l.]: Cambridge University Press, 2008. 1 p. Citado na página 35.

PEINADO, J.; AGUIAR, G. Compreendendo o kanban: um ensino interativo ilustrado. *Revista DaVinci. Curitiba (PR)*, v. 4, n. 1, p. 133–146, 2007. Citado 2 vezes nas páginas 16 e 17.

PEREIRA, C. R. *Construindo APIs REST com Node. js: Caio Ribeiro Pereira*. [S.l.]: Editora Casa do Código, 2016. Citado na página 20.

SAUDATE, A. *REST: Construa API's inteligentes de maneira simples*. [S.l.]: Editora Casa do Código, 2014. Citado na página 16.

SCHWAB, K. *A quarta revolução industrial*. [S.l.]: Edipro, 2019. Citado na página 16.

SCHWABER, J. S. K. *Scrum Guide*. [S.l.], 2020. Citado 2 vezes nas páginas 19 e 29.

Anexos

ANEXO A – Licença MIT

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.