INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS

JAHN CARLOS B. S. BRITO GUEDES
LUCAS MARINHO GOMES

IMPLEMENTAÇÃO DE API PARA CENTRALIZAR E GERENCIAR DADOS PESSOAIS POSSIBILITANDO O CONSUMO POR TERCEIROS

ARAGUAÍNA
2021

JAHN CARLOS BARBOSA SANTANA BRITO GUEDES
LUCAS MARINHO GOMES

IMPLEMENTAÇÃO DE API PARA CENTRALIZAR E GERENCIAR DADOS PESSOAIS POSSIBILITANDO O CONSUMO POR TERCEIROS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do Título de tecnólogo em Análise e desenvolvimento de sistemas do Curso Superior de Tecnologia do Instituto Federal do Tocantins, *Campus* Araguaína.

Orientador: Prof. Msc. José Roberto Cruz e Silva
Co-orientador: Prof. Msc. Mateus Dall'Agnol

ARAGUAÍNA
2021

Dados Internacionais de Catalogação na Publicação (CIP)
Bibliotecas do Instituto Federal do Tocantins

G633i Gomes, Lucas Marinho
IMPLEMENTAÇÃO DE API PARA CENTRALIZAR E GERENCIAR DADOS PESSOAIS POSSIBILITANDO O CONSUMO POR TERCEIROS / Lucas Marinho Gomes, Jahn Carlos Barbosa Santana Brito Guedes. – Araguaína, TO, 2021.
40 f. : il. color.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, Campus Araguaína, Araguaína, TO, 2021.

Orientador: Me. José Roberto Cruz e Silva
Coorientador: Me. Mateus Dall'agnol

1. API REST. 2. LGPD. 3. Java Script. I. Guedes, Jahn Carlos Barbosa Santana Brito. II. Cruz e Silva, José Roberto III. Dall'agnol, Mateus. IV. Título.

CDD 004.2

A reprodução total ou parcial, de qualquer forma ou por qualquer meio, deste documento é autorizada para fins de estudo e pesquisa, desde que citada a fonte.
Elaborado pelo sistema de geração automatica de ficha catalográfica do IFTO com os dados fornecidos pelo(a) autor(a).

FOLHA DE APROVAÇÃO

JAHN CARLOS BARBOSA SANTANA BRITO GUEDES

LUCAS MARINHO GOMES

IMPLEMENTAÇÃO DE API PARA CENTRALIZAR E GERENCIAR DADOS PESSOAIS POSSIBILITANDO O CONSUMO POR TERCEIROS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do Título de tecnólogo em Análise e desenvolvimento de sistemas do Curso Superior de Tecnologia do Instituto Federal do Tocantins, *Campus* Araguaína.

Data de aprovação: 12 / 01 / 2021

Banca Examinadora

Prof. Esp. Julio Aires Azevedo Leite, IFTO

Prof. Msc. Walisson Pereira de Sousa, IFTO

Prof. Msc. José Roberto Cruz e Silva, IFTO

Araguaína - TO, 2021.

AGRADECIMENTOS

Primeiro ao nosso Deus que nos deu essa oportunidade. Às nossas esposas Darllyane e Gabryella e demais familiares que sempre nos apoiaram.

RESUMO

A LGPD é uma lei aprovada recentemente que tem como objetivo proteger os dados pessoais dos clientes bem como sua privacidade. Até o ano de 2020, poucas empresas se adaptaram a esta nova lei. Este trabalho tem como objetivo criar uma ferramenta API que centralize e torne mais prático às empresas a consulta de dados pessoais dos clientes de acordo com a LGPD. Nesta API, foram utilizadas linguagens e ferramentas de programação bem conhecidas e consolidadas na área do desenvolvimento. Os resultados mostram que a API pode ser efetivamente utilizadas por terceiros para consultar dados pessoais.

Palavras-chaves: API REST, LGPD, JAVASCRIPT.

ABSTRACT

The LGPD is a recently passed law that aims to protect customers' personal data as well as their privacy. By the year 2020, few companies have adapted to this new law. This work aims to create an API tool that centralizes and makes it more practical for companies to consult their personal data according to the LGPD. In this API, well-known and consolidated programming languages and tools in the development area were used. The results show that the API can be effectively used by third parties to consult personal data.

Key Words: API REST, LGPD, JAVASCRIPT.

SUMÁRIO

1 INTRODUÇÃO.....15
1.1 Problema.....17
1.2 Objetivo Geral.....17
1.3 Objetivos específicos.....18
1.4 Organização do trabalho.....18
2 FUNDAMENTAÇÃO TEÓRICA.....20
2.1 API – Aplication Programming Interface.....20
2.2 JavaScript.....22
2.3 Node.js.....23
2.4 TypeScript.....24
2.5 Docker.....25
2.6 PostgreSQL.....26
2.7 Lei Geral de Proteção de Dados.....27
3 METODOLOGIA.....29
3.1 Metodologia de desenvolvimento da api.....29
4 RESULTADOS E DISCUSSÃO.....31
4.1 diagramas uml.....31
4.2 COMPANIES.....32
4.3 USERS.....34
4.4 address.....35
4.5 diagrama entidade-relacionamento.....36
5 CONCLUSÃO.....38
REFERÊNCIAS.....39

1 INTRODUÇÃO

Segundo Ribeiro (2020), o mercado da tecnologia vem crescendo muito rápido ultimamente, devido à enorme praticidade e redução de custos que o emprego da tecnologia pode oferecer. Segundo uma previsão da empresa de telecomunicações IDC Telecom, Ortiz (2020), o mercado brasileiro de tecnologia e comunicação crescerá 4,9% no ano de 2020.

A programação comercial é uma área dentro do mercado da tecnologia que permite desenvolver soluções para diversos problemas diferentes. Como exemplo, é possível citar o uso da programação para o contexto da logística, construção civil, saúde, prestação de serviços, etc.

Um dos desafios encontrados dentro da programação comercial é a necessidade de trocar informações com segurança, praticidade e rapidez. E, para garantir estas características, o homem vive buscando uma forma de aperfeiçoar suas ferramentas no que se refere a troca de dados.

De acordo com Giles (2010), a internet emergiu no contexto da Guerra Fria na década de 1960, a partir de um projeto do exército norte-americano. Os dois principais propósitos eram: criar um sistema de informação e comunicação em rede, que sobrevivesse a um ataque nuclear e dinamizar a troca de informações entre os centros de produção científica.

Figura 1 - Uso da internet tende a subir entre os brasileiros

Fonte: Roberta Jaworski, 2019

O avanço da Internet proporcionou diversas possibilidades de aplicações com o intuito de auxiliar as pessoas em todas as áreas da vida. Hoje inúmeras pessoas estão mais presentes no mundo virtual através de redes sociais, blogs e outros serviços. De acordo com matéria jornalística de Lavado (2019), o número de brasileiros que usam a internet subiu de 67% em 2017 para 70% em 2018, como observado na Figura 1, o que equivale a 126,9 milhões de pessoas.

Da mesma forma as empresas estão mais atuantes de forma virtual, pois perceberam o grande potencial da internet para a promoção de seus produtos, serviços e promoção de suas marcas (Santos, 2018).

Desta forma, este trabalho tem por objetivo apresentar um projeto que pode contribuir com a população ao levar comodidade e praticidade às pessoas. Este projeto envolve o desenvolvimento de um programa capaz de armazenar e disponibilizar dados pessoais, com o objetivo de dar praticidade e agilidade para o usuário ao precisar cadastrar ou atualizar suas informações pessoais em alguma

empresa.

Entretanto, as formas como o acesso às informações pessoais mudaram devido a uma nova lei implementada no ano de 2020 no Brasil. Esta nova lei entrou em visibilidade a partir de recorrentes casos de vazamento de dados pessoais, que se tornou frequente com o avanço da tecnologia.

A lei anteriormente mencionada é chamada de Lei Geral de Proteção de Dados (LGPD) e, conforme em matéria publicada pelo portal de notícias TechTudo (Bastos 2020), o objetivo da LGPD é:

> Regulamentar o uso de dados pessoais pelas empresas, de forma que os cidadãos brasileiros tenham mais segurança e controle sobre as suas informações. A vigência da LGPD estava prevista para agosto de 2020, mas algumas tramitações legais acabaram alterando os prazos iniciais.

Com base nestas informações, este trabalho se concentra no desenvolvimento de uma API REST [1]que é responsável por armazenar e disponibilizar dados pessoais, com o objetivo de dar praticidade e agilidade para o usuário ao precisar cadastrar ou atualizar suas informações pessoais em alguma empresa.

1.1 PROBLEMA

Para disponibilizar o melhor atendimento, dentre outras coisas, as empresas buscam obter informações de seus clientes, tais como: números de documentos pessoais e endereços. Mas essa situação leva a uma possível dificuldade para os usuários quando alguma dessas informações precisa ser alterada. No entanto, como os usuários são clientes de mais de uma empresa, é necessário um retrabalho para modificar as informações em todas as plataformas cadastradas. Desta forma, este trabalho busca responder a seguinte pergunta: Como a centralização de dados pessoais através de uma API REST pode colaborar para disponibilizar dados às empresas?

1 API REST: *Representational State Transfer*, em português Transferência Representacional de Estado, é um estilo de arquitetura de software que define um conjunto de restrições a serem usadas para a criação de web *services*.

1.2 OBJETIVO GERAL

Este trabalho tem como objetivo geral desenvolver uma API REST para centralizar os dados pessoais dos usuários em apenas uma plataforma, e então, disponibilizar os serviços de consulta às empresas.

1.3 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

1. Levantar artigo e trabalhos correlacionados a tecnologias de API;
2. Estudar os conceitos de API REST;
3. Estudar as ferramentas que serão utilizadas para o desenvolvimento da aplicação;
4. Codificar a API utilizando as ferramentas escolhidas e executar testes na aplicação;

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em seções, que abrangem assuntos relacionados às tecnologias que foram utilizadas para o desenvolvimento da API, Lei Geral de Proteção de Dados. Mais especificamente, este trabalho se divide em cinco seções, assim discriminados:

- Subseção 1: Refere-se à introdução do presente trabalho, destacando as vantagens de se utilizar uma ferramenta que disponibiliza seus dados pessoais facilmente para uso de empresas, respeitando as leis sobre acesso aos dados pessoais.
- Subseção 2: Apresenta uma revisão da literatura abordando conceitos básicos sobre API's, JavaScript, Node.js, TypeScript, Docker, PostgreSQL e Lei Geral sobre Proteção de Dados focando nas regras de negócios que envolvem a construção de uma API voltada para o armazenamento e compartilhamento de dados pessoais.
- Subseção 3: Apresentação da metodologia que será adotada neste trabalho para a implementação de uma API, assim como as

ferramentas utilizadas e as restrições adotadas devido à Lei de Proteção de Dados.

- Subseção 4: Apresentação dos resultados adquiridos após a implementação da API proposta. Nesta subseção, é demonstrado e explicado partes específicas do código e também, apresentado retornos obtidos das *endpoints* ao fazer uma consulta dentro da API.
- Subseção 5: Nesta subseção vai a conclusão. É a parte do texto onde são tiradas as conclusões do trabalho após a implementação da API. Ainda nesta subseção, é apresentada e explicada uma proposta de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho foi desenvolvido seguindo os conceitos das ferramentas que foram escolhidas para a implementação da API, que foram elas:

- JavaScript: a linguagem de programação escolhida para o desenvolvimento da API;
- Node.js: Framework que será utilizado para facilitar a codificação da aplicação;
- Typescript: o Superset escolhido;
- Docker: Plataforma para criação de ambientes isolados via container;
- PostgreSQL: o Sistema de Gerenciamento de Banco de Dados (SGBD) que será utilizado como a ferramenta específica para persistir dados da aplicação;

Nesta subseção, foi abordado os conceitos e funcionalidades de todas estas ferramentas citadas. Na subseção 2.1 foi explicado a história e os conceitos sobre as API's, na subseção 2.2 foi abordado conceitos acerca da linguagem JavaScript, já na subseção 2.3 o que foi abordado foram os conteúdos relacionados ao *framework* Node.js, o *superset* escolhido Typescript será mencionado na subseção 2.4, além da ferramenta Docker que também foi abordada na subseção 2.5 e, por último, o SGBD PostgreSQL foi abordado na subseção 2.6.

É também importante fazer uma explicação acerca da Lei Geral de Proteção de Dados (LGPD) que influenciará no projeto pois é essa lei quem dita as regras sobre as informações que podem e não podem ser acessadas pelas empresas. Foi feita a abordagem sobre esta lei na subseção 2.7.

2.1 API – APLICATION PROGRAMMING INTERFACE

A utilização de API's para o desenvolvimento de aplicações tem se tornado bastante expressiva, principalmente nos anos 2020. O motivo para a popularização das *Application Programming Interface* tem uma explicação simples: em vez de incorporar o produto de um fornecedor em uma aplicação, os desenvolvedores podem chamar uma API para consumir os serviços desse fornecedor. Os desenvolvedores não precisam saber o que está respondendo às chamadas no *backend*; eles só precisam saber o que a API do fornecedor espera do código produzido e vice-versa [CITATION Bri17 \l 1046].

As API's têm o propósito de simplificar o desenvolvimento de programas e aplicações ao abstrair os diversos elementos que compõem um *software*. Através deste conceito, o programador não precisa ater-se aos detalhes da implementação de todas as partes que compõem um sistema, ele precisando apenas saber como utilizar os serviços fornecidos pela API.

A API trata-se de um conjunto de requisições que permite a comunicação de dados entre aplicações. Para isso, a API utiliza requisições HTTP [2]responsáveis pelas operações básicas necessárias para a manipulação dos dados [CITATION SOU20 \l 1046]. As principais requisições são:

- *POST*: criar dados no servidor;
- *GET*: leitura de dados no host;
- *DELETE*: excluir as informações;
- *PUT*: atualizações de registros;

A Figura 2 abaixo apresenta uma breve explicação sobre a responsabilidade de cada requisição da API REST.

Figura 2 - Requisições API REST

Métodos	**Responsabilidade**
GET	Buscar dados
POST	Salvar os dados
PUT	Substitui determinado dado
DELETE	Apaga determinado dado

Fonte: Gabriel Cardoso, 2019

Um dos padrões de arquitetura presente no conceito da API é o REST: *Representational State Transfer*. Este padrão é um conjunto de restrições utilizadas para que as requisições HTTP atendam às diretrizes definidas na arquitetura. Basicamente, as restrições determinadas pela arquitetura REST são:

- Cliente-servidor: as aplicações existentes no servidor e no cliente devem ser separadas;
- Sem estado: as requisições são feitas de forma independente, ou seja, cada uma executa apenas uma determinada ação;

2 HTTP é a sigla de HyperText Transfer Protocol

- *Cache*[3]: a API deve utilizar o cache para evitar chamadas recorrentes ao servidor;
- Interface uniforme: agrupa outros quatro conceitos em que determina que os recursos devem ser identificados, a manipulação dos recursos deve ser por meio de representação, com mensagens auto descritivas e utilizar *links* para navegar pelo aplicativo.

2.2 JAVASCRIPT

JavaScript é uma linguagem de programação *web*. A grande maioria dos sites modernos utiliza JavaScript e todos os navegadores modernos incluem interpretadores JavaScript, tornando-a a linguagem de programação mais ubíqua da história. O JavaScript faz parte das 3 linguagens que todos desenvolvedores *web* devem conhecer, juntamente com HTML e CSS [CITATION Dav13 \l 1046].

Inicialmente, o JavaScript foi criado para tornar as páginas da web mais dinâmicas. Os programas nesta linguagem são chamados de *scripts*. Eles podem ser escritos diretamente no HTML de uma página da web e executados automaticamente conforme a página carrega.

Ao contrário da linguagem Java, na qual é erroneamente associada pelos programadores iniciantes, os scripts JavaScript são interpretados e não precisam de uma preparação ou compilação especial para serem executados. O JavaScript pode ser executado não apenas no navegador, mas também no servidor ou, na verdade, em qualquer dispositivo que tenha um programa especial chamado mecanismo JavaScript [CITATION ILY19 \l 1046].

A linguagem JavaScript básica define uma API mínima para trabalhar com texto, *arrays*, datas e expressões regulares, mas não inclui funcionalidade alguma de entrada ou saída. Entrada e saída são responsabilidade do "ambiente hospedeiro" dentro do qual o JavaScript está incorporado. Normalmente, este ambiente hospedeiro é um navegador web[CITATION Dav13 \l 1046].

Não há dúvida de que o ecossistema JavaScript muda rapidamente. Não apenas novas ferramentas e *frameworks* são introduzidos e desenvolvidos em um ritmo rápido, a própria linguagem passou por grandes mudanças com a introdução

3 Área de memória onde é mantida uma cópia temporária de dados armazenados em um meio de acesso mais lento, com o objetivo de acelerar a recuperação dos dados.

do ES2015 [4](também conhecido como ES6). Compreensivelmente, muitos artigos foram escritos reclamando sobre como é difícil aprender a desenvolver com JavaScript moderno nos dias de hoje [CITATION BUC18 \l 1046].

Apesar de o JavaScript normalmente ser utilizado para a plataforma web, esta linguagem pode ser utilizada para o desenvolvimento de aplicações para *desktop*. Através de *frameworks* como ElectronJS, Node.js entre outros. Na próxima subseção, será aborda o framework Node.js que foi escolhido para a implementação da API.

2.3 NODE.JS

Ao longo dos anos, vários *frameworks* de JavaScript foram lançados, entre eles o Node.js. Este *framework* é um dos mais utilizados pelo fato de entregar ao desenvolvedor um ótimo desempenho e alta escalabilidade, ou seja, utilização sob demanda.

O Node.js possui o código aberto e é um ambiente de execução multiplataforma usado para o desenvolvimento de aplicativos web do lado do servidor. Os aplicativos Node.js são escritos em JavaScript e podem ser executados em uma grande variedade de sistemas operacionais[CITATION KRI16 \l 1046].

Ainda segundo Rungta (2016), Node.js possui uma arquitetura orientada a eventos e uma API de I/O (Input/Output) sem bloqueio que é projetado para otimizar o rendimento de um aplicativo e escalabilidade em tempo real.

As vantagens de se usar o Node.js são amplas. No que se refere a flexibilidade, o Node.js possui um gerenciador de pacotes chamado NPM (*Node Package Manager*), que se trata de um dos maiores repositórios de *softwares* do mundo. Isso faz do Node.js uma plataforma com potencial para ser utilizada em qualquer situação.

Já em relação ao desempenho, o Node.js é um dos *softwares* mais indicados. A tarefa de criar um ambiente Node.js e colocar em produção uma aplicação não exige muitos recursos computacionais em comparação com outras tecnologias mais tradicionais. Se utilizado juntamente com ferramentas como o Docker, o ganho na velocidade de *deploy* e replicação de máquinas pode ser muito significativo e em ambientes escaláveis isso significa menor custo e mais eficiência [CITATION

4 ECMAScript é uma linguagem de programação, baseada em scripts, padronizada pela Ecma International na especificação ECMA-262 e ISO/IEC 16262

LEN18 \l 1046].

Através de benefícios como a leveza e a flexibilidade citados anteriormente, o Node.js tornou-se uma tecnologia apropriada para implementação de serviços e componentes de arquiteturas como a de microsserviços e *serveless* (sem necessidade de utilizar servidor). Além de tudo, possui o suporte das principais empresas de produtos e serviços em nuvem, como AWS, Google Cloud e Microsoft Azure que oferecem, na maioria de seus produtos, suporte nativo [CITATION LEN18 \l 1046].

2.4 TYPESCRIPT

O TypeScript é um *superset* da linguagem JavaScript criada pela Microsoft. Ou seja, todo código JavaScript também é um código TypeScript. O TypeScript adiciona funcionalidades que não estão disponíveis nativamente ao JavaScript ou requerem grande esforço para utilização, como tipagem de dados e orientação a objetos. Segundo Maharry (2013), o TypeScript possui basicamente três componentes:

- O compilador TypeScript: escrito em TypeScript, o compilador recompila o TypeScript em JavaScript idiomático.
- O TypeScript Language Service (TLS): O TLS é uma extensão do Visual Studio. É essa extensão que fornece o IntelliSense para TypeScript, reforçando a segurança de tipo e vários recursos do TypeScript no Visual Studio.
- Arquivos de declaração: O TLS usa arquivos de declaração para fornecer IntelliSense para os tipos, chamadas de função e variáveis esperadas por bibliotecas de terceiros, como jQuery, Prototype, MooTools e assim por diante.

O TypeScript é uma alternativa para quem deseja utilizar JavaScript para desenvolver suas aplicações mas quer ter acesso a alguns recursos que o JS não oferece. Em um livro publicado pela Microsoft (2019), estes recursos do TypeScript são enumerados:

- O Typescript adota os blocos de construção básicos do seu programa a partir do JavaScript. Portanto, você só precisa saber JavaScript para usar o

TypeScript. Todo o código TypeScript é convertido em seu equivalente em JavaScript para fins de execução.

- TypeScript oferece suporte a outras bibliotecas JS. O TypeScript compilado pode ser consumido de qualquer código JavaScript. JavaScript gerado por TypeScript pode reutilizar todas as estruturas, ferramentas e bibliotecas JavaScript existentes.
- JavaScript é TypeScript. Isso significa que qualquer arquivo .js válido pode ser renomeado para .ts e compilado com outros arquivos TypeScript.
- TypeScript é portátil. O TypeScript é portátil em navegadores, dispositivos e sistemas operacionais. Ele pode ser executado em qualquer ambiente em que o JavaScript seja executado. Ao contrário de suas contrapartes, o TypeScript não precisa de uma máquina virtual dedicada ou de um ambiente de execução específico para ser executado.

De forma geral, o TypeScript se sai como uma excelente alternativa para o desenvolvimento de aplicações JavaScript de forma mais fácil devido uma série de recursos adicionais. Além disso, o TypeScript é superior a suas outras concorrentes, como as linguagens de programação CoffeeScript e Dart, de maneira que o TypeScript é uma extensão do JavaScript.

2.5 DOCKER

O Docker é uma ferramenta DevOps frequentemente utilizada por programadores criarem ambientes apropriados para desenvolvimento ao compreender todo o necessário para executar o software de forma independente. Através da virtualização a nível de sistema operacional, o Docker é capaz de entregar *softwares* em pacotes chamados *containers*. Cada contêiner agrega seus próprios *softwares*, bibliotecas e arquivos de configuração e são isolados uns dos outros, bem como da máquina hospedeira.

Uma definição mais exata é dada por Jeeva et. al. (2015):

> O Docker é um mecanismo de contêiner de código aberto, que automatiza o empacotamento, o envio e a implantação de qualquer aplicativo de *software* apresentado como contêiner leve, portátil e autossuficiente, que será executado em praticamente qualquer lugar.

O Docker é uma ferramenta de fácil utilização para pequenas, médias e

pequenas equipes de desenvolvimento, pois através de seu funcionamento por containers, todos os aplicativos que a compõem parecem essencialmente os mesmos além das perspectivas dos sistemas de hospedagem.

Alguns dos benefícios do Docker são listados por Matthias & Kane (2015):

- Empacotar o *software* de forma que aproveite as habilidades que os desenvolvedores já possuem.
- Uma única imagem padronizada contém todos os pacotes de softwares e sistemas de arquivos do SO necessários para executar um ambiente.
- Utilização de artefatos empacotados para testar e entregar exatamente o mesmo artefato para todos os sistemas em todos os ambientes.
- Maior aproveitamento de recursos de *hardware* através da eliminação dos *hypervisors* quando se utilizam as soluções tradicionais de virtualização.

O Docker é uma das ferramentas mais indicadas quando se trata de criar um ambiente que ofereça todo o aparato para codificar, testar e implantar um *software*. Esta ferramenta é importante para o desenvolvimento da API pois ela é capaz de oferecer toda a infraestrutura necessária para o desenvolvimento do projeto.

2.6 POSTGRESQL

O PostgreSQL é atualmente um dos Sistemas de Gerenciamento de Banco de Dados (SGBD) objeto relacional mais populares do mundo. É bastante avançado no que se refere a recursos, além de possuir o código aberto. Dentro da própria documentação do Postgresql.org,2021, que pode ser encontrada no *website* oficial, aqui estão alguns dos recursos disponíveis em uma distribuição Postgres padrão:

- Consultas complexas.
- Chaves estrangeiras.
- Integridade transacional.
- Controle de concorrência multiversão.
- Suporte ao modelo híbrido objeto-relacional.
- Facilidade de Acesso.
- Visões.
- Linguagem Procedural em várias linguagens (PL/pgSQL, PL/Python, PL/Java,

PL/Perl) para Procedimentos armazenados.

- Indexação por texto.
- Estrutura para guardar dados Georreferenciados. PostGIS.

Desta forma, o PostgreSQL é uma excelente escolha no que tange a um servidor de banco de dados para persistir os dados da API a ser desenvolvida. Além dos benefícios já citados anteriormente, ainda possui um ótimo suporte pela comunidade de desenvolvedores.

2.7 LEI GERAL DE PROTEÇÃO DE DADOS

Aprovada em agosto de 2018, a Lei Geral de Proteção de Dados visa proteger os dados pessoais dos clientes bem como sua privacidade. De acordo com o governo brasiliero a lei está assim definida (Brasil,2020):

> Esta Lei dispõe sobre o tratamento de dados pessoais, inclusive nos meios digitais, por pessoa natural ou por pessoa jurídica de direito público ou privado, com o objetivo de proteger os direitos fundamentais de liberdade e de privacidade e o livre desenvolvimento da personalidade da pessoa natural.

Esta lei surgiu após uma polêmica envolvendo os escândalos de privacidade do Facebook, em que uma empresa chamada Cambridge Analytica utilizou dados dos usuários para que pudessem fazer uma campanha política mais assertiva e customizada na eleição de Donald Trump em 2016 [CITATION G1N19 \l 1046].

Após este episódio, a preocupação com a segurança de dados bem como a privacidade têm sido colocados cada vez mais em evidência no Brasil. Além deste caso que aconteceu nos Estados Unidos, há também casos que aconteceram no Brasil. Em 2018, por exemplo, o MPF notificou a Microsoft para que Windows 10 pare de coletar dados pessoais sem autorização dos usuários. Neste caso, a opção padrão de instalação do Sistema Operacional habilitava a transferência de informações sem alertar consumidores. Este procedimento viola a legislação e representa riscos para a privacidade [CITATION ANA18 \l 1046].

A Lei Geral de Proteção de Dados Pessoais (LGPD) prevê os requisitos necessários para o tratamento de dados pessoais. São eles [CITATION Bra18 \l 1046]:

- Mediante o fornecimento de consentimento pelo titular;
- Para o cumprimento de obrigação legal ou regulatória pelo controlador;
- Pela administração pública, para o tratamento e uso compartilhado de dados

necessários à execução de políticas públicas previstas em leis e regulamentos ou respaldadas em contratos, convênios ou instrumentos congêneres;

- Para a realização de estudos por órgão de pesquisa, garantida, sempre que possível, a anonimização dos dados pessoais;
- Quando necessário para a execução de contrato ou de procedimentos preliminares relacionados a contrato do qual seja parte o titular, a pedido do titular dos dados;
- Para o exercício regular de direitos em processo judicial, administrativo ou arbitral, esse último nos termos da Lei nº 9.307, de 23 de setembro de 1996 (Lei de Arbitragem);
- Para a proteção da vida ou da incolumidade física do titular ou de terceiros;
- Para a tutela da saúde, exclusivamente, em procedimento realizado por profissionais de saúde, serviços de saúde ou autoridade sanitária;
- Quando necessário para atender aos interesses legítimos do controlador ou de terceiros, exceto no caso de prevalecerem direitos e liberdades fundamentais do titular que exijam a proteção dos dados pessoais; ou
- Para a proteção do crédito, inclusive quanto ao disposto na legislação pertinente.

Estes requisitos foram fundamentais no momento em que será elaborado as regras de negócio da API a ser desenvolvida. Cada item exigido na Lei foi levado em conta no comportamento da solução desenvolvida para evitar transtorno e descumprimento da Lei por quem irá utilizar a solução.

3 METODOLOGIA

Este trabalho foi realizado utilizando-se de propósito exploratório, uma vez que o assunto abordado é pouco conhecido ou explorado, visto que a LGPD é uma lei bem recente e que ainda possuem poucos sistemas adaptados a esta nova lei. Ademais, em relação a abordagem da pesquisa, é utilizada uma abordagem qualitativa, dado que é pretendida a construção de uma API que torne possível a centralização de dados pessoais de usuários, para então, estar disponível para consulta por outras empresas.

Em relação à natureza, este trabalho trata-se de uma pesquisa aplicada, uma vez que existe esforço para criar novos conhecimentos para aplicação prática, orientada à solução de um problema específico. Os procedimentos e técnicas utilizadas são baseadas em pesquisas documentais de assuntos relacionados ao tema proposto. A partir destas pesquisas documentais realizadas em artigos e jornais eletrônicos foi possível entender melhor o contexto no qual o trabalho está inserido e qual a melhor forma de construção da API proposta.

Para melhor explicar a metodologia aplicada para a implementação da API, foram desenvolvidos 2 diagramas que, além de ajudar a explicar a construção da API, também podem ser utilizados como base da implementação. Estes diagramas serão apresentados na subseção 1.1, além dos principais materiais que ser serão usados para o desenvolvimento deste trabalho.

1.1 METODOLOGIA DE DESENVOLVIMENTO DA API

Existe uma enorme quantidade de tecnologias e ferramentas utilizadas atualmente para implementação de API's, cada uma com suas peculiaridades, vantagens e desvantagens. A escolha no conjunto de ferramentas a ser utilizada para o desenvolvimento desta foi baseada na familiaridade com tais ferramentas, além da simplicidade e maior suporte da comunidade frente às concorrentes.

Para realizar a construção da API foi elaborado um diagrama de classes e um diagrama de casos de uso que pode ser observado na seção 4.1 que foram utilizados para orientar a implementação. Os diagramas UML são utilizados para estabelecer uma linguagem visual comum no complexo mundo do desenvolvimento de software, que também poderia ser compreendida por usuários do mundo dos

negócios e qualquer pessoa que queira entender mais sobre um sistema.

Os materiais utilizados para realizar os testes do algoritmo desenvolvido são basicamente 1(um) computador de especificações bastante simples. Além disso, o algoritmo será desenvolvido utilizando o editor de código Visual Studio Code versão 1.52 e a linguagem de programação Node.js.

Tabela 1 - Especificações do computador

Processador:	Core i5 5200u
Memória RAM:	6GB DDR3
Armazenamento:	SSD 240GB

Fonte: O autor, 2020

4 RESULTADOS E DISCUSSÃO

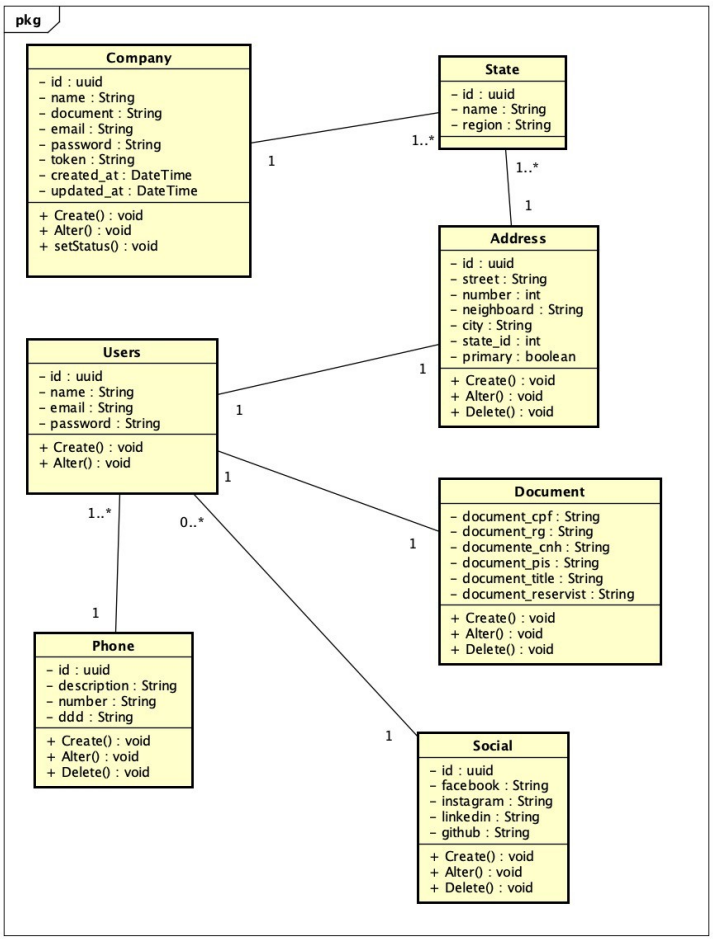
Inicialmente, foram criados diagramas UML que serviram para ilustrar a implementação da API. Estes diagramas podem ser observados na seção 4.1.

Após a implementação da API, foram realizados testes das funcionalidades implementadas, para validar o funcionamento e a viabilidade de se usar a API por sistemas de terceiros que consumam informações pessoais de clientes. As subseções a seguir serão divididas em cada *endpoint* específico para melhor explicação da API.

4.1 DIAGRAMAS UML

Na Figura 3 pode ser observado o diagrama de classes que foi elaborado para guiar a implementação da API. Dentro desse diagrama está descrito as classes com seus atributos e métodos. A partir deste ponto a base do código pôde ser iniciado, levando em conta também os relacionamentos entre cada classe.

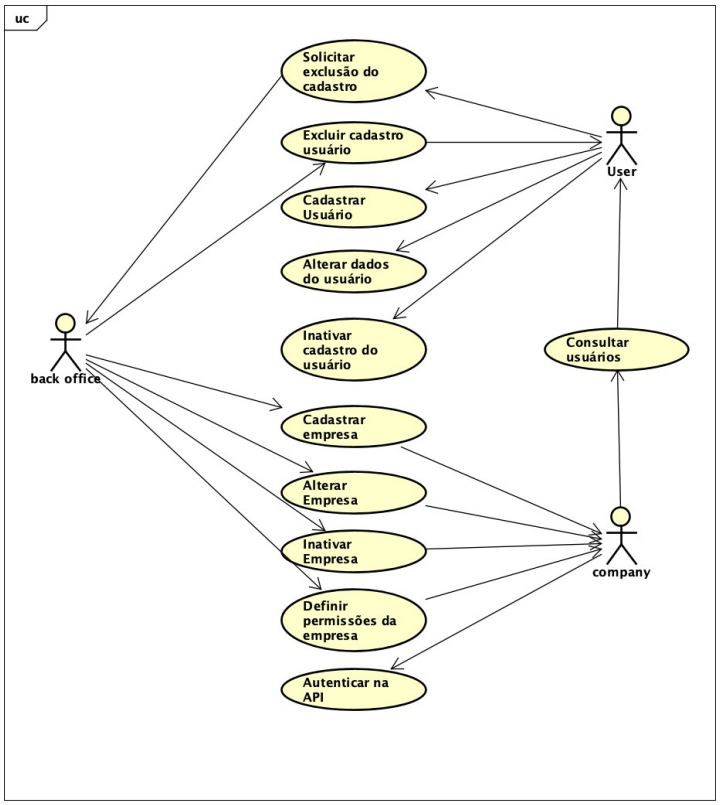
Figura 3 - Diagrama de classes



Fonte: O autor, 2020

Além do diagrama de classes, foi criado também um diagrama de caso de uso, que foi utilizado para descrever como a API funciona na prática. Este diagrama tem como objetivo demonstrar diferentes maneiras que o usuário poderá interagir com a API. O diagrama de caso de uso resume os detalhes dos usuários do seu sistema (também conhecidos como atores) e as interações deles com o sistema. Na Figura 4 pode ser observado o diagrama, que mostra os processos que cada ator pode fazer dentro do sistema.

Figura 4 - Diagrama de caso de uso

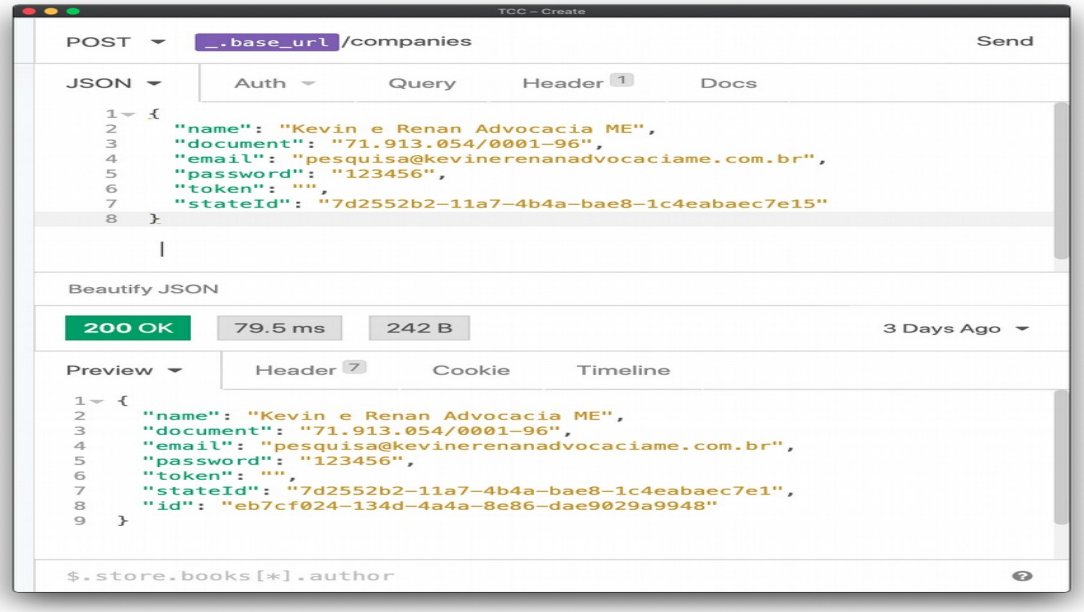


Fonte: O autor, 2020

4.2 COMPANIES

O *endpoint companies* é responsável pelo armazenamento e gerenciamento das empresas cadastradas dentro da API. Nesta parte do código, será disponibilizada as opções de criar uma nova empresa a partir do comando POST, como observado na Figura 5 abaixo.

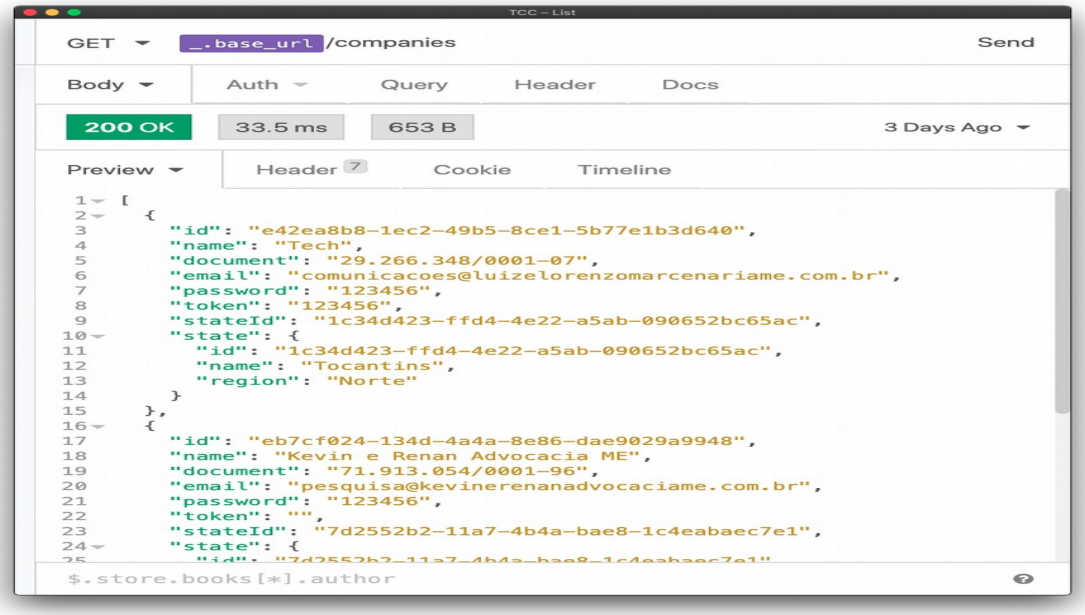
Figura 5 – Companies (POST)



Fonte: O autor, 2020

Para fazer a recuperação ou listagem das informações de empresas cadastradas no *companies* foi utilizado o comando GET, como mostrado na Figura 6.

Figura 6 - Companies (GET)

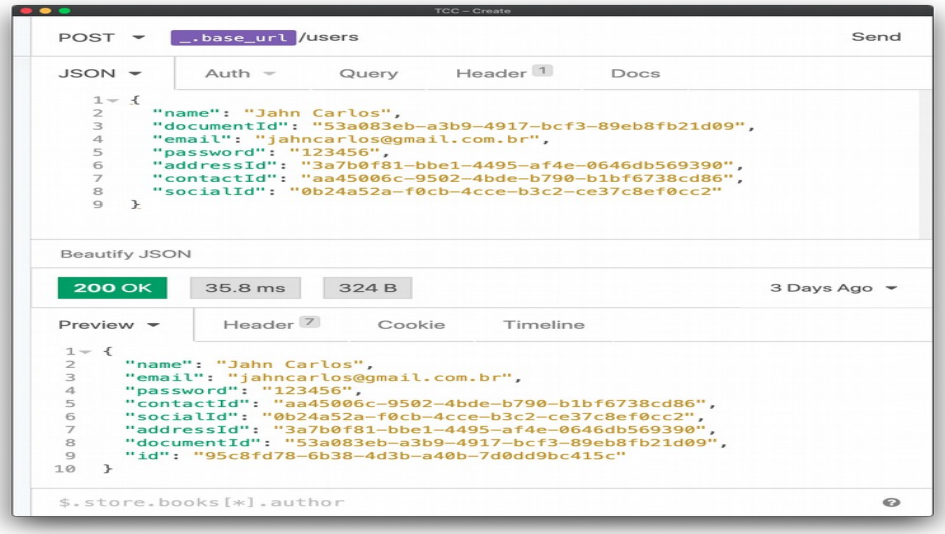


Fonte: O autor, 2020

4.3 USERS

O *endpoint users* é responsável por fazer o armazenamento e gerenciamento dos dados dos usuários. Dentro dela, é possível criar novos usuários, alterá-los e listá-los para consulta. A Figura 7 abaixo apresenta a parte do código onde são feitas criações de novos usuários.

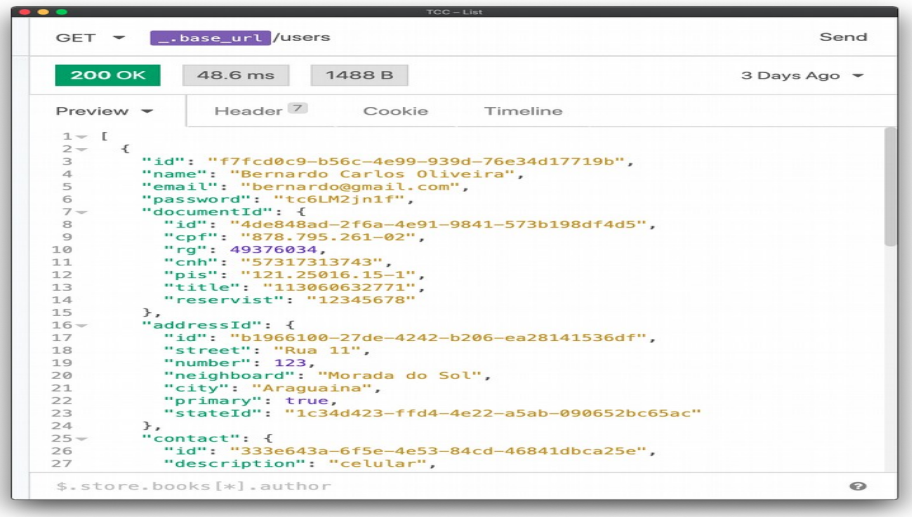
Figura 7 – Users (POST)



Fonte: O autor, 2020

Já para fazer a recuperação ou listagem das informações de usuários que foram cadastrados na *endpoint users* foi utilizado o comando GET, como mostrado na Figura 8 abaixo.

Figura 8 – Users (GET)

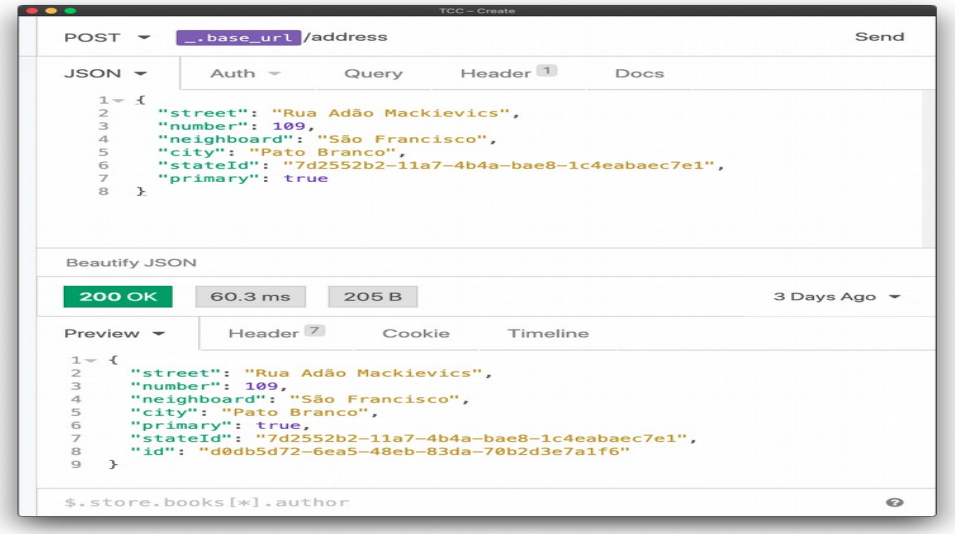


Fonte: O autor, 2020

4.4 ADDRESS

O *endpoint address* é responsável por fazer o armazenamento e gerenciamento dos dados relacionados. Dentro dela, é possível criar novos endereços de usuários, alterá-los e listá-los para consulta. A Figura 9 abaixo apresenta a parte do código onde são feitas criações de novos usuários.

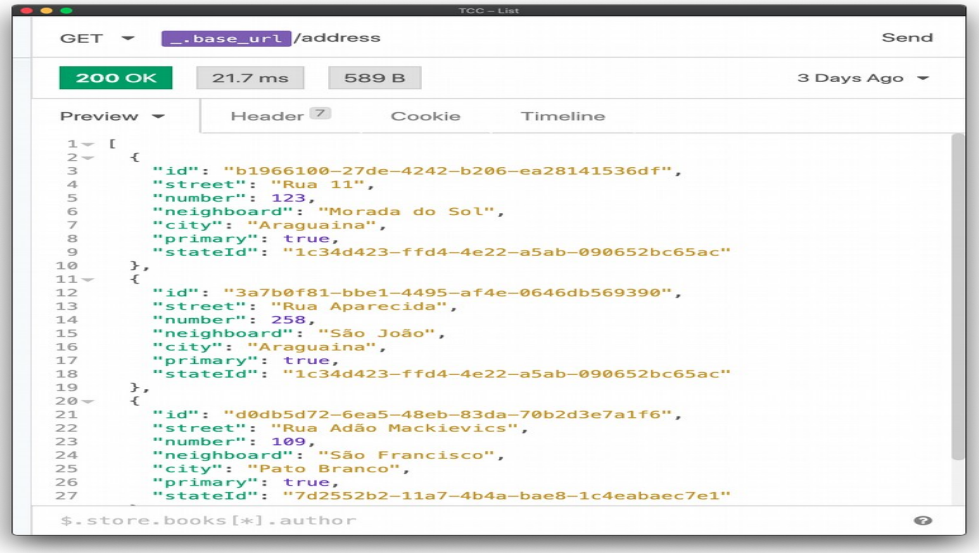
Figura 9 – Address (POST)



Fonte: O autor, 2020

Quando se torna necessário fazer a recuperação ou listagem das informações relacionadas a endereços que foram cadastrados na *endpoint address,* utiliza-se o comando GET, como mostrado na Figura 10 abaixo.

Figura 10 – Address (GET)

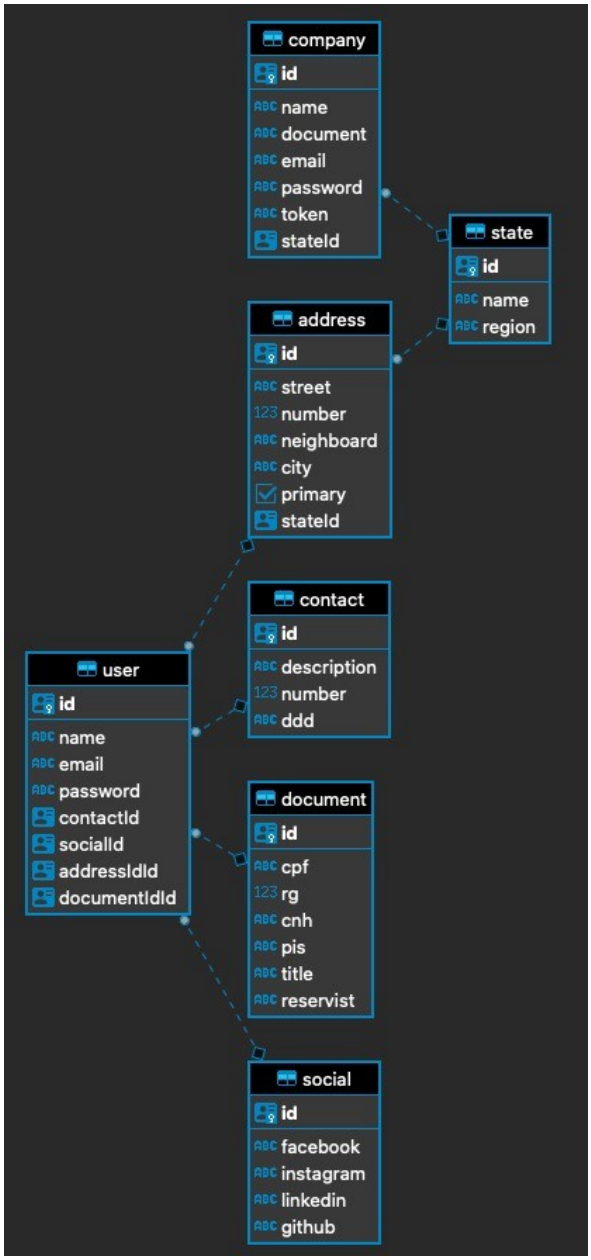


Fonte: O autor, 2020

4.5 DIAGRAMA ENTIDADE-RELACIONAMENTO

O diagrama entidade-relacionamento foi construído para basear a construção do banco de dados utilizado. Este diagrama foi utilizado para facilitar a comunicação entre os autores na hora de criar o banco de dados. O diagrama de entidade-relacionamento pode ser observado na Figura 11 a seguir.

Figura 11 - Diagrama Entidade-Relacionamento



Fonte: O autor, 2020

Vale frisar que todas as funcionalidades do código da API seguem a Lei Geral de Proteção de Dados, como é o objetivo da criação deste projeto. Todas as informações pessoais dos clientes serão recuperadas somente mediante prévia autorização da mesma.

A API pode ser consumida por aplicações de terceiros com objetivos comerciais e facilita a adequação da empresa a lei de proteção de dados, uma vez que a API já oferece as ferramentas necessárias relacionadas a autorização dos clientes a compartilhar seus dados.

5 CONCLUSÃO

A API REST proposta para centralizar os dados pessoais dos usuários em apenas uma plataforma e poder gerenciá-las faz o uso de técnicas de programação avançadas, sendo elas o uso de banco de dados para persistir os dados, DevOps para fornecer o ambiente e recursos necessários para programar e linguagens de programação para a implementação propriamente dita.

A aplicação implementada foi capaz de cumprir seus objetivos, ou seja, ela armazena e disponibiliza para consulta todas as informações pessoas previamente cadastradas. Empresas e instituições podem utilizar desta API para construir aplicações próprias de cadastros de clientes, buscando os dados pessoais e mantendo-se dentro da Lei Geral de Proteção de Dados

A partir de testes feitos com a API, pode-se chegar à conclusão de que é possível qualquer empresa continuar acessando dados pessoais dos seus clientes com seus próprios sistemas sem ferir a nova lei de proteção de dados. Ao utilizar a API criada, é possível utilizar de funções básicas como: cadastrar, excluir e inativar usuários, alterar os dados dos usuários, cadastrar e inativar empresas, além de definir as permissões de acesso aos dados por parte destas empresas.

Em uma proposta de trabalho futuro, pode ser implementada uma aplicação CRM – *Customer Relationship Management* que tem como finalidade gerenciar todo tipo de informações relacionadas a venda, marketing e atendimentos ao cliente. Esta aplicação consumiria os dados pessoais do cliente através da API REST que foi desenvolvida e já estaria adequada à nova Lei Geral de Proteção de Dados.

Desta forma, seria construída uma aplicação que seria utilizada comercialmente por empresas que precisam de uma tecnologia que auxilia a se relacionar e fidelizar o cliente.

REFERÊNCIAS

BASTOS, ISYS. *TechTudo.* 2020. https://www.techtudo.com.br/noticias/2020/09/lgpd-ja-esta-em-vigor-entenda-prazo-multa-e-mais-detalhes-da-lei-no-brasil.ghtml (acesso em 19 de Novembro de 2020).

Brasil. “Lei Geral de Proteção de Dados.” 14 de Agosto de 2018. http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm (acesso em 11 de Janeiro de 2021).

BUCKLER, CRAIG, et al. *JavaScript.* SitePoint Pty, Limited, 2018.

FLANAGAN, DAVID. *JavaScript O guia definitivo.* Bookman Editora LTDA, 2013.

G1 NOTÍCIAS. “G1 NOTÍCIAS.” *G1 NOTÍCIAS.* 2019. https://g1.globo.com/economia/tecnologia/noticia/2019/01/09/cambridge-analytica-se-declara-culpada-por-uso-de-dados-do-facebook.ghtml (acesso em 04 de Novembro de 2020).

GILES, DAVID. *Psychology of the Media.* Palgrave Macmillan, 2010.

JEEVA, CHELLADHURAI, RAJ PETHURU, e VINOD SINGH. *Learning Docker.* Packt Publishing, 2015.

KANTOR, ILYA. *The Modern JavaScript Tutorial - Part I. The JavaScript Language.* Github, 2019.

LAVADO, THIAGO. *G1.* 2019. https://g1.globo.com/economia/tecnologia/noticia/2019/08/28/uso-da-internet-no-brasil-cresce-e-70percent-da-populacao-esta-conectada.ghtml (acesso em 01 de Outubro de 2020).

LENON. *opus-software.* 2018. https://www.opus-software.com.br/node-js/ (acesso em 14 de Dezembro de 2020).

MAHARRY, DAN. *TypeScript revealed.* Apress, 2013.

MATTHIAS, KARL, e SEAN KANE. *Docker: Up and Running.* O'Reilly Media, 2015.

Ortiz, Silvia Maria. “IDC.” *IDC.* 27 de Fevereiro de 2020. https://www.idc.com/getdoc.jsp?containerId=prLA46093420 (acesso em 2021 de 2021).

Postgresql.org. “postgresql.org.” *postgresql.org.* s.d. https://www.postgresql.org/ (acesso em 04 de Janeiro de 2021).

REALE, BRIAN. *opensource.* 2017. https://opensource.com/article/17/8/open-core-vs-open-perimeter (acesso em 19 de Novembro de 2020).

REYES, ANA LUÍZA. “Ministério Público Federal.” *Ministério Público Federal.* 2018. http://www.mpf.mp.br/sp/sala-de-imprensa/noticias-sp/mpf-processa-microsoft-para-que-windows-10-deixe-de-coletar-dados-pessoais-sem-autorizacao-dos-usuarios (acesso em 04 de Dezembro de 2020).

RIBEIRO, FELIPE. *Canaltech.* 2020. https://canaltech.com.br/negocios/mercado-brasileiro-de-tecnologia-e-comunicacao-crescera-49-em-2020-preve-idc-160055/ (acesso em 22 de Dezembro de 2020).

RUNGTA, KRISHNA. *Learn NodeJS in 1 Day: Complete Node JS Guide with Examples.* Independently publishied, 2016.

SOUZA, IVAN DE. *rockcontent.* 2020. https://rockcontent.com/br/blog/rest-api/ (acesso em 21 de Dezembro de 2020).