SIGLA: SISTEMA DE GERENCIAMENTO DE LICENÇAS AMBIENTAIS

Maury Sayão Lobato Abreu[1]

Silvio Cesar Viegas[2]

RESUMO

O gerenciamento dos prazos e condicionantes de licenças ambientais é um aspecto fundamental dentro do licenciamento ambiental de empreendimentos sujeitos a tais controles. A não solicitação ou a solicitação de licenças fora do prazo gera multas a um empreendimento. Deste modo, o presente artigo realiza a análise de um protótipo MVP (mínimo produto viável) de uma aplicação que permite realizar o gerenciamento dos prazos de licenças cadastradas, bem como oferecer apoio ao processo de renovação de licenças ambientais de empreendimentos da região sul do Brasil. A aplicação será desenvolvida com tecnologias de aplicação *web* com uso de banco de dados NoSQL. São apresentados protótipos de tela, proposta arquitetural e linhas de código para algumas das funcionalidades propostas.

Palavras-chave: licenciamento ambiental; *Embedded JavaScript*; *Node.js*; *MongoDB*; sul do Brasil.

ABSTRACT

The management of the term and conditions of environmental permits is a fundamental aspect in the environmental permits process subject to such controls. The failure in request or the request after the deadline generate fines to the enterprise. This way, this paper analyzes a minimum viable product (MVP) prototype of an application that allows managing the deadlines of registered permits, as well as offer support to the process of renewing environmental permits for enterprises in the southern Brazil. The application will be developed with web development technologies using NoSQL database. Screen prototypes, architectural proposal and code lines for some features are presented.

Keywords: environmental licensing; Embedded JavaScript; Node.js; MongoDB; southern Brazil.

1 INTRODUÇÃO

O licenciamento ambiental é o processo administrativo pelo qual os órgãos ambientais emitem as licenças ambientais que autorizam a instalação, ampliação e

[1] Mestre em Biologia (Unisinos, São Leopoldo, RS), Graduando em Análise e Desenvolvimento de Sistemas (FAQI, Gravataí, RS). Contato: maury.abreu@gmail.com. GitHub: https://www.github.com/MaurySLA .

[2] Professor mestre, coordenador dos cursos do Eixo Tecnologia e Educação da FAQI, Gravataí/RS. Contato: silvio.viegas@qi.edu.br

operação de empreendimento e atividades utilizadoras de recursos ambientais ou que sejam potenciais causadoras de degradação ambiental (CONAMA, 1997). Estes documentos, além de autorizar as atividades também estabelecem condições, restrições e atividades para controle ambiental que devem ser obedecidas e cumpridas pelo empreendedor. Toda licença ambiental possui prazo de validade que, se não renovado, acarreta em multas ao empreendedor.

Embora existam ferramentas disponíveis para o gerenciamento de atividades e prazos em geral, muitas destas não têm boa aceitação em certos públicos ligados ao segmento ambiental. Deste modo, este estudo analisa o desenvolvimento de um mínimo produto viável (MVP) para uma aplicação que apoie a realização de tais atividades, alertando quanto a prazos através de mensagens eletrônicas e indicação de normas, portarias e resoluções a serem obedecidas durante o processo de licenciamento.

Este estudo está organizado em cinco seções, sendo a primeira delas introdutória, apresentando o tema e a delimitação do tema, a problemática envolvida, a justificativa e os objetivos. A seção seguinte apresenta a fundamentação teórica, contextualizando o leitor quanto ao conhecimento sobre licenças ambientais e as tecnologias a serem utilizadas. Na sequência é apresentada a metodologia utilizada no estudo e, na quarta sessão, o desenvolvimento do mesmo, destacando a modelagem, a arquitetura, protótipos de tela e codificações importantes a serem utilizadas. A quinta seção apresenta os resultados obtidos, seguido de considerações finais sobre o estudo.

1.1 TEMA E DELIMITAÇÃO DO TEMA

O tema do presente estudo será o *licenciamento ambiental*. Considerando-se a amplitude do assunto, o tema irá se limitar aos *prazos de vencimento de licenças ambientais de empreendimentos realizados nos estados do sul do Brasil*.

1.2 PROBLEMA

O licenciamento ambiental é o processo administrativo segundo o qual os órgãos ambientais competentes autorizam a instalação, ampliação e operação de empreendimento e atividades utilizadoras de recursos ambientais ou que sejam potenciais causadoras de degradação ambiental (CONAMA, 1997). Isto inclui, por exemplo, empreendimentos como usinas elétricas, parques eólicos, linhas de transmissão de energia, construção de rodovias, postos de gasolina, empreendimentos imobiliários, entre muitos outros.

Os processos de licenciamento ambiental envolvem vários procedimentos, entre eles a emissão de licenças ambientais. É através das licenças ambientais que os órgãos competentes autorizam a implantação, operação ou ampliação de um empreendimento, estabelecendo condições, restrições e atividades para o controle ambiental (CONAMA, 1997). Na região sul do Brasil os processos de licenciamento ambiental são realizados pela Fundação Estadual de Proteção Ambiental Henrique Luiz Roessler (FEPAM) no Rio Grande do Sul, o Instituto do Meio Ambiente (IMA) de Santa Catarina (antes FATMA – Fundação do Meio Ambiente) e o Instituto Água e Terra (IAT) no Paraná (antes IAP – Instituto Ambiental do Paraná).

Embora os processos de licenciamento variem entre órgãos ambientais e tipos de empreendimentos, existem algumas constâncias. Por exemplo, licenças ambientais sempre possuem prazos de vigência, as quais podem ser renovadas através do cumprimento das condições e restrições supracitadas e solicitação formal perante o órgão ambiental. Tal solicitação, porém, deve ser realizada com pelo menos 120 dias de antecedência em relação ao vencimento da licença, sob pena de multa.

1.3 JUSTIFICATIVA

O gerenciamento de atividades e prazos é essencial para o cumprimento das exigências relacionadas ao licenciamento ambiental. Embora existam muitas ferramentas disponíveis, muitas delas não tem boa aceitação em alguns públicos,

como o das empresas ligadas a este segmento. Gestores ambientais geralmente se valem de planilhas eletrônicas e anotações diversas a fim de controlar seus prazos, embora nem todos os envolvidos no projeto tenham acesso a tais documentos. Além disso, uma vez que as normas e procedimentos variam entre tipos de empreendimentos e órgãos ambientais envolvidos, muitas vezes é necessária uma ampla pesquisa prévia a fim de descobrir como proceder – o que pode se tornar um grave problema quando os prazos estão muito perto do vencimento. Deste modo, este estudo propõem a análise de um mínimo produto viável para uma aplicação que apoie a realização de tais atividades, alertando quanto a prazos através de mensagens eletrônicas e indicação de quais normas, portarias e resoluções devem ser obedecidas durante o processo de licenciamento.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Analisar o desenvolvimento de um protótipo de mínimo produto viável para uma aplicação que auxilie no gerenciamento de prazos e indique as normas e procedimentos para a renovação de licenças ambientais emitidas por órgãos da região sul do Brasil, intitulada SiGLA (Sistema de Gerenciamento de Licenças Ambientais).

1.4.2 Objetivos Específicos

- Analisar o desenvolvimento uma aplicação que permita o cadastro de licenças ambientais, armazenando dados do empreendimento (nome, CNPJ e tipo de empreendimento) e da licença (nº da licença, tipo de licença, órgão expedidor e prazo de validade);
- Permitir que a aplicação defina três categorias de prazo, baseado no prazo de validade da licença: licença vigente (mais de 200 dias até o vencimento

da licença), renovação necessária (entre 120 e 200 dias até o vencimento) e prazo de renovação expirado (119 dias até o vencimento);

- Permitir que a aplicação indique a Portaria, Instrução Normativa ou Resolução que deve ser obedecida para o processo de renovação, de acordo com o tipo de empreendimento e o órgão expedidor.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CONHECIMENTOS SOBRE LICENÇAS AMBIENTAIS

A Constituição Federal de 1988 determina, em seu Art. 225:

> Todos têm direito ao meio ambiente ecologicamente equilibrado, bem de uso comum do povo e essencial à sadia qualidade de vida, impondo-se ao Poder Público e à coletividade o dever de defendê-lo e preservá-lo para as presentes e futuras gerações (BRASIL, 1988).

Na sequência, o § 1º do referido artigo cita as incumbências do Poder Público para assegurar tal direito, incluindo, entre elas, a exigência de estudos ambientais prévios à instalação de empreendimentos potencialmente causadores de degradação ambiental (inciso IV). Neste contexto, a Resolução Conama de 1997 define o licenciamento ambiental como processo administrativo pelo qual os órgãos ambientais competentes autorizam a instalação, ampliação e operação destes mesmos empreendimentos (CONAMA, 1997). Esse processo é feito através da emissão de licenças ambientais, as quais podem ser classificadas em três tipos principais: Licença Prévia (LP), expedida na fase de planejamento do empreendimento e que determina requisitos para as próximas fases; Licença de Instalação (LI), a qual autoriza a implantação do empreendimento e especifica todas as medidas de controle que devem ser obedecidas na fase de implantação; e Licença de Operação (LO), a qual autoriza a operação da atividade, também determinando medidas de controle (CONAMA, 1997, Art. 8º). Independentemente de seu tipo, todas as licenças possuem prazos de vigência, os quais podem ser

renovados. Tal renovação é especialmente importante nos casos de LO, uma vez que tais licenças devem permanecer válidas durante todo o período de operação do empreendimento ou atividade. Adicionalmente, a formalização de solicitação de renovação da LO deve ocorrer com antecedência mínima de 120 dias da expiração de seu prazo de validade (CONAMA, 1997, Art. 18 § 4º).

Na região sul do Brasil os processos de licenciamento ambiental são realizados pela Fundação Estadual de Proteção Ambiental Henrique Luiz Roessler (FEPAM) no Rio Grande do Sul, o Instituto do Meio Ambiente (IMA) de Santa Catarina e o Instituto Água e Terra (IAT) no Paraná. Empreendimentos que se localizem em mais de um estado tem seus processos iniciais de licenciamento (LP e LI) realizados pelo Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (IBAMA) (CONAMA, 1997, Art. 4º), enquanto para os processos de operação (emissão de LO) o empreendedor deve obedecer às normas determinadas por todos os estados compreendidos pela atividade.

Situações específicas preveem outros tipos de licença, como a Licença Prévia de Instalação para Ampliação (LPIA), expedida em situações de alteração de um empreendimento (FEPAM, 2020) e as chamadas licenças de fauna ou Autorização Ambiental para Manejo de Fauna Silvestre, obrigatórias para quaisquer situações que exijam manejo de fauna (como relocação de espécies ou captura de qualquer natureza). Atualmente, porém, estas situações específicas não constituem escopo do SiGLA.

2.2 TECNOLOGIAS

O presente artigo propõe a análise de um MVP de uma aplicação para gerenciamento de licenças ambientais intitulado SiGLA. Trata-se de uma aplicação, desenvolvida em linguagem HTML em um ambiente de execução *JavaScript*, o *Node.js*. Para o armazenamento dos dados será utilizado o banco de dados NoSQL *MongoDB*. Aspectos sobre as ferramentas e técnicas utilizadas no desenvolvimento serão abordados e discutidos a seguir.

2.2.1 Linguagem de Modelagem Unificada e Diagrama de Casos de Uso

Segundo Morais e Zanin (2017) a Linguagem de Modelagem Unificada (ou UML, do inglês *Unified Modeling Language*) “traz para o contexto computacional a padronização de uma linguagem utilizada para representar informações referentes a um sistema”. Tal linguagem é usada para auxiliar na representação da forma como os objetos se comunicam dentro de um sistema, muito utilizada em atividades de engenharia de *software* e é o padrão adotado pelo Grupo de Gerenciamento Dirigido a Objeto (PAGE-JONES, 2001). A linguagem é usada, por exemplo, para a criação de diagramas de casos de uso, os quais, segundo Reinehr (2020), ajudam o analista de requisitos a se comunicar com o usuário; ou seja, eles fornecem uma representação visual dos requisitos do sistema, os quais são entendidos como condições ou capacidades que devem ser atendidos pelo sistema para satisfazer as especificações estabelecidas (REINEHR, 2020). Segundo Sommerville (2007, p.102) os casos de uso “se tornaram uma característica fundamental da notação UML para descrição de modelos de sistemas orientados a objetos”. O autor ainda comenta que “em sua forma mais simples um caso de uso identifica o tipo da interação e os agentes envolvidos”.

Será montado um diagrama de caso de uso para representar o uso do SiGLA, indicando os autores que participam do caso de uso, as ações e suas interações.

2.2.2 HiperText Markup Language (HTML) e Cascading Style Sheet (CSS)

A linguagem de marcação de hipertexto (do inglês *HiperText Markup Language* – HTML) é a principal linguagem utilizada para o desenvolvimento de *websites* e aplicações *web*. Um hipertexto é um documento formado por distintos blocos de informação interligados por elos de associação (FLATSCHART, 2011). Deste modo, a linguagem utiliza certas marcações (chamadas de etiquetas ou *tags*) para informar o que aquele item em particular representa: um parágrafo, uma imagem, uma tabela, etc. Além disso, o HTML aceita uma formatação semântica de

apresentação, o que permite definir a maneira como as informações serão mostradas ao usuário (FLATSCHART, 2011). Essa linguagem precisa ser interpretada, o que é feito através de um navegador *web*, como Google Chrome, Mozilla Firefox, Apple Safari entre outros, sem a necessidade de que os dados HTML sejam compilados, como ocorre com outras linguagens de programação (SEGURADO, 2015).

Enquanto o HTML é uma linguagem de marcação, o CSS (do inglês *Cascading Style Sheet*, Folha de Estilos em Cascata) é uma linguagem de estilos, responsável pela formatação e apresentação do conteúdo em um HTML (FLATSCHART, 2011). Embora ainda seja possível inserir a configuração de estilo de um *website* dentro do próprio arquivo HTML, essa prática não é muito recomendada, uma vez que no caso de *websites* com muitos arquivos HTML uma modificação ou correção em um estilo de formatação se torna muito mais complicado e demorado em situações como essa. Deste modo, o CSS concentra em si toda a formatação de estilos, deixando para o HTML apenas o trabalho de descrever o conteúdo do *website* ou aplicação. Isso permite que o conteúdo e os estilos sejam trabalhados de forma independente, conferindo flexibilidade e modularidade ao fluxo de trabalho (FLATSCHART, 2011). Além disso, é importante destacar que o CSS não define apenas características como cor e tipo de fonte, mas também dimensionamento, margens, espaçamento, posicionamento dos elementos na tela entre outras características. (SEGURADO, 2015).

Embora o desenvolvimento de *websites* seja o uso mais frequentemente associado à linguagem de marcação de hipertexto, atualmente muitos desenvolvedores fazem uso dessa linguagem para criação de APIs (do inglês *Application Programming Interface*), definida por Flatschart (2011, p.24) como “uma interface que permite a interação entre os *softwares* facilitando sua integração de maneira semelhante a uma interface que auxilia na comunicação entre nós, usuários, e os diversos dispositivos com os quais convivemos no nosso dia a dia”. Deste modo, o HTML se mostra como uma boa opção para o desenvolvimento de sistemas simples que não requerem grandes servidores ou sistemas operacionais, como é o caso do SiGLA.

2.2.3 JavaScript (JS) e JavaScript Incorporado (EJS)

Chamada de linguagem de comportamento, o *JavaScript* é uma linguagem que pode ser incorporada diretamente ao HTML com o intuito de aumentar sua interatividade (FLATSCHART, 2011). Segundo Segurado (2015), “a maioria dos sites modernos usa *JavaScript*, e todos os navegadores atuais [...] incluem interpretadores para essa linguagem”. Pelo uso dessa linguagem o programador oferece ao usuário a capacidade de alterar textos, ocultar e exibir objetos, modificar estilos, entre outras funcionalidades (SILVA; MILETTO, 2014). A linguagem *JavaScript* é considerada uma linguagem *cliente side*, ou seja, funciona do lado do cliente, sendo executada utilizando apenas o navegador no computador do usuário (FLATSCHART, 2011; SILVA; MILETTO, 2014). Entretanto, ela também pode ser usada no *server-side* (lado do servidor) pelo uso de ambientes de execução.

Para o desenvolvimento do SiGLA o uso do *JavaScript* será essencial para a comunicação entre os formulários HTML e o banco de dados que irá armazenar as informações. A linguagem também permitirá maior interação entre o usuário e a aplicação, exibindo e escondendo janelas conforme sua necessidade. Por fim, também será feito uso do *JavaScript* Incorporado (*Embedded JavaScript* – EJS) uma *engine* de visualização que permite o uso de *JavaScript* e HTML simultaneamente. Através do EJS é possível incorporar uma lógica de programação no código HTML antes de enviá-lo ao navegador, fazendo com que a informação apresentada ao usuário seja ainda mais dinâmica. Em uma postagem na internet, Harumi (2021) afirma que com o EJS “conseguimos de uma maneira fácil e simples transportar dados do *back-end* para o *front-end*”, o que é essencial para a comunicação entre o HTML e o banco de dados que será utilizado.

2.2.4 Node.js e Express

O *Node.js* pode ser definido como um ambiente de execução *JavaScript* no lado do servidor com o qual é possível criar aplicações *JavaScript* que rodem em uma máquina sem a necessidade de um servidor ou de um navegador (TOTVS, 2020). O *Node.js* é uma tecnologia assíncrona que trabalha em uma única *thread* (linha ou encadeamento) de execução. Desse modo, cada requisição ao *Node.js* não bloqueia o processo, atendendo a um volume absurdamente grande de solicitações simultaneamente. Assim, quando o *Node.js* recebe uma requisição, ele executa essa requisição, delegando a tarefa conforme a necessidade, e não espera uma resposta, seguindo para a próxima requisição. Quando a requisição termina e é respondida, ela entra no fluxo principal para ser devolvida. É diferente do funcionamento da maioria das linguagens de programação que trabalham com o *multi-thread*, onde cada requisição recebida interrompe o processo até que haja uma resposta. O modo *multi-thread* é mais simples para programar, mas muito mais oneroso para o *hardware*, que consome mais recursos para o processamento.

Para o desenvolvimento do SiGLA será criado um ambiente de execução com *Node.js* e o *framework* Express. Este *framework* é flexível e oferece um conjunto robusto de ferramentas e será usado, entre outras, para o desenvolvimento das rotas. Conforme mencionado anteriormente, também será feito uso do *JavaScript* Incorporado (EJS), o qual será instalado como dependência da aplicação.(ALVES, 2020).

2.2.5 Banco de Dados NoSQL e MongoDB

Medeiros (2013, p.15) define banco de dados como “um conjunto de dados com certa organização característica, com o objetivo de armazenamento persistente dos dados e dotado de mecanismos de manipulação para obtenção de informações e recuperação posterior dentro de um sistema de informação”. Deste modo, um banco de dados organiza informações relacionadas dentro de um sistema de modo que seja possível acessá-las posteriormente.

Os primeiros bancos de dados eram do tipo relacional, os quais tinham uma estrutura tabular dos dados e utilizavam a linguagem SQL (*Structured Query Language*, traduzido como Linguagem de Consulta Estruturada). Depois dos anos 2000 começaram a surgir os bancos de dados NoSQL (*Not Only SQL*, Não Apenas SQL), mais apropriados para grandes volumes de dados. Foi perto desta época que surgiu o MongoDB, um banco de dados de documentos projetado para facilitar o desenvolvimento e o dimensionamento (MONGODB INC., 2021).

O MongoDB oferece opções de implementação local e hospedagem em nuvem. Entre suas principais características estão a alta performance, a linguagem de consulta avançada, alta disponibilidade, escalabilidade horizontal e suporte para múltiplos mecanismos de armazenamento (MONGODB INC., 2021). Elmasri e Navathe (2018) explicam também que no MongoDB é um banco de dados orientado a documentos, os quais são armazenados em BSON (*Binary JavaScript Object Notation*) dentro de coleções e que a maioria de suas atualizações é atômica quando se refere a um único documento, embora também ofereça padrões para especificar transações em vários documentos.

Por todas estas características, além de ser um banco de dados de código aberto e gratuito, o MongoDB será utilizado como sistema de gerenciamento de banco de dados na aplicação SiGLA. (LUIZTOOLS, 2017).

3 METODOLOGIA

Lozada (2019) define o método de uma pesquisa científica como “o caminho pelo qual se chega a determinado resultado” e que o método “indica como o pesquisador deve proceder ao longo do caminho para obter o resultado pretendido”. Neste sentido, no atual projeto a metodologia se apresenta como um roteiro a ser seguido para se alcançar os objetivos propostos. Além disso, considerando a aplicabilidade do que se propõem, será abordada neste projeto a pesquisa aplicada, a qual surge da necessidade de resolver um problema que faz parte do contexto profissional do pesquisador (NUNES, 2019). A estrutura proposta para o

desenvolvimento da pesquisa está sumarizada no Quadro 1, e é detalhada na sequência.

Quadro 1 - Sumarização da metodologia proposta para o SiGLA.

1. Definição	1.1. Tema e delimitação do tema abordado; 1.2. Linguagem de programação utilizada; 1.3. Banco de dados utilizado.
2. Conhecimento	2.1. Licenciamento ambiental; 2.2. Regras de negócio.
3. Desenvolvimento	3.1. Diagrama de caso de uso. 3.2. Arquitetura da API; 3.3. Desenvolvimento da Interface HTML; 3.4. Codificação *JavaScript*; 3.5. Integração com banco de dados.
4. Resultado	4.1. Testes e validação; 4.2. Propostas de melhoria.

Fonte: elaborado pelo autor, 2022.

1. Definição. Neste ponto foram definidas as especificações do projeto, contemplando o tema e a delimitação do tema a ser abordado, a linguagem de programação a ser utilizada, bem como o banco de dados onde as informações serão armazenadas.

1.1. Tema e delimitação do tema abordado. Para o projeto atual se determinou como tema e delimitação do tema os prazos de vencimento das licenças ambientais de empreendimentos realizados na região sul do Brasil.

1.2. Linguagem de programação utilizada. Se definiu o uso da linguagem HTML juntamente de *JavaScript* para o desenvolvimento da aplicação. Também se definiu o *Node.js* como o ambiente de execução *server-side* da aplicação.

1.3. Banco de dados utilizado. Como Sistema de Gestão de Banco de Dados (SGBD) determinou-se o uso do MongoDB devido às suas opções de implementação local e hospedagem em nuvem e alta performance.

2. Conhecimento. Houve a necessidade de revisão teórica a respeito do tema abordado no projeto, com atenção aos aspectos que influenciam o desenvolvimento da aplicação.

2.1. Licenciamento ambiental. Definida a delimitação do tema, o mesmo foi pesquisado, todos os aspectos relativos à renovação de licenças ambientais foram verificados, incluindo normas a serem seguidas em cada órgão ambiental e a necessidade de protocolo de solicitação de renovação com antecedência mínima de 120 dias.

2.2. Regras de negócio. As regras de negócio são padrões que condicionam o funcionamento do negócio, as diretrizes do negócio.(SCHEMES, 2020). Estas regras foram definidas neste ponto.

3. Desenvolvimento. Neste segmento da metodologia são apresentados os aspectos relativos à modelagem, codificação e estruturação do projeto.

3.1. Diagrama de caso de uso. Como método de representação do funcionamento da aplicação proposta, se escolheu o uso de diagrama de caso de uso, o qual utiliza a Linguagem de Modelagem Unificada (UML).

3.2 Arquitetura da API. Para a estruturação da aplicação optou-se por utilizar o padrão MVC (*model-view-controller*), utilizado para separar as partes distintas do projeto, reduzindo suas dependências.(ZENKER, 2019a). Também se objetiva seguir os princípios S.O.L.I.D., que compreendem a responsabilidade única, aberto/fechado, substituição de Liskov, segregação de interfaces e inversão de dependência com o intuito de facilitar a manutenção e reaproveitamento do código.(ZENKER, 2019b).

3.3. Desenvolvimento da Interface HTML. Neste tópico será utilizada a linguagem HTML para o desenvolvimento da interface do projeto.

3.4. Codificação JavaScript. Neste tópico será utilizada a linguagem *JavaScript* para desenvolver a interação do usuário com a interface HTML. Alguns processos serão desenvolvidos de maneira concomitante, a fim de habilitar e desabilitar telas dentro da interface HTML.

3.5. Integração com banco de dados. Desenvolvido de forma concomitante à codificação *JavaScript*, será realizada a integração entre as informações inseridas

pelo usuário na interface com o banco de dados, a fim de criar, ler, atualizar e apagar as informações (CRUD – *create, read, update, delete*).

4. Resultado. Neste ponto serão apresentados os protótipos de telas, a estrutura proposta para o projeto, bem como considerações a respeito de possíveis aprimoramentos ao projeto.

4.1. Testes e validação. Durante o desenvolvimento da aplicação serão realizados testes unitários; com a conclusão do projeto, porém, serão executados novos testes cobrindo amplas entradas de modo a verificar a coerência do armazenamento dos dados. Também será feita validação dos formulários disponíveis na interface.

4.2. Propostas de melhoria. Serão apresentadas e discutidas algumas propostas de melhoria e aprimoramento da aplicação, de modo a ampliar sua utilização dentro do tema geral (licenciamento ambiental).

4 DESENVOLVIMENTO

4.1 MODELAGEM

A seguir serão apresentados os requisitos funcionais do SiGLA. A Figura 1 apresenta o diagrama de caso contemplando os requisitos funcionais (RF) descritos a seguir.

4.1.1 RF01: Cadastro de Licença

A aplicação deve possuir um mecanismo para o cadastro de novas licenças, informando os seguintes campos: (i) nome do empreendimento; (ii) CNPJ do empreendimento; (iii) tipo de empreendimento; (iv) número da licença; (v) tipo de licença; (vi) órgão expedidor da licença; (vii) prazo de validade da licença; (viii) usuários de interesse.

O campo *CNPJ do Empreendimento* deve aceitar apenas números; o campo *Número da Licença* poderá aceitar números e letras, visto que algumas licenças são numeradas com códigos de letras e números. O campo *Tipo de Empreendimento*

deve oferecer uma lista de opções, contemplando minimamente as seguintes opções: Geração de Energia Hídrica, Geração de Energia Eólica, Transmissão de Energia, Setor Imobiliário. O campo Tipo de Licença deve oferecer as seguintes opções: Licença Prévia, Licença de Instalação, Licença de Operação e Outra Licença. O campo *Órgão Expedidor da Licença* deve oferecer as seguintes opções: FEPAM, IMA ou IAT. O campo *Usuários de Interesse* deverá ser preenchido com uma lista de e-mails de todos os usuários que serão alertados quando a licença mudar sua categoria de prazo (descrito no RF02).

O sistema não deve permitir o cadastro de duas licenças com o mesmo número e expedidas pelo mesmo órgão, mesmo que os demais campos sejam diferentes.

4.1.2 RF02: Categoria de Prazo

A aplicação deverá categorizar cada licença cadastrada no sistema em uma das seguintes categorias, de acordo com seu prazo de validade cadastrado: *licença vigente* (mais de 200 dias até o vencimento da licença), *renovação necessária* (entre 200 e 120 dias até o vencimento) e *prazo de renovação expirado* (119 dias até o vencimento). O sistema deve atualizar a categoria de cada uma das licenças cadastradas às 00:01 horas de cada dia. Toda vez que uma licença trocar sua categoria de prazo, os usuários de interesse cadastrados para aquela licença deverão ser alertados por mensagens de e-mail e um alerta será emitido assim que o sistema for acessado pelo usuário.

4.1.3 RF03: Normas de Renovação

Utilizando as informações do tipo de empreendimento e órgão expedidor, o sistema deve ser capaz de determinar qual a Portaria, Instrução Normativa, Resolução ou outra norma reguladora que o usuário deverá obedecer durante o processo de renovação da licença. Essa informação deve ser inserida na mesma mensagem de e-mail enviada aos usuários de interesse cadastrados para aquela licença mencionada no RF02.

4.1.4 RF04: Lista de Licenças Cadastradas

O sistema deve permitir que o usuário verifique todas as licenças cadastradas no sistema. A lista deve apresentar o nome e CNPJ do empreendimento, o número da licença e o prazo de validade.

4.1.5 RF05: Consulta de Licença

A aplicação deve permitir que o usuário faça uma pesquisa e encontre uma licença cadastrada, informando nome do empreendimento, CNPJ do empreendimento, número da licença ou licenças na categoria “renovação necessária” (entre 200 e 120 dias até o vencimento).

4.1.6 RF06: Atualização de Licença Cadastrada

O sistema deve permitir que o usuário atualize os dados de uma licença previamente cadastrada. O sistema deve alertar e solicitar confirmação antes de realizar as mudanças nos dados da licença.

4.1.7 RF07: Deleção de Licença Cadastrada

O sistema deve permitir que o usuário apague uma licença previamente cadastrada. O sistema deve alertar quanto aos riscos de apagar uma licença do sistema, solicitando confirmação.

4.2 ARQUITETURA

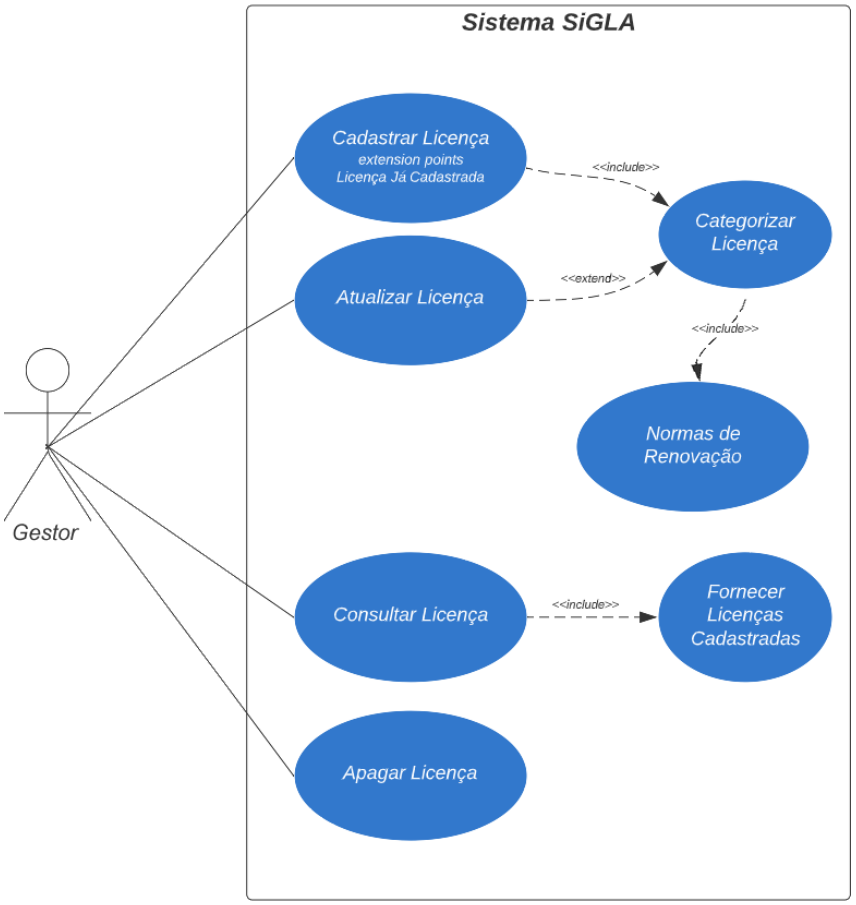
Para o desenvolvimento do SiGLA será adotado o padrão arquitetural MVC – *model-view-control*. A Figura 2 apresenta a distribuição das pastas e arquivos que compõem a aplicação. O padrão MVC pode ser observado pela presença de três camadas características:

- **models.** Esta camada armazena os modelos utilizados pela aplicação. Um arquivo *JavaScript* chamado Licence.js serve de modelo para a criação das licenças que serão armazenadas no banco de dados.
- **views.** Nesta camada estão armazenados os arquivos visualizados pelo usuário, basicamente todos os arquivos que podem ser acessados dentro da aplicação (arquivos EJS, linguagem HTML e JavaScript).
- **control.** Aqui estão armazenados os arquivos de controle da aplicação. Um arquivo *JavaScript* chamado *LicenceController.js* é responsável por fazer a "ponte" entre o modelo disponível na pasta "models" e os arquivos visualizados pelo usuário, disponíveis na pasta "views".

Além destas três camadas do padrão MVC a Figura 2 também mostra a presença de outras pastas e arquivos que fazem parte da estrutura da aplicação:

- **node_modules.** Os arquivos contidos nesta pasta compõem os módulos do *Node.js* utilizado na aplicação. Ela é criada automaticamente quando é feita a instalação no *Node.js* na aplicação.
- **public.** Pasta de arquivos públicos. Contém todos os arquivos estáticos que precisam ser acessados pelo servidor, incluindo as folhas de estilo (CSS), imagens (img) e os *scripts* utilizados na aplicação.
- **package-lock.json** e **package.json.** Estes arquivos JSON (*JavaScript Object Notation*) contém os pacotes utilizados pelo *Node.js*. Estes arquivos são gerados na instalação do *Node.js* e são atualizados quando novas dependências são adicionadas à aplicação.
- **routes.js.** Arquivo *JavaScript* contendo todas as rotas utilizadas na aplicação. É este arquivo que define o "caminho" que deve ser utilizado para acessar cada um dos arquivos armazenados na pasta "views".
- **server.js.** Arquivo *JavaScript* contendo as configurações do servidor, incluindo o acesso aos arquivos estáticos (armazenados na pasta "public"), a conexão com o banco de dados MongoDB, entre outras configurações.

Figura 1 - Diagrama de casos de uso contemplando os requisitos funcionais propostos para o SiGLA.

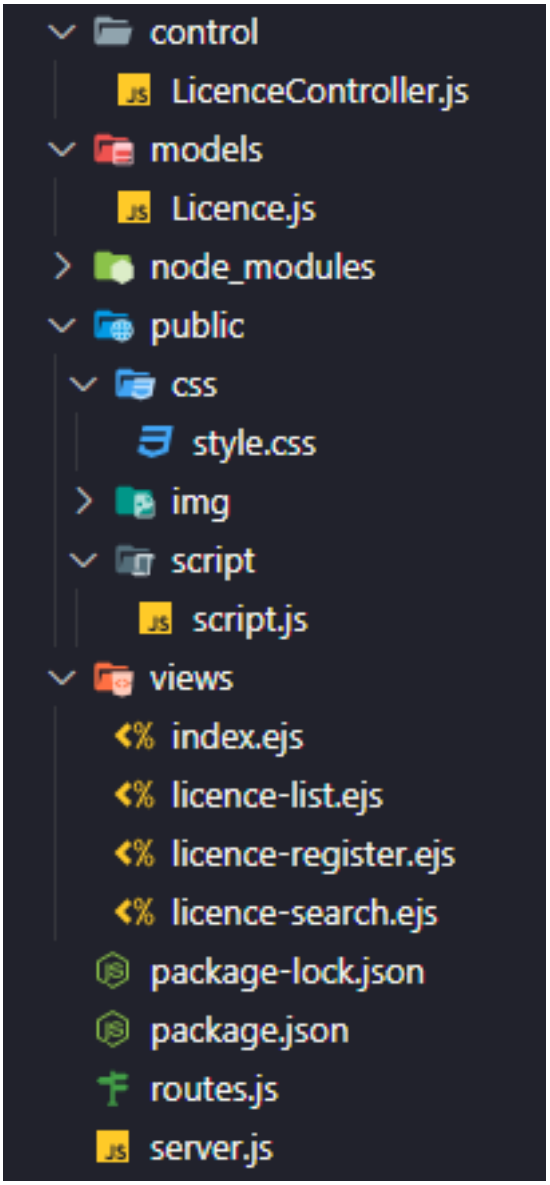


Fonte: elaborado pelo autor, 2022.
Nota: Desenvolvido no App *Lucidchart Diagrams* do *Google*.

4.3 INTERFACES

A interface de uma aplicação é basicamente tudo aquilo que é utilizado para interagir com a aplicação. (JOÃO, 2017). Foram desenvolvidos alguns protótipos de interfaces para o SiGLA, apresentadas entre a Figura 3 e a Figura 8.

Figura 2 - Padrão de arquitetura MVC adotado para o desenvolvimento do SiGLA.



Fonte: elaborado pelo autor, 2022.

As interfaces de login (Figura 3) e de busca por licença cadastrada (Figura 7) são acessadas através de telas que sobrepõem as demais sob comando de funções *JavaScript*. As demais telas de interface são constituídas por arquivos EJS (linguagem html) acessados através do arquivo *routes.js* (veja em *Arquitetura*, acima). Todos os arquivos EJS são constituídos por um cabeçalho (<header>), uma barra de navegação (<nav>) contendo botões (<button>) para transitar entre os arquivos e funções, uma sessão principal (<section>) contendo os elementos principais da página em questão, uma aba lateral (<aside>) onde ficam disponíveis *links* para *websites* de apoio e um rodapé (<footer>) com informações sobre a criação e desenvolvimento da aplicação.

Figura 3 - Protótipo de interface de login para acesso ao SiGLA.

Fonte: elaborado pelo autor, 2022.

Figura 4 - Protótipo de interface da tela inicial do SiGLA, apresentando informações sobre a aplicação e seu uso.

Fonte: elaborado pelo autor, 2022.

Figura 5 - Protótipo de interface da tela de cadastro de licenças no banco de dados do SiGLA.

Fonte: elaborado pelo autor, 2022.

Figura 6 - Protótipo de interface da tela com a listagem das licenças já cadastradas no sistema. O protótipo apresenta apenas três licenças cadastradas, mas a proposta é que nesta interface sejam apresentadas todas as licenças armazenadas no banco de dados.

Fonte: elaborado pelo autor, 2022.

Figura 7 - Protótipo de interface da tela de busca por licença cadastrada no SiGLA. A interface de busca também é utilizada nas opções para atualizar e deletar licenças já cadastradas.

Fonte: elaborado pelo autor, 2022.

Figura 8 - Protótipo de interface da tela de resultados da busca por licença cadastrada no SiGLA. O protótipo ilustra uma busca que resultou em apenas uma licença.

Fonte: elaborado pelo autor, 2022.

4.4 CODIFICAÇÃO JAVASCRIPT

A aplicação SiGLA faz uso de vários arquivos *JavaScript* com variadas funcionalidades. A seguir são apresentados alguns dos códigos a serem utilizados na aplicação. O Quadro 2 apresenta a codificação utilizada para gerar o servidor *Node.js* (arquivo *server.js* na Figura 2), enquanto o Quadro 3 exemplifica a criação da rota (arquivo *routes.js* na Figura 2) para acesso à página principal da aplicação (index.ejs).

Quadro 2 - Codificação *JavaScript* utilizada para a criação do servidor *Node.js*.

1	const http = require('http');
2	const express = require('express');
3	const path = require('path');
4	const app = express();
5	const routes = require('./routes');
6	const mongoose = require('mongoose');
7	const bodyParser = require('body-parser');
8	//Configuração de arquivos estáticos
9	app.use(express.static(path.join(__dirname, 'public')));
10	//Configuração das rotas
11	app.use(routes);
12	//Configuração do Mongoose (https://www.npmjs.com/package/mongoose)
13	mongoose.connect('mongodb://localhost/dbsigla',{
14	useNewUrlParser: true,
15	useUnifiedTopology: true

```
16  }).then(() => {
17      console.log('MongoDB conectado')
18  }).catch((err) => {
19      console.log('Houve um erro ao se conectar ao MongoDB: ' + err)
20  });
21  //Configuração do BodyParser para capturar informações do formulário
22  app.use(bodyParser.urlencoded({extended: true}));
23  app.use(bodyParser.json());
24  //Configuração do EJS
25  app.set('view engine', 'ejs');
26  //Rodando o servidor na porta 5000
27  http.createServer(app).listen(5000, () => console.log('Servidor rodando na porta 5000'));
```

Fonte: elaborado pelo autor, 2022.

Quadro 3 - Codificação *JavaScript* utilizada para a criação da rota principal (*index.ejs*).

```
1  const express = require('express');
2  const router = express.Router();
3  const LicenceController = require('./control/LicenceController')
4  router.get("/", function(req, res){
5      res.render(__dirname + "/views/index.ejs");
6  });
7  module.exports = router;
```

Fonte: elaborado pelo autor, 2022.

4.5 INTEGRAÇÃO COM O BANCO DE DADOS

A integração com o banco de dados MongoDB será executada através da ferramenta mongoose do *Node.js*¸ devidamente instalado na aplicação. A conexão inicial com o mongoose se dará juntamente com a inicialização do servidor (arquivo *server.js*) conforme exemplificado entre as linhas 12 e 20 do Quadro 2. O Quadro 4 ilustra o modelo de licença que será utilizado para o armazenamento dos dados, enquanto o Quadro 5 ilustra como o arquivo de controle (*LicenceController.js*) fará a captura das informações no formulário e armazenamento no banco de dados. Por fim, o Quadro 6 ilustra a rota de tipo “post” responsável por ativar essa função.

Quadro 4 - Codificação *JavaScript* utilizada para a criação do modelo de licença (arquivo *Licence.js*).

```
1  const mongoose = require('mongoose');
2  const Schema = mongoose.Schema;
3  const Licence = new Schema({
```

```
4     name: String,
5     cnpj: Number,
6     typeEnterprise: String,
7     number: String,
8     typeLicence: String,
9     state: String,
10    date: Date,
11    usersAlert: String
12 });
13 mongoose.model('licences', Licence);
14 module.exports = Licence;
```

Fonte: elaborado pelo autor, 2022.

Quadro 5 - Codificação *JavaScript* utilizada para o armazenamento das informações no banco dedados (arquivo *LicenceController.js*).

```
1  const mongoose = require('mongoose');
2  require('../models/Licence');
3  const Licence = mongoose.model('licences')
4  module.exports ={
5      createNew(req, res) {
6          const newLicence = {
7              name: req.body.name,
8              cnpj: req.body.cnpj,
9              typeEnterprise: req.body.typeEnterprise,
10             number: req.body.number,
11             typeLicence: req.body.typeLicence,
12             state: req.body.state,
13             date: req.body.date,
14             usersAlert: req.body.usersAlert
15         }
16         new Licence(newLicence).save().then(()=> {
17             console.log("Dados de Licença Salvos no Sistema!")
18         }).catch((err) => {
19             console.log("Erro ao cadastrar licença: " + err)
20         });
21         res.redirect('/');
22     })
23 }
```

Fonte: elaborado pelo autor, 2022.

Quadro 6 - Codificação *JavaScript* utilizada para a criação da rota principal (linhas 4 a 6) e a rota de armazenamento das informações no banco de dados (linha 7)

```
1 const express = require('express');
2 const router = express.Router();
3 const LicenceController = require('./control/LicenceController')
```

4	router.get("/", function(req, res){
5	res.render(__dirname + "/views/index.ejs");
6	});
7	router.post('/new', LicenceController.createNew);
8	module.exports = router;

Fonte: elaborado pelo autor, 2022.

5 RESULTADOS

Os protótipos de interface e as exemplificações de codificação apresentadas ao longo da análise de desenvolvimento deste projeto sugerem um funcionamento adequado do sistema, contemplando sua proposta. A aplicação ainda não foi completamente projetada e executada e ainda pode sofrer alterações ao longo de sua implementação, mas as análises sugerem que o SiGLA possa se tornar uma ferramenta efetiva no apoio ao gerenciamento de licenças ambientais.

5.1 TESTES E VALIDAÇÃO

Testes executados em uma versão preliminar do sistema apresentaram resultados satisfatórios em relação à proposta da aplicação, sobretudo quanto ao armazenamento, atualização, leitura e deleção dos dados no banco de dados (funções CRUD). Novos testes e validações ainda precisarão ser executados ao longo da implementação. Testes unitários serão aplicados ao longo da implementação e todos os formulários utilizados dentro da aplicação deverão passar por processos de validação, a fim de confirmar que as informações oferecidas no formulário são válidas.

Após a implementação, a aplicação também deverá ser testada por usuários da área de gestão de projetos ambientais a fim de se obter *feedbacks* sobre o uso e possíveis melhorias ao SiGLA através da experiência de usuário. Estes *feedbacks* podem influenciar melhorias tanto na interface quanto nas funcionalidades da

aplicação, sem se distanciar do escopo principal, que é o de gerenciamento de licenças ambientais.

5.2 PROPOSTAS DE MELHORIA

Embora o escopo do SiGLA seja apenas o gerenciamento de licenças ambientais através do controle dos prazos de vencimento, existem muitas oportunidades de melhoria e aperfeiçoamento para a aplicação.

Uma destas oportunidades de melhoria se relaciona a restrições do formulário de cadastro de licenças. Em sua proposta inicial o formulário se restringe aos três principais tipos de licenças ambientais (licença prévia, de implantação e de operação). Futuras versões do SiGLA podem contemplar outros tipos de licenças, como autorização de manejo de fauna, licença por compromisso, licença prévia para implantação e ampliação, entre outras.

Similarmente, a versão inicial do mesmo formulário também restringe os tipos de empreendimentos a minimamente quatro opções. A Resolução CONAMA nº 237, porém, menciona em seu Anexo 1 um total de 22 grandes grupos de atividades sujeitas ao licenciamento ambiental, os quais ainda podem ser desmembrados uma vasta lista de opções atualmente não contempladas pela aplicação.

Considerando-se o amplo uso dos *smartphones* nos tempos recentes, outra importante implementação é uma versão *mobile* do SiGLA. Considerando-se seu desenvolvimento com tecnologias *web*, seu uso pode ser convertido para os sistemas operacionais mais utilizados atualmente sem grandes dificuldades.

Por fim, uma expansão da aplicação, mas que foge um pouco de seu escopo atual, poderia oferecer opções para cadastrar outros tipos de documentos, como lembretes quanto à necessidade de protocolar documentos dentro de certo prazo (relatórios ambientais periódicos, por exemplo) e outras condicionantes da própria licença. Deste modo, o SiGLA tem o potencial de se expandir e se tornar uma plataforma de grande porte que auxilie todo o processo de licenciamento ambiental no Brasil, conectando empresas de licenciamento ambiental, seus clientes e as plataformas dos órgãos ambientais.

REFERÊNCIAS

ALVES, Niag de Souza. **MandaJobs:** sua plataforma de freelancers. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) – Faculdades e Escolas Técnicas QI, 19 p. 2020.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**. Disponível em: http://www.planalto.gov.br/ccivil_03/constituicao/constituicaocompilado.htm. Acesso em: 06 mar. 2022.

CONAMA. Resolução n. 237, de 19 de dezembro de 1997. **Diário Oficial da União**. Brasília, 22 dez. 1997. Seção 1, p. 30841-30843.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. 7.ed. São Paulo: Pearson Education do Brasil, 2018.

FEPAM. Portaria n. 60, de 24 de agosto de 2020. **Diário Oficial do Estado**. Porto Alegre, 24 ago. 2020.

FLATSCHART, Fábio. **HTML 5:** embarque imediato. Rio de Janeiro: Brasport, 2011.

HARUMI, Karina. **Utilizando a engine EJS para aplicações em Node.js**. 2021. Disponível em: https://forum.casadodesenvolvedor.com.br/topic/26-utilizando-a-engine-ejs-para-aplica%C3%A7%C3%B5es-em-nodejs/ . Acesso em: 07 abr. 2022.

JOÃO, Belmiro do Nascimento. **Usabilidade e interface homem-máquina**. São Paulo: Pearson Education, 2017.

LOZADA, Gisele. Introdução ao método de pesquisa. *In:* LOZADA, Gisele; NUNES, Karine da Silva (org.). **Metodologia Científica**. Porto Alegre: SAGAH, 2019.

LUIZTOOLS. **Boas práticas com MongoDB**. 2021. Disponível em: https://blog.umbler.com/br/boas-praticas-com-mongodb. Acesso em: 06 mar. 2022.

MEDEIROS, Luciano Frontino de. **Banco de dados**: princípios e práticas. Curitiba: InterSaberes, 2013.

MONGODB INC. **MongoDB Manual**. 2021. Disponível em: https://docs.mongodb.com/manual/. Acesso em 06 mar. 2022.

MORAIS, Izabelly Soares de; ZANIN, Aline. **Engenharia de software**. Porto Alegre: SAGAH, 2017.

NUNES, Karine da Silva. Tipos de pesquisa. *In:* LOZADA, Gisele; NUNES, Karine da Silva (org.). **Metodologia Científica**. Porto Alegre: SAGAH, 2019.

PAGE-JONES, Meilir. **Fundamentos do desenho orientado a objetos com UML**. São Paulo: MAKRON Books, 2001.

REINEHR, Sheila. **Engenharia de requisitos**. Porto Alegre: SAGAH, 2020.

SCHEMES, Taynara. **Regras de negócio em TI**: descubra esse importante conceito e entenda como aplicá-lo no seu negócio. 2020. Disponível em: https://conteudo.movidesk.com/regras-de-negocio-ti/. Acesso em: 14 mar. 2022.

SILVA, Júlia Marques Carvalho da; MILETTO, Evandro Manara. Comportamento com JavaScript. *In:* MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro (org.). **Desenvolvimento de software II**: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. p. 95-100.

SEGURADO, Valquiria Santos. **Projeto de interface com o usuário**. São Paulo: Pearson Education do Brasil, 2015.

SOMMERVILLE, Ian. **Engenharia de software**. 8. ed. São Paulo: Pearson, 2007.
TOTVS. **Node.js**: o que é, quais as características e vantagens? 2020. Disponível em: https://www.totvs.com/blog/developers/node-js. Acesso em: 02 mar.2022.

WIKIPEDIA. **Visual Studio Code**. Disponível em: https://pt.wikipedia.org/wiki/Visual_Studio_Code . Acesso em: 01 mar. 2022.

ZENKER, Aline Maciel. Padrões arquiteturais MVC, MVP e MVVM. *In:* ZENKER, Aline Maciel et al. (org.). **Arquitetura de sistemas**. Porto Alegre: SAGAH, 2019a.

ZENKER, Aline Maciel. Princípios de projeto e modelagem SOLID. *In:* ZENKER, Aline Maciel et al. (org.). **Arquitetura de sistemas**. Porto Alegre: SAGAH, 2019b.