UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
DEPARTAMENTO DE COMPUTAÇÃO

MAYCON RODRIGUES VIEIRA PEREIRA

SYSLYRA: DESENVOLVIMENTO DE UM SISTEMA DE APOIO ADMINISTRATIVO E OPERACIONAL A ESCOLA MUSICAL LYRA CARLOS GOMES

ALEGRE
2022

MAYCON RODRIGUES VIEIRA PEREIRA

SYSLYRA: DESENVOLVIMENTO DE UM SISTEMA DE APOIO ADMINISTRATIVO E OPERACIONAL A ESCOLA MUSICAL LYRA CARLOS GOMES

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo como requisito parcial para obtenção de grau no Curso de Sistemas de Informação. Orientador: Prof. M.Sc. Giuliano Prado de Morais Giglio.

ALEGRE
2022

MAYCON RODRIGUES VIEIRA PEREIRA

SYSLYRA: DESENVOLVIMENTO DE UM SISTEMA DE APOIO ADMINISTRATIVOE OPERACIONAL A ESCOLA MUSICAL LYRA CARLOS GOMES

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Aprovado em 09 de agosto de 2022.

COMISSÃO EXAMINADORA

Prof. MSc. Giuliano Prado de Morais Giglio
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Antônio Almeida de Barros Junior
Universidade Federal do Espírito Santo

Prof.ª. MSc. Valéria Alves da Silva
Universidade Federal do Espírito Santo

AGRADECIMENTOS

A Deus, pela minha vida e por ter me dado a oportunidade de ter força e saúde para chegar até aqui sempre me ajudando a ultrapassar todas as dificuldades encontradas ao longo do caminho.

Aos meus pais por todo o apoio, amor e confiança que depositaram em mim durante toda minha vida e principalmente durante todos esses anos da graduação estando longe de casa. Por toda a preocupação, pelas orações e por todo o cuidado que tiveram comigo mesmo que de longe.

Ao meu irmão e cunhada, casal que torceu por mim e me ajudou por um bom período a me estabilizar em Alegre, minha gratidão pelo amor e incentivo de vocês.

A todos os amigos e familiares que me deram apoio, ajuda e compreenderam os momentos de ausência para que eu conseguisse me dedicar nos estudos. Em especial aos amigos que a cidade Alegre me presenteou através dos grupos ABU Alegre, PG, Academic, Rangão e Vai trabalhar cabra, por todos os momentos incríveis vivenciados com cada um. Aos amigos do apartamento Rodney's Rep, pelo companheirismo, força e amizade durante os anos de moradia na república. A Igreja Batista Renovada de Alegre que me acolheu com muito carinho.

Ao professor Giuliano Prado de Morais Giglio pelos ensinamentos e orientações ao longo do desenvolvimento deste projeto. Aos professores do departamento de computação pelo conhecimento compartilhado ao longo dos anos.

A Universidade Federal do Espírito Santo, todo o seu corpo docente e servidores pelos serviços prestados e conhecimento compartilhado.

Obrigado a todos que fizeram parte dessa jornada junto comigo.

RESUMO

O presente trabalho tem como principal foco apresentar o projeto de desenvolvimento de um sistema de informação em uma instituição filantrópica da cidade de Alegre, a banda e escola de música Lyra Carlos Gomes, como parte do projeto de extensão UFES PRO TI. O objetivo geral é projetar e desenvolver um sistema web que permita fornecer um apoio na gestão administrativa e operacional da escola, de modo a auxiliar na execução e organizações dos processos, visando impactos positivos como agilidade e qualidade no tratamento dos dados e informações. Ao longo do trabalho foi abordado também sobre a informatização de entidades filantrópicas do terceiro setor, tendo em vista que essas entidades exercem um papel importante para a sociedade, contudo, pelo fato de se sustentarem de forma autônoma e voluntária em sua maioria, acabam não dispondo de recursos suficientes que permitam utilizar da tecnologia para auxiliar e facilitar o tratamento e organização dos dados, os quais possuem um papel importante para qualquer organização. Foi utilizado as tecnologias Node.js para a construção do projeto *back-end* e o *framework* de JavaScript, Vue.js, para o projeto *front-end* durante o desenvolvimento do projeto, além do banco de dados MySQL.

Palavras-chave: Sistemas de informação, terceiro setor, UFES PROTI, desenvolvimento web, Node.js, Vue.js.

ABSTRACT

The main focus of this work is to present the project to develop an information system in a philanthropic institution in the city of Alegre, the band and music school Lyra Carlos Gomes, as part of the UFES PRO TI extension project. The general is to design and develop a system that provides support in the administrative and operational management of the school, in order to assist in the execution and in the processes, aiming at the treatment of positive impacts such as agility and quality, in the treatment of data and information. Throughout the work, most of them were also important about the existence of transformation entities, having mostly philanthropists in the sector, however, because they see themselves as a way of transforming society, but having in their majority, the fact that see themselves as a form of existence and not available in their fact. of resources necessary for the treatment and technology to be important for the treatment of data, which have an important role for any organization. Node.js technologies were used to build the back-end project and the JavaScript framework, Vue.js, for the front-end project during project development, in addition to the MySQL database.

Keywords: Information systems, third sector, UFES PROTI, web development, Node.js, Vue.js.

LISTA DE FIGURAS

Figura 1 – Diagrama de casos de uso.....25
Figura 2 - Diagrama de classes.....26
Figura 3 - Modelo físico do banco de dados.....27
Figura 4 - Desenho da arquitetura.....34
Figura 5 - *Migration* de criação da tabela cursos.....35
Figura 6 - Estrutura de pastas.....36
Figura 7 - Configuração de conexão com o banco de dados.....37
Figura 8 - Configuração do .sequelizerc.....37
Figura 9 - Método *findAll*().....38
Figura 10 - Exemplo de arquivo de rotas.....39
Figura 11 - Método de atualização de curso no *Controller*.....39
Figura 12 - Arquivo de modelo de cursos.....40
Figura 13 - *Controller* de cursos.....41
Figura 14 - Requisição de *login* de usuário válido.....42
Figura 15 - *Middleware* de autenticação.....43
Figura 16 - Estrutura de pastas do projeto *front-end*.....45
Figura 17 - Arquivo de configuração do axios.....46
Figura 18 - Exemplo de requisição GET.....46
Figura 19 - Exemplo de requisição POST.....47
Figura 20 - Bloco *template* do componente App.vue.....48
Figura 21 - Exemplo de rotas nomeadas do projeto *front-end*.....48
Figura 22 - Pastas de *view*s.....49
Figura 23 - Tela de *login*.....50
Figura 24 - Tela inicial.....51
Figura 25 - Exemplo de tela de listagem de alunos.....52
Figura 26 - Tela de Cadastro de um novo aluno.....52
Figura 27 - Tela de edição de um aluno.....53
Figura 28 - Tela de listagem de turmas.....53
Figura 29 - Tela de cadastro de turmas.....54
Figura 30 - Gatilho *created* do componente de cadastro de turmas.....54
Figura 31 - Tela de registro de empréstimo de instrumento.....55
Figura 32 - Popup de exclusão de empréstimo de instrumento.....56

Figura 33 - Tela de Cadastro de Matrículas 57
Figura 34 - Tela de Cadastro de Matrículas quando selecionado um aluno 57
Figura 35 - Tela de relatório de aulas.....58
Figura 36 - Relatório de aulas em pdf 58
Figura 37 - Relatório de empréstimo 59
Figura 38 - Teste de requisição GET no *endpoint* de cursos por id 61
Figura 39 - Teste de requisição PUT no *endpoint* atualização de curso 62
Figura 40 - Fotos da implantação.....72

LISTA DE TABELAS

Tabela 1 - Tabela aluno 28
Tabela 2 - Tabela aula 28
Tabela 3 - Tabela curso 29
Tabela 4 - Tabela emprestimo.....29
Tabela 5 - Tabela endereco 29
Tabela 6 - Tabela escola.....30
Tabela 7 - Tabela estado 30
Tabela 8 - Tabela funcionário.....30
Tabela 9 - Tabela instrumento.....31
Tabela 10 - Tabela matrícula.....31
Tabela 11 - Tabela responsavel.....31
Tabela 12 - Tabela turma 31
Tabela 13 - Tabela usuário.....32
Tabela 14 - Tabela de inspeções. 60
Tabela 15 - Plano de testes.....63
Tabela 16 - Teste de usabilidade. 67

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

CRUD – *Create, Read, Update, Delete*

HMAC - *Hash-based Message Authentication Code*

HTTP – *Hypertext Transfer Protocol*

I/O – *Input/Output*

JWT – *Json Web Token*

MDN – *Mozilla Developer Network*

MIT - *Massachusetts Institute of Technology*

ORM – *Object Relational Mapper*

OSCIP – Organização da Sociedade Civil de Interesse Público

REST – *Representational state transfer*

RFC – *Request for Comments*

RSA - *Rivest-Shamir-Adleman*

SFC – *Single File Component*

SGBD – Sistema de Gerenciamento de Banco de Dados

SPA – *Single Page Application*

SQL – *Structured Query Language*

UI – *User Interface*

UML – *Unified Modeling Language*

XP – *Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO 12

1.1 O PROBLEMA E SUA IMPORTÂNCIA 13

1.2 OBJETIVOS 13

1.2.1 Objetivo geral 14

1.2.2 Objetivos específicos 14

2 REVISÃO DE LITERATURA 15

2.1 SISTEMAS DE INFORMAÇÃO 15

2.2 TERCEIRO SETOR 16

2.2.2 OSCIP 16

2.3 LYRA CARLOS GOMES 16

2.4 SISTEMAS DE INFORMAÇÃO E O TERCEIRO SETOR 17

2.5 SISTEMAS DE INFORMAÇÃO DE GESTÃO ESCOLAR 18

2.6 VUE.JS 19

2.7 NODE.JS 20

2.8 MYSQL 20

3 METODOLOGIA 21

3.1 METODOLOGIA DE DESENVOLVIMENTO 21

3.2 ANÁLISE DE REQUISITOS 23

3.2.1 Requisitos Funcionais 23

3.3 MODELAGEM DO SISTEMA 24

3.3.1 Diagrama de Casos de Uso 25

3.3.2 Diagrama de Classes 26

3.4 MODELAGEM DE BANCO DE DADOS 27

3.5 DESCRIÇÃO DAS TABELAS DO BANCO 28

3.6 A APLICAÇÃO 32

4 DESENVOLVIMENTO 34

4.1 ARQUITETURA 34
4.2 BACK-END 35
4.2.1 Conexão com o banco de dados 37
4.2.2 Endpoints 38
4.2.3 Models 39
4.2.4 Controllers 40
4.2.5 Autenticação 42
4.3 FRONT-END 44
4.3.1 Comunicação com o servidor 45
4.3.2 Router 47
4.3.3 Views 49
4.4 INTERFACES 50
4.4.1 Tela de Login 50
4.4.2 Tela Inicial 51
4.4.3 Tela de Cadastro de Alunos 51
4.4.4 Tela de Cadastro de Turmas 53
4.4.5 Tela de Registro de empréstimos de instrumentos 55
4.4.6 Tela de Cadastro de Matrículas 56
4.4.7 Tela de Relatórios 57
5 TESTES E VALIDAÇÕES 60
5.1 INSPEÇÕES 60
5.2 TESTE DE DESENVOLVIMENTO 61
5.3 TESTE DE USUÁRIO 65
5.4 TESTE DE USABILIDADE 67
6 CONSIDERAÇÕES FINAIS E CONCLUSÕES 71
REFERÊNCIAS 74

1 INTRODUÇÃO

É perceptível que o uso de tecnologias de informação ao longo dos anos vem avançando e se mantendo em constante processo de evolução, impactando diretamente na forma como a sociedade lida com o acesso às informações. Segundo Batista (2012) “vivemos em plena revolução tecnológica e é fundamental que a nossa economia e a sociedade se comportem de maneira semelhante às demais revoluções que tiveram impactos na sociedade”.

Faz parte dessa ascensão tecnológica o uso de sistemas de informação que são responsáveis pelo armazenamento, tratamento e disponibilização de dados que podem ser considerados como um dos maiores bens de empresas e organizações, uma vez que a partir do tratamento e análise dos dados é possível obter informações valiosas e a partir delas, fazer projeções que podem auxiliar na tomada de decisões, através da utilização de ferramentas e técnicas de inteligência de negócios. Assim, compreende-se que o valor da informação está diretamente ligado em como ela ajuda os tomadores de decisões a alcançar os objetivos da organização, uma vez que as informações valiosas possuem a capacidade de auxiliar as pessoas em suas organizações a realizarem as tarefas de forma mais eficiente e eficaz (STAIR, REYNOLDS, 2016).

Porém, mesmo nesse cenário de crescente evolução da tecnologia, ainda é possível identificar alguns desafios em instituições filantrópicas, que não possuem a mesma realidade que as grandes empresas e grandes organizações possuem em se manter de modo eficiente e sustentável, tendo essas entidades, a necessidade de depender do apoio de forças políticas e econômicas e além disso trabalhar com empecilhos na comunicação e gestão da organização, muitas vezes presas à procedimentos administrativos ultrapassados e limitados. Diante disso uma forma de modernizar a gestão tornando as entidades mais conectadas à realidade da atual sociedade é apostar no investimento e no planejamento de recursos tecnológicos que permitam melhorar o desempenho das atividades exercidas por essas organizações e ampliar a atuação delas em meio à sociedade (SALES, 2017).

Este trabalho surge então como parte do projeto de extensão “UFES PRO-TI: Projeto de Assessoria Socialmente Responsável em Tecnologia da Informação” que tem como objetivo principal atender a comunidade em geral da região sul do estado

do Espírito Santo e região do Caparaó com a prestação de serviços na área de computação a instituições filantrópicas, através do desenvolvimento e implantação de sistemas de informação, que atendam às demandas internas administrativas destas instituições, promovendo automatização, flexibilidade e qualidade no tratamento das informações (UFES PROTI, 2021).

A escola de música Lyra Carlos Gomes localizada em Alegre/ES é classificada como uma organização da sociedade civil de interesse público (OSCIP) que em 2009 foi reconhecida como entidade de Utilidade Pública Municipal pela Lei 3040/2009 e como entidade cultural pelo Conselho Estadual da Cultura, em dezembro de 2009 e foi objeto de estudo para realização deste trabalho (LYRA CARLOS GOMES, 2020).

1.1 O PROBLEMA E SUA IMPORTÂNCIA

A escola Lyra Carlos Gomes foi objeto de estudo, devido às suas características assistenciais e condições estruturais. Após entrevista com o diretor musical e conhecimento prévio do aluno foi identificado que a maioria dos processos utilizados na gestão, tanto no âmbito operacional quanto administrativo, são processos manuais em sua maioria, pois há também a utilização de recursos informáticos como planilhas para controle de empréstimos de instrumentos e formulários online para inscrições de alunos. Entretanto, mesmo utilizando alguns recursos computacionais, a escola não possui com essas ferramentas uma integração dos dados, ficando as informações dispersas e separadas em lugares diferentes, tanto digitalmente quanto manualmente, o que dificulta o controle e organização dos processos da direção da escola.

Tendo em vista a escola Lyra Carlos Gomes, como instituição filantrópica, o desenvolvimento de um sistema que auxilie nas demandas internas desta instituição pode representar uma evolução nas formas como os processos administrativos e operacionais são executados, além de gerar uma maior facilidade no manuseio e obtenção de informações com agilidade e segurança.

1.2 OBJETIVOS

A seguir serão apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.2.1 Objetivo geral

Este trabalho tem como objetivo o desenvolvimento de um sistema web que permita fornecer um apoio na gestão administrativa e operacional da escola de música Lyra Carlos Gomes, que auxilie diretamente na maneira como os processos são executados, de forma a promover agilidade, precisão e qualidade no tratamento das informações.

1.2.2 Objetivos específicos

De modo a alcançar o objetivo geral deste trabalho, destacam-se como objetivos específicos:

1. Conceituar sistemas de informação e apresentar a escola Lyra Carlos Gomes, como instituição filantrópica e objeto de estudo para este trabalho.
2. Fazer a análise e levantamento de requisitos para auxiliar no projeto do sistema.
3. Modelar e criar o banco de dados.
4. Desenvolver o sistema utilizando boas práticas de programação.
5. Implantar o sistema e disponibilizar para utilização da Lyra Carlos Gomes.

2 REVISÃO DE LITERATURA

Nesta seção serão apresentados os conceitos das áreas que envolvem o presente trabalho, além de tecnologias e informações que serão utilizadas para o seu desenvolvimento.

2.1 SISTEMAS DE INFORMAÇÃO

A utilização de tecnologia de informação dentro das organizações se tornou notória ao decorrer dos anos. O uso dessas tecnologias pode ser considerado como uma vantagem competitiva para as organizações que buscam por uma melhor gestão de tempo e organização de seus processos internos.

Uma dessas tecnologias, se trata dos sistemas de informação que Stair e Reynolds (2015) definem como um conjunto de elementos ou componentes inter-relacionados que coleta, manipula, armazena e dissemina dados e informações, além de fornecer reação corretiva para alcançar um objetivo. Segundo Laudon e Laudon (2010) os sistemas de informação além de dar apoio à tomada de decisões, à coordenação e ao controle, também auxiliam os gerentes e trabalhadores na análise de problemas, visualização de assuntos complexos e criação de novos produtos. Batista (2012) define sistemas de informação como todo e qualquer sistema que apresente dados ou informação de entrada que tenha por finalidade a geração de informações de saída para suprir determinadas necessidades.

Em conformidade com os conceitos já apresentados, existem três atividades básicas em um sistema de informação que geram resultados para as organizações que são as atividades de entrada, processamento e saída. A entrada captura ou coleta dados brutos de dentro da organização ou se seu ambiente externo, o processamento converte esses dados em uma forma mais significativa e a saída transfere as informações processada às pessoas que as utilizarão. Os sistemas de informação também precisam de um *feedback,* que é uma resposta a uma ação adotada à determinados membros de uma organização para auxiliar na avaliação ou na correção do estágio de entrada (LAUDON; LAUDON, 2015).

2.2 TERCEIRO SETOR

As organizações de terceiro setor se destacam por atividades de caráter social a fim de oferecer ações importantes para a sociedade. Kurz (1997, pag. 152) define o terceiro setor como sendo um “composto da união de inúmeros agrupamentos voluntários, destinados a conter a miséria social e barrar a destruição ecológica. A maioria desses grupos dá, grande valor à administração autônoma.”

Costa Júnior (1998) define o terceiro setor como sendo as organizações privadas que são públicas por suas finalidades que não se encaixam nem no primeiro setor que é o setor estatal, nem no segundo setor que é o setor privado. O autor cita como sendo organizações do terceiro setor as organizações não-governamentais, institutos, fundações, entidades de classe, associações profissionais, movimentos sociais dos mais variados e outras entidades que atuam nas mais diversas áreas sociais. Segundo França Filho (2002) o termo terceiro setor foi herdado de um tradição anglo-saxônica impregnada praticamente pela ideia de filantropia, de forma que essa abordagem identifica o terceiro setor ao universo das organizações sem fins lucrativos. Levando em consideração as definições já elaboradas, entende-se que o conceito de terceiro setor está intrinsecamente ligado a atividades em prol da sociedade que são executadas por diferentes tipos de associações e entidades que não visam lucro.

2.2.2 OSCIP

Instaurada a partir da Lei 9.970/99, uma Organização da Sociedade Civil de Interesse Público (OSCIP) é uma qualificação que pessoas jurídicas de direito privado sem fins lucrativos podem receber. Segundo informações do SEBRAE, 2019:

> Uma **OSCIP é uma qualificação jurídica atribuída a diferentes tipos de entidades privadas atuando em áreas típicas do setor público com interesse social, que podem ser financiadas pelo Estado ou pela iniciativa privada sem fins lucrativos (negrito do autor.)**. Ou seja, as entidades típicas do terceiro setor (SEBRAE, 2019).

2.3 LYRA CARLOS GOMES

Fundada em 29 de maio de 1938 a banda de música Lyra Carlos Gomes teve como primeiro presidente o Coronel Sebastião Monteiro da Gama e recebeu os

instrumentos e acervo da extinta sociedade “Euterpe Santo Antônio” além da incorporação dos músicos desta. Foram integrados também à banda músicos da Sociedade Musical Nova Aurora e da Lyra Alegrense, bandas que antecederam a Lyra Carlos Gomes. A entidade foi reconhecida como de Utilidade Pública Municipal pela Lei 3040/2009 e como entidade cultural pelo Conselho Estadual de Cultura – CEC, cfe. Registro 03/2009 (LYRA CARLOS GOMES, 2020).

A Lyra Carlos Gomes possui um projeto de educação musical que foi desenvolvido pelo Professor Elivelton Silvestre, que consiste no processo de oferecimento de ensino musical com inserção nas escolas públicas da cidade, buscando musicalizar não só alunos do ensino fundamental, mas também pessoas interessadas em aprender algum dos instrumentos que tenham interesse em se integrar à banda. O objetivo do projeto é utilizar o ensino da música como uma forma de inclusão e transformação sociocultural de comunidades, ofertando uma educação musical de qualidade que permita crescimento pessoal e a oportunidade de profissionalização.

Atualmente a escola e banda musical se encontra localizada no endereço Rua Monsenhor Pavesi, nº 116, Centro, Alegre – ES, e tem como diretor musical e responsável pelo projeto, o professor Elivelton Silvestre.

2.4 SISTEMAS DE INFORMAÇÃO E O TERCEIRO SETOR

O avanço tecnológico, a globalização e a mutabilidade dos ambientes nos quais uma organização está inserida reflete na importância que os sistemas de informação têm como uma ferramenta no processo de gestão, não se limitando apenas à grandes empresas, mas também abrangendo organizações do terceiro setor. Segundo Audy (1999) as organizações para continuar crescendo ou até mesmo continuar sobrevivendo, necessitam de mudanças pois os paradigmas políticos, econômicos e sociais estão mudando e os modelos de organização empresarial estão sendo fortemente questionados. Entretanto o autor também ressalta que a construção de sistemas de informação e o processo de informatização das organizações tomam muito tempo e têm altos custos.

No que diz respeito às entidades do terceiro setor, estes custos podem ser considerados ainda maiores, uma vez que em paralelo ao avanço da tecnologia, muitas dessas instituições funcionam de forma assistencial e voluntariada e,

consequentemente, não possuem os recursos necessários para a aplicação de tecnologia da informação dentro dessas organizações.

A tecnologia da informação proporciona a disponibilização de recursos tecnológicos necessários que promovem transformação de atividades e seus procedimentos de modo que se torna possível encontrar erros, desperdício de tempo ou de qualidade, sendo que através de uma reformulação dessas atividades alocando recursos tecnológicos é possível alcançar melhorias drásticas de custos, qualidade, atendimento e agilidade (TEÓFILO; DE FREITAS, 2007).

Entretanto segundo Dufloth (2005), o nível de integração proporcionado pela tecnologia pode não ser tão homogêneo a ponto de garantir um fluxo informacional uniforme, que consiga criar condições equilibradas de acesso e obtenção de informações para todas as organizações ou entidades do terceiro setor, sendo então as redes informais, como as conexões interpessoais promovidas através de encontros da área, cursos, congressos ou outras entidades promotoras, alternativas que também são utilizadas para promover essa integração.

Nesse sentido surge então a proposta deste trabalho que é, através de uma integração entre a universidade e uma entidade social de ensino musical, promover o desenvolvimento de um sistema de informação que consiga por meio de uma aplicação, proporcionar à organização agilização de processos internos operacionais e de gestão da escola, de modo a facilitar a dinâmica de funcionamento, gestão, organização e disponibilização das informações.

2.5 SISTEMAS DE INFORMAÇÃO DE GESTÃO ESCOLAR

Existe disponíveis no mercado vários *softwares* de gestão escolar que realiza o controle de vários processos de uma escola, a fim de centralizar as informações em um só lugar para as tornar facilmente acessível de modo organizado e centralizado. Nestes tipos de sistemas é possível integrar diferentes áreas de uma escola como por exemplo secretaria, financeiro, pedagógico e administrativo centralizando e organizando as informações dentro de uma plataforma, tornando-se possível por exemplo otimizar tempo e material, realizar um acompanhamento escolar e acompanhar os trabalhos dos professores.

Alguns *softwares* que podem ser citados e que são conhecidos por fornecer esses tipos de serviços são o Sponte (https://www.sponte.com.br/), WPensar

(https://wpensar.com.br/), Galileu (https://www.sistemagalileu.com.br/), Eduxe (http://eduxe.com.br/), entre outros (XAVIER, 2022). Entretanto a maioria desses *softwares* apesar de bem completos são *softwares* proprietários que ofertam apenas com a compra, a licença pelo uso do *software*, o treinamento e suporte. Como opção gratuita, foi encontrado o *software* UnifiedTransform (https://changeweb.github.io/Unifiedtransform/) que é um *software* de código aberto, onde se é possível baixar o código fonte e editá-lo para personalização.

Entretanto, tendo em vista os primeiros *software*s citados, estes não se enquadram como uma boa opção a ser escolhida pela Lyra Carlos Gomes, pelo fato de serem sistemas pagos, que não compensam o alto investimento tendo em vista às dificuldades enfrentadas pela instituição que recebe apoio voluntariado para se manter, além de doações e contribuições. Já o *software open-source* citado, é um sistema que poderia atender à escola, porém da forma em que é disponibilizado precisaria sofrer alterações de código para tradução da língua e retirada de opções, uma vez que o sistema oferta uma gama maior de funcionalidades como a criação de semestres, seções, exames e notícias, por exemplo, que são funções que a escola para o seu estado atual não pretende utilizar.

2.6 VUE.JS

O Vue.js é um *framework* progressivo do JavaScript de código aberto utilizado para desenvolvimento *front-end* em aplicações web, lançado em fevereiro de 2014 por Evan You, desenvolvedor que trabalhava em um dos projetos do Google Creative Labs. Segundo Incau (2017), o Vue.js é uma biblioteca JavaScript para o desenvolvimento de componentes reativos para web que ganhou muita visibilidade no mercado após ser adotado como padrão pelo Laravel, sendo uma característica do Vue.js, a possibilidade de ele ser utilizado em qualquer projeto que possua *front-end*, independente da linguagem de programação escolhida.

Para desenvolvimento deste projeto, foi escolhido o *framework* Vue.js como ferramenta para construção do *front-end* da aplicação, tendo em vista a rápida curva de aprendizado que o *framework* possui e a facilidade de integração com APIs, tornando-se possível através de uma biblioteca HTTP, implementar as devidas requisições necessárias para comunicação entre as camadas *front-end* e *back-end*.

2.7 NODE.JS

O Node é uma plataforma para o desenvolvimento de aplicações web que utiliza o JavaScript Engine V8 do Google. Segundo Pereira (2014), o Node é uma plataforma altamente escalável e de baixo nível, cuja linguagem de programação é o JavaScript e a programação se dá diretamente com diversos protocolos de rede e internet ou utilizando bibliotecas que usem recursos do sistema operacional.

Por se tratar de uma tecnologia voltada para a programação de aplicações em rede, o Node.js foi construído para ser orientado a eventos e I/O não bloqueante, de modo que a entrada e a saída de dados não bloqueiam os processos de execução das aplicações (CARVALHO, 2015). Segundo Pereira (2014), a arquitetura não bloqueante apresenta uma boa performance com consumo de memória e utiliza ao máximo e de forma eficiente o poder de processamento dos servidores, principalmente em sistemas que produzem uma alta carga de processamento, de modo que os usuários não tenham que esperar muito tempo pelos resultados dos processos e não sofram *dead-locks* no sistema, que ocorre quando um programa fica bloqueando o outro e a execução não consegue ser processada.

2.8 MYSQL

Um sistema de gerenciamento de banco de dados (SGBD) é um sistema responsável por acessar e administrar dados em diferentes fontes. Silberschatz, Korth e Sudarshan (2020) definem um SGBD como uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados, sendo a coleção de dados conhecida como banco de dados que contém informações relevantes para uma organização. “O programa MySQL é um sistema de gerenciamento de banco de dados relacional que utiliza a linguagem de consulta estruturada SQL como interface de acesso e extração de informações do banco de dados em uso” (MANZANO, 2011).

O MySQL foi o SGBD utilizado no desenvolvimento deste projeto se responsabilizando pelo armazenamento e manipulação dos dados tendo em vista ser um produto que possui um servidor confiável e de fácil utilização.

3 METODOLOGIA

Considerando o objetivo central do trabalho, que visa o desenvolvimento de um sistema de informação para apoio administrativo e operacional da escola de música Lyra Carlos Gomes, foi feito um estudo de viabilidade para conhecer melhor a escola, através de entrevistas com o diretor musical dela, a fim de entender os principais problemas que motivam o desenvolvimento do sistema e levantar seus requisitos preliminares. A partir destes últimos, foram identificados os métodos e ferramentas a serem utilizados para o desenvolvimento deste trabalho, os quais serão descritos nos tópicos a seguir.

3.1 METODOLOGIA DE DESENVOLVIMENTO

Para a o desenvolvimento do sistema foi utilizada a metodologia de desenvolvimento XP (*Extreme Programming)* que pode ser traduzido como Programação Extrema.

Sommerville (2007) descreve que a metodologia XP é uma das metodologias ágeis que teve seu nome cunhado por Kent Beck pois a abordagem foi desenvolvida pelo avanço de reconhecida boa prática, tal como o desenvolvimento iterativo e o envolvimento do cliente em níveis extremos. O XP é composto por práticas que se enquadram nos princípios de métodos ágeis:

> 1. O desenvolvimento incremental é apoiado por pequenos e frequentes *releases* do sistema e por uma abordagem de descrição de requisitos baseada nas histórias ou cenários do cliente que podem ser a base para o planejamento do processo.
> 2. O envolvimento do cliente é apoiado pelo engajamento em tempo integral deste na equipe de desenvolvimento. O representante do cliente faz parte do desenvolvimento e é responsável pela definição de testes de aceitação do tema. (SOMMERVILLE, 2007, p. 264)

De acordo com Beck (2000, apud SATO, 2007, p. 13) as principais características do XP é ser um método leve onde o time só deve fazer o necessário para trazer o valor para o cliente; ser um método que enfatiza o desenvolvimento de *software*; ser um método que funciona para times de qualquer tamanho e ser um método que se adapta a requisitos vagos ou em constante mudança.

O XP possui uma proposta baseada em cinco valores que norteiam a aplicação das práticas e princípios da metodologia. Sato (2007) apresenta esses cinco valores

em seu trabalho sendo o primeiro valor, o da Comunicação, que é evidenciada em muitas das práticas onde a equipe funciona como um único time que trabalha em uma área de trabalho informativa, onde todos possam estar juntos a fim de facilitar a comunicação da equipe. Além disso, a presença e o envolvimento real com o cliente e o desenvolvimento em pares também são algumas das práticas que fortalecem a comunicação. O segundo valor é a Simplicidade que prega a busca pela solução mais simples para resolver os problemas, de forma a promover a preocupação com o que é mais simples e evitando desperdícios com soluções genéricas visando modificações futuras.

Ainda de acordo com o autor citado, o *Feedback* é o terceiro valor, que na metodologia promove ciclos curtos e constantes em diversos aspectos durante o desenvolvimento de *software*, sendo uma parte importante da comunicação e da simplicidade. A Coragem é o quarto valor que é descrita em XP como a ação tomada diante do medo, porém em um cenário onde este valor é seguido junto aos demais, de forma com que o time tenha coragem, por exemplo, para manter a ênfase na simplicidade, permitir ao cliente priorizar funcionalidades e adotar um ritmo sustentável. O quinto e último valor apresentado é o Respeito que é enfatizado na segunda edição de XP, servindo de base para os outros quatro valores, onde é ressaltado que a excelência no desenvolvimento de *software* depende das pessoas e para extrair o máximo do potencial de cada indivíduo, é preciso haver o respeito interpessoal para atingir tal fim.

As atividades que foram desenvolvidas utilizando a metodologia XP, planejamento, projeto, codificação e testes, seguem detalhadas:

- **Planejamento**: atividade em que é feito o levantamento e análise de requisitos que neste trabalho se deu através de entrevistas virtuais a fim de organizar e documentar as principais funcionalidades pretendidas pela escola de música.

- **Projeto**: atividade em que é tomada a decisão sobre a organização lógica do sistema a ser desenvolvido. Para essa organização é feita a confecção dos diagramas dispostos no trabalho, que utilizam a linguagem de modelagem unificada (UML) e que foram desenvolvidas utilizando as ferramentas *Astah Community,* para os diagramas de casos de uso e diagrama de classes, *MySQL Workbench 8.0 CE*, para o diagrama entidade relacionamento do modelo lógico e a ferramenta brModelo para o diagrama entidade relacionamento do modelo conceitual.

- **Codificação**: atividade em que é iniciada a implementação do sistema, de acordo com o projeto e o planejamento previamente feitos. Para desenvolvimento desta atividade as principais ferramenta utilizadas foram o editor de códigos *Visual Studio Code*, Postman como ferramenta para teste de API e o navegador Microsoft Edge.
- **Testes**: atividade em que é feita a validação do que foi codificado nas tarefas sendo uma atividade fundamental para que com a produção de novos incrementos, os erros não sejam introduzidos.

3.2 ANÁLISE DE REQUISITOS

A atividade de análise de requisitos é uma das principais atividades do processo de desenvolvimento de *software* pois ela tem como objetivo entender e detalhar os requisitos do sistema. Segundo Hirama (2003), com base nas definições da atividade de Engenharia de Sistemas, nesta atividade deve-se compreender o domínio da informação, a funcionalidade requerida e as interfaces exigidas.

A partir do levantamento e análise de requisitos que deverão ser atendidos no desenvolvimento deste trabalho, decidiu-se pela implementação de um sistema web que seja capaz de ser utilizado através de navegadores da internet em diferentes plataformas.

As principais atividades que o SysLyra deverá oferecer são os cadastros como os de alunos, funcionários, cursos, turmas, instrumentos entre outros, além de emissão de relatórios para controle de empréstimo de instrumentos, relatórios de alunos cadastrados e de aulas ministradas.

3.2.1 Requisitos Funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve fazer. Eles dependem do tipo de *software* a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos (SOMERVILLE, 2007). Os requisitos funcionais coletados durante o levantamento de requisitos foram os seguintes:

RF001 - Manter usuário: deverá permitir ao usuário administrador a criação, edição, consulta e deleção um determinado usuário do sistema.

RF002 - Manter funcionário: deverá permitir à direção da escola, a criação, edição, consulta ou exclusão de um funcionário.

RF003 - Manter curso: deverá permitir à direção da escola, a criação, edição e deleção um determinado curso.

RF004 - Manter alunos: deverá permitir à direção da escola, a criação, edição e exclusão de registros de alunos.

RF005 - Manter turmas: deverá permitir à direção da escola, a criação, edição, consulta e exclusão de turmas.

RF006 - Manter matrícula: deverá permitir à direção da escola, a criação, edição, consulta e exclusão das matrículas de alunos.

RF007 - Manter aula: deverá permitir aos professores o lançamento, edição, consulta e exclusão de aulas no sistema.

RF008 – Linkar repositório drive em aula: deverá permitir ao usuário linkar uma *url* de um repositório de armazenamento em nuvem, como o Google Drive.

RF009 - Manter instrumento musical: permite à direção da escola, a criação, edição, consulta e exclusão de registros de instrumentos musicais da escola.

RF010 - Manter empréstimo: permite à direção da escola, a criação, edição, consulta e exclusão de registros de empréstimos de instrumentos aos alunos.

RF011 - Emitir relatório de frequência: permitirá a visualização e emissão de relatórios de frequência de alunos em uma determinada turma.

RF012 - Emitir relatório de aula: permitirá a visualização e emissão de relatórios contendo as informações da aula e a descrição do conteúdo ministrado nelas.

RF013 - Emitir relatório de avaliação: permitirá a emissão relatórios das avaliações aplicadas em uma determinada turma.

RF014 - Emitir relatório para controle de empréstimo: o sistema permitirá a visualização e a emissão de um relatório de empréstimos de instrumentos para controle de empréstimo, tanto geral quanto por aluno.

3.3 MODELAGEM DO SISTEMA

A UML (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) é uma linguagem visual utilizada para modelar *software*s baseados no paradigma de orientação a objetos, sendo esta linguagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. A modelagem de *software* então implica em criar

modelos que capturam uma visão de um sistema físico, abstraindo o sistema com um determinado propósito, como o de descrever aspectos estruturais ou comportamentais do *software* (GUEDES, 2011).

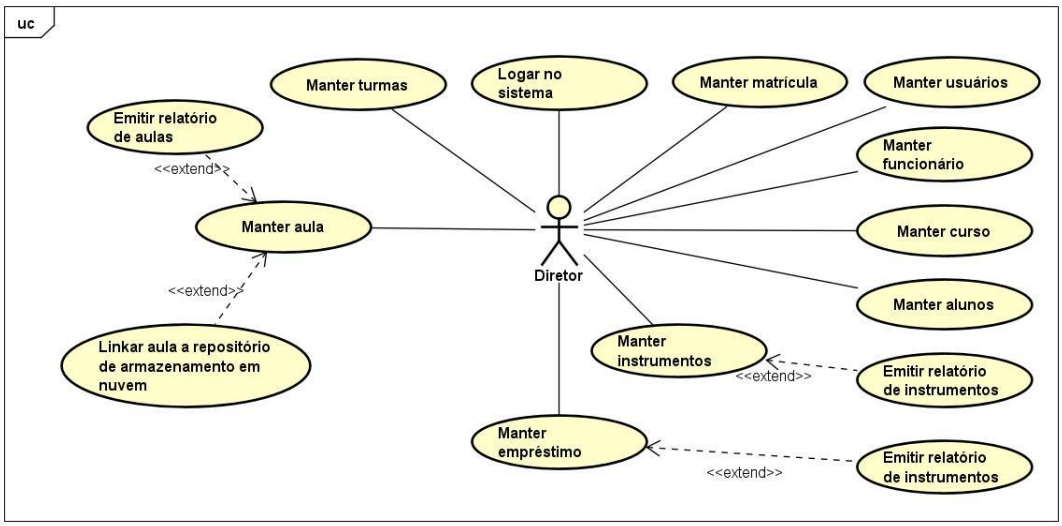
Neste trabalho foram elaborados os diagramas de classe e os diagramas de caso de uso que são apresentados nos tópicos seguintes.

3.3.1 Diagrama de Casos de Uso

Guedes (2011) apresenta o diagrama de casos de uso como sendo o diagrama que procura possibilitar a compreensão do comportamento externo do sistema considerando as funcionalidades por ele oferecidas, tentando apresentar o sistema por meio da perspectiva dos usuários. O diagrama em si é composto por dois itens principais que são os atores, que representam os papéis dos diversos usuários que poderão utilizar o sistema de alguma maneira e os casos de usos que capturam os requisitos funcionais, podendo se referir a serviços, tarefas ou funcionalidades necessárias ao *software*.

No diagrama de casos de uso, os atores são representados por bonecos magros e os casos de uso são representados por elipses que contém uma descrição sucinta da funcionalidade que o caso de uso se refere. Para definir as funcionalidades que cada tipo de usuário poderá executar no sistema proposto, foi elaborado o diagrama de casos de uso que é apresentado na Figura 1.

Figura 1 – Diagrama de casos de uso



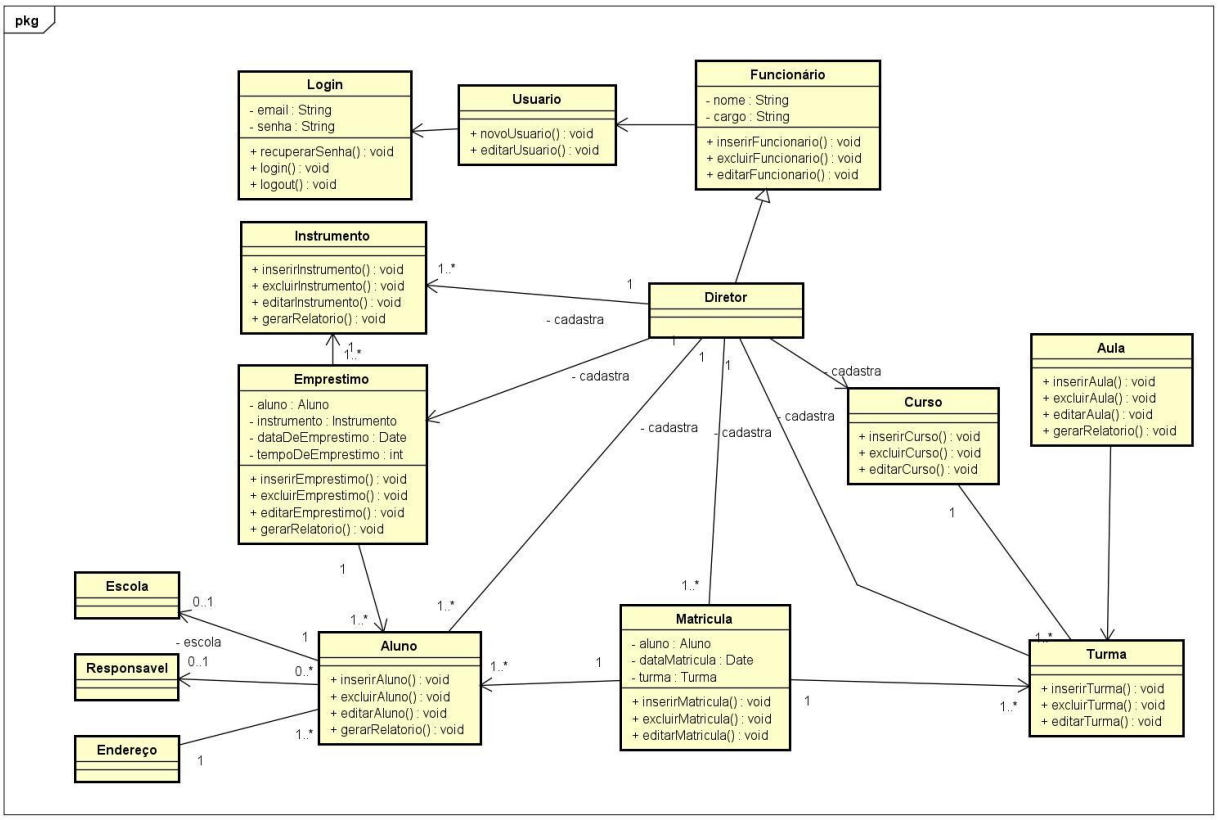
Fonte: Do autor.

3.3.2 Diagrama de Classes

O diagrama de classes é um dos diagramas da UML comumente mais utilizado que serve de base para a maioria dos demais diagramas e tem a função de definir a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem e de estabelecer como as classes se relacionam e trocam informações entre si (GUEDES, 2011).

O diagrama de classes construído para a aplicação web do SysLyra está apresentado na Figura 2, ilustrando as classes de domínio da aplicação, que seguem representados pelos retângulos, com seus atributos e métodos principais, e os devidos tipos de associações existentes entre as classes que são representados pelas linhas.

Figura 2 - Diagrama de classes



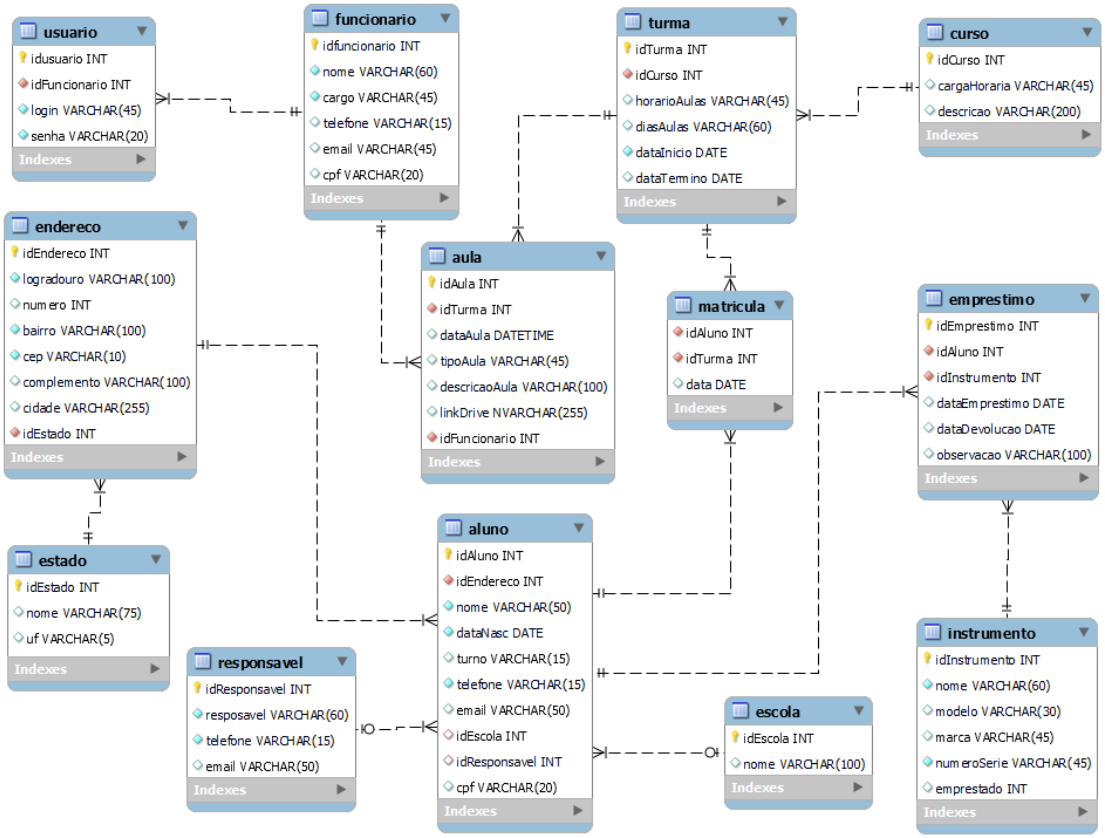
Fonte: Do autor.

3.4 MODELAGEM DE BANCO DE DADOS

Um modelo de dados, conjunto de conceitos que podem ser usados para descrever a estrutura de um banco de dados, fornece o significado necessário para permitir a abstração dos dados (HEUSER, 2004). Portanto através da modelagem torna-se possível colocar em forma a abstração das ideias e dos conjuntos de dados que serão necessárias para ser armazenadas e manipuladas em um banco de dados.

O diagrama entidade-relacionamento do modelo conceitual, segundo Alves (2014), representa a realidade por meio de uma visão global e genérica dos dados e seus relacionamentos, tendo como objetivo conter as informações dessa realidade que serão armazenadas no banco de dados, mas sem retratar particularidades do banco de dados que será utilizado. Já o modelo físico, apresentado na Figura 3, é o modelo no qual se encontram as definições detalhadas da estrutura física do banco de dados, como tipo de dados, tamanho dos campos e índices.

Figura 3 - Modelo físico do banco de dados



Fonte: Do autor.

3.5 DESCRIÇÃO DAS TABELAS DO BANCO

A seguir é descrito nas tabelas criadas o dicionário de dados das entidades do modelo físico, detalhando suas colunas, as quais posteriormente serão criadas no processo de criação do banco de dados.

Tabela 1 - Tabela aluno

Coluna	Tipo de dados	Descrição
idAluno	Int	Chave primária da tabela
idEndereco	Int	Chave estrangeira da tabela endereço
nome	Varchar(255)	Descreve o nome de um aluno
dataNasc	Date	Descreve a data de nascimento de um aluno
turno	Varchar(255)	Descreve o turno que o aluno tem disponibilidade para estudo na escola
telefone	Varchar(255)	Descreve o telefone de contato do aluno
email	Varchar(255)	Descreve o e-*mail* de contato do aluno
cpf	Varchar(255)	Descreve o cpf do aluno
idEscola	Int	Chave estrangeira para a tabela escola que será preenchida com o código da escola, caso o aluno seja estudante de alguma escola da cidade
idResponsavel	Int	Chave estrangeira para a tabela responsável que conterá o código do responsável pelo aluno, caso este seja menor de idade

Fonte: Do autor.

Tabela 2 - Tabela aula

Coluna	Tipo de dados	Descrição
idAula	Int	Chave primária da tabela
idTurma	Int	Chave estrangeira da tabela turma
dataAula	Date	Descreve a data da aula
tipoAula	Int	Descreve os horários de aulas da turma
descricaoAula	Varchar(100)	Descreve a descrição das aulas

linkDrive	Varchar(255)	Armazena o link do diretório do repositório em nuvem (drive)
idProfessor	Int	Chave estrangeira da tabela funcionário que referencia o professor que ministrará a aula.

Fonte: Do autor.

Tabela 3 - Tabela curso

Coluna	Tipo de dados	Descrição
idCurso	Int	Chave primária da tabela
cargaHoraria	Int	Descreve a carga horária do curso
descricao	Varchar(45)	Descreve a descrição do curso

Fonte: Do autor.

Tabela 4 - Tabela emprestimo

Coluna	Tipo de dados	Descrição
id	Int	Chave primaria da tabela
idAluno	Int	Chave estrangeira da tabela aluno
idInstrumento	Int	Chave estrangeira da tabela instrumento
dataEmprestimo	Date	Descreve a data que foi realizado o empréstimo de um instrumento
dataDevolucao	Date	Descreve a data de devolução do instrumento.
observacao	Varchar(100)	Descreve observações que podem ser incluídas em um empréstimo.

Fonte: Do autor.

Tabela 5 - Tabela endereco

Coluna	Tipo de dados	Descrição
idEndereco	Int	Chave primária da tabela
cidade	Varchar(255)	Descreve a cidade.
logradouro	Varchar(255)	Descreve o logradouro do endereço
numero	Varchar(255)	Descreve o número do endereço

bairro	Varchar(255)	Descreve o nome do bairro do endereço
cep	Varchar(10)	Descreve o CEP do endereço
complemento	Varchar(255)	Descreve o complemento do endereço
idEstado	Int	Chave estrangeira da tabela de estado

Fonte: Do autor.

Tabela 6 - Tabela escola

Coluna	Tipo de dados	Descrição
idEscola	Int	Chave primária da tabela
nome	Varchar(255)	Descreve o nome da escola
telefone	Varchar(255)	Descreve o telefone da escola

Fonte: Do autor.

Tabela 7 - Tabela estado

Coluna	Tipo de dados	Descrição
idEstado	Int	Chave primária da tabela
nome	Varchar(75)	Descreve o nome do estado brasileiro
uf	Varchar(5)	Descreve a sigla de unidade federativa do estado

Fonte: Do autor.

Tabela 8 - Tabela funcionário

Coluna	Tipo de dados	Descrição
idFuncionario	Int	Chave primária da tabela
Nome	Varchar(60)	Descreve o nome de um funcionário
Cargo	Varchar(45)	Descreve o cargo do funcionário
telefone	Varchar(15)	Descreve o telefone do funcionário
Email	Varchar(45)	Descreve o e-*mail* do funcionário
idEndereco	Int	Chave estrangeira da tabela endereço
cpf	Varchar(20)	Descreve o cpf do funcionário

Fonte: Do autor.

Tabela 9 - Tabela instrumento

Coluna	Tipo de dados	Descrição
idInstrumento	Int	Chave primaria da tabela
descricao	Varchar(255)	Descreve o nome do instrumento musical
modelo	Varchar(255)	Descreve o modelo do instrumento musical
marca	Varchar(255)	Descreve a marca do instrumento musical
numeroSerie	Varchar(255)	Descreve o número de série do instrumento
observacoes	Varchar(255)	Descreve as observações sobre o instrumento.
emprestado	Int	Flag para identificar se o instrumento está emprestado ou não.

Fonte: Do autor.

Tabela 10 - Tabela matrícula

Coluna	Tipo de dados	Descrição
idAluno	Int	Chave primária e estrangeira da tabela aluno
idTurma	Int	Chave primária e estrangeira da tabela turma
data	Date	Descreve a data em que foi efetuada a matrícula

Fonte: Do autor.

Tabela 11 - Tabela responsavel

Coluna	Tipo de dados	Descrição
idResponsavel	Int	Chave primária da tabela
nome	Varchar(60)	Descreve o nome do responsável
telefone	Varchar(15)	Descreve a data em que foi efetuada a matrícula
email	Varchar(60)	Descreve o *e-mail* de contato do responsável

Fonte: Do autor.

Tabela 12 - Tabela turma

Coluna	Tipo de dados	Descrição
idTurma	Int	Chave primária da tabela

idCurso	Int	Chave estrangeira da tabela curso
horarioAulas	Int	Descreve os horários de aulas da turma
diasAulas	Varchar(60)	Descreve os dias das aulas da turma
dataInicio	Date	Descreve a data de início da turma
dataTermino	Date	Descreve a data de término da turma

Fonte: Do autor.

Tabela 13 - Tabela usuário

Coluna	Tipo de dados	Descrição
idUsuario	Int	Chave primária da tabela
#idFuncionario	Int	Chave estrangeira para a tabela funcionário
login	Varchar(45)	Descreve o nome de *login* do usuário
senha	Varchar(20)	Descreve a senha de *login* do usuário

Fonte: Do autor.

3.6 A APLICAÇÃO

A aplicação SysLyra é uma aplicação web que poderá ser executada nos principais navegadores de Internet como o *Google Chrome, Mozilla Firefox* e *Microsoft Edge*. Para dar início à etapa de desenvolvimento da aplicação, primeiro foi desenvolvido o *back-end* utilizando a tecnologia Node.js na versão 14.17.1, que é o servidor da aplicação e posteriormente passou-se para o desenvolvimento *front-end,* que foi construído utilizando o *framework* Vue.js.

A arquitetura do sistema é baseada no modelo de arquitetura *REST API*, que é definida como uma interface de programação de aplicações, que está em conformidade com as restrições do estilo de arquitetura *REST* criado pelo cientista da computação Roy Fielding. Os critérios para que uma *API* seja considerada do tipo *REST* é ter uma arquitetura cliente/servidor com solicitações gerenciadas por HTTP; estabelecer um comunicação *stateless* de modo que para cada solicitação do cliente ao servidor, nenhuma informação do cliente seja armazenada entre solicitações *GET* e todas solicitações sejam separadas e desconectadas; armazenar dados em cache para otimizar a eficiência da rede; ter uma interface uniforme para que as informações sejam transferidas de forma padronizada; ter um sistema em camadas hierárquicas que restringe o comportamento dos componentes; e uma restrição opcional que é

possibilitar a extensão de funcionalidade em forma de código, que pode ser baixado e executado no cliente (RED HAT, 2020).

Para auxiliar na construção da *API* foi utilizado o *framework Express* que é de código aberto, sobre a licença do *MIT* e que foi desenvolvido para otimizar a construção de aplicações web e de *APIs*, tornando-se um *framework* bem popular e que utiliza o Node.js para a execução do JavaScript como linguagem *back-end* (ANDRADE, 2021).

Para armazenamento dos dados da aplicação foi utilizado o sistema de gerenciamento de banco de dados *MySQL* na versão 8.0.27 que se comunica com a aplicação através do Sequelize que é um ORM (*Object Relational Mapper*) para Node.js e que tem suporte aos bancos de dados PostgreSQL, MariaDB, MySQL, SQLite, que como ORM faz o mapeamento de dados relacionais para objetos JavaScript permitindo criação, busca, alteração e remoção de dados dos bancos de dados através de métodos JavaScript (ORLANDI, 2018).

A ferramenta utilizada para desenvolvimento foi o editor de códigos *Visual Studio Code* e o ambiente de desenvolvimento utilizado foi um *notebook* com as seguintes características:

Marca: Acer

Modelo: Aspire E5-571

Memória RAM: 8GB RAM

Armazenamento: SSD 480GB

Sistema operacional: Windows 10 Home 64 bits

Processador: Intel(R) Core(TM) i5-4210U

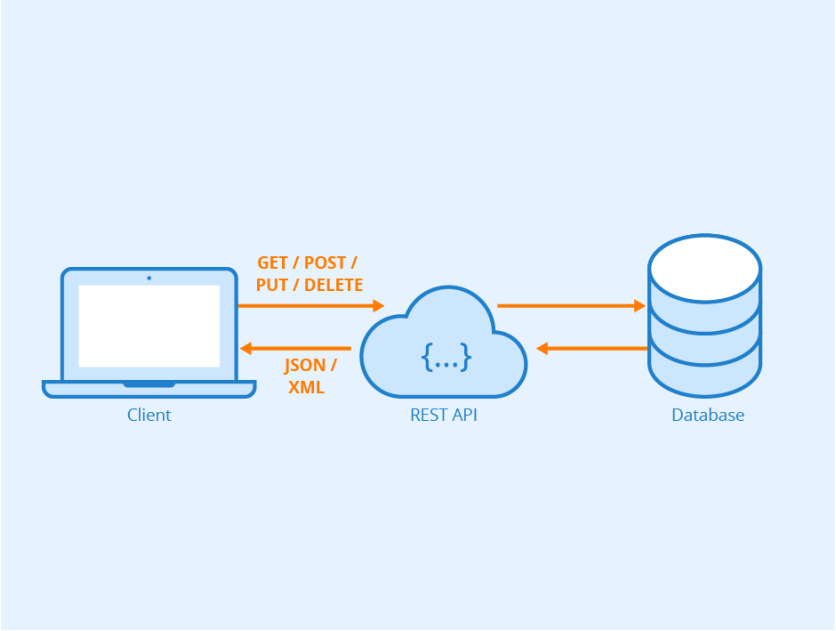
4 DESENVOLVIMENTO

Após realizada a etapa de levantamento e análise de requisitos e os estudos sobre as tecnologias escolhidas para a implementação do projeto, sendo as principais tecnologias o Node.js com o *framework* Express, Vue.js e o ORM Sequelize para manipulação do banco de dados MySQL, foi então realizado a etapa de desenvolvimento do sistema.

4.1 ARQUITETURA

A arquitetura escolhida para desenvolvimento do sistema foi o modelo cliente-servidor onde o *back-end* e *front-end* da aplicação são construídos em bases de código distintas, de forma com que se interajam por meio de uma interface de aplicação utilizando o padrão de comunicação REST. A Figura 4 ilustra o desenho da arquitetura cliente-servidor onde tem-se um dispositivo que ao enviar uma requisição, esta será processada pela API que então se comunicará com o banco de dados e após obter uma resposta, a API se responsabilizará em encaminhar a resposta para o cliente solicitante.

Figura 4 - Desenho da arquitetura



Fonte: Seobility (2022)

4.2 BACK-END

O *back-end* é onde os dados são tratados, e onde são feitas as requisições para o banco de dados para inserir, ler, arquivar esses dados, sendo ele a camada de trás da aplicação (ANTUNES, 2019). O *back-end* da aplicação SysLyra foi desenvolvido em Node.js, utilizando o *framework* Express.

A primeira tarefa feita foi a criação do servidor de desenvolvimento e em seguida passou-se para a tarefa de criação do banco de dados e as respectivas tabelas que constam no modelo físico exibido na Figura 3. A criação das tabelas no banco de dados se deu por meio de *migrations* (migrações) do Sequelize. As *migrations* permitem a criação e manipulação de banco de dados, tendo como objetivo fornecer recursos que permitam melhorar o gerenciamento das alterações realizadas no banco de dados, como por exemplo manter um histórico de alterações à medida que a base vai sendo modificada e reverter qualquer alteração feita (BORSATO, 2017). A Figura 5 mostra como foi criada a *migration* de criação da tabela de cursos.

Figura 5 - *Migration* de criação da tabela cursos

```
1  'use strict';
2  module.exports = {
3    async up(queryInterface, Sequelize) {
4      await queryInterface.createTable('Cursos', {
5        id: {
6          allowNull: false,
7          autoIncrement: true,
8          primaryKey: true,
9          type: Sequelize.INTEGER
10       },
11       descricao: {
12         type: Sequelize.STRING
13       },
14       cargaHoraria: {
15         type: Sequelize.STRING
16       },
17       nivel: {
18         type: Sequelize.INTEGER
19       },
20       createdAt: {
21         allowNull: false,
22         type: Sequelize.DATE
23       },
24       updatedAt: {
25         allowNull: false,
26         type: Sequelize.DATE
27       }
28     });
29   },
30   async down(queryInterface, Sequelize) {
31     await queryInterface.dropTable('Cursos');
32   }
33 };
```

Fonte: Do autor

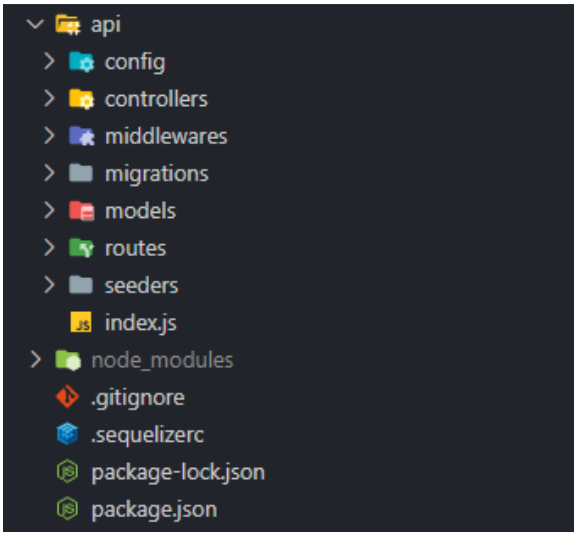
O arquivo de migração como mostrado na Figura 5, consiste na exportação de duas funções assíncronas, *up* e *down*, que são funções referentes ao *script* da migration e o *rollback* da mesma. A função *up* do exemplo da Figura 5 é a que se responsabiliza por criar a tabela cursos e seus respectivos atributos e a função *down*, a responsável por reverter a operação que neste caso, excluiria a tabela.

O projeto foi desenvolvido dividido em diretórios que seguem a seguinte estrutura:

- api/config – contém arquivos de configuração do projeto.
- api/controllers – contém os arquivos responsáveis em implementar os métodos tratam as requisições e respostas.
- api/middlewares – contém o *middleware* de autenticação.
- api/migrations – contém as migrations criadas para manipulação no banco de dados (criação, alteração e remoção).
- api/models – contém os arquivos de modelo que representam uma tabela no banco de dados, onde são criados os modelos, implementado as validações, regras de negócios e associações.
- api/routes – contém os arquivos de rotas da API.
- api/seeders – contém arquivos que foram utilizados para popular o banco com alguns dados.

A estrutura do projeto feita após a criação das tabelas no banco de dados é mostrada na Figura 6.

Figura 6 - Estrutura de pastas



Fonte: Do autor

4.2.1 Conexão com o banco de dados

Toda a parte de conexão e comunicação com o banco de dados ficou sob responsabilidade do ORM Sequelize cuja configuração de conexão foi estabelecida no arquivo config.json dentro da pasta /config. Após isso, foi criado o arquivo /.sequelizerc na raiz da pasta API que é o arquivo que se responsabiliza pelo mapeamento de cada pasta que o Sequelize usa de acordo com a estrutura de pastas definida no projeto. A Figura 7 mostra a configuração de conexão feita no arquivo config.json.

Figura 7 - Configuração de conexão com o banco de dados

```
{
  "development": {
    "username": "root",
    "password": "123456",
    "database": "syslyra",
    "host": "127.0.0.1",
    "port": "3307",
    "dialect": "mysql"
  }
}
```

Fonte: Do autor

A Figura 8 mostra como ficou configurado o arquivo .sequelizerc

Figura 8 - Configuração do .sequelizerc

```
const path = require('path');

module.exports = {
  'config': path.resolve('./api/config', 'config.json'),
  'models-path': path.resolve('api', 'models'),
  'seeders-path': path.resolve('api', 'seeders'),
  'migrations-path': path.resolve('api', 'migrations')
};
```

Fonte: Do autor

O Sequelize provê vários métodos que auxiliam na consulta e nos *scripts* de CRUD (*Create, Read, Update, Delete*) padrão, sem a necessidade do desenvolvedor escrever *scripts* em SQL, mas no servidor do banco de dados são os *scritps* em SQL que são executados. É através dos métodos da instância de um modelo do Sequelize

que as *queries* de CRUD são executadas, onde por exemplo se tem através do método *Model.create()* um atalho para construir uma instância não salva com o *Model.build()* e salvar a instância com *instance.save().* Um outro método que pode ser citado como exemplo para ler a tabela inteira do banco de dados é o *findAll()*¸ sendo possível a aplicação de cláusulas como a cláusula *where* para o filtro da consulta (SEQUELIZE, 2022). A Figura 9 ilustra a aplicação do método *findAll().*

Figura 9 - Método *findAll*()

```
Model.findAll({
  attributes: [
    'foo',
    [sequelize.fn('COUNT', sequelize.col('hats')), 'n_hats'],
    'bar'
  ]
});
```

Fonte: SEQUELIZE (2022)

4.2.2 Endpoints

O e*ndpoint* de uma API é o local em que as solicitações, também conhecidas como as chamadas de API, são atendidas. Os localizadores uniformes de recursos (URLs) que são usados na web para várias finalidades como para a localização de páginas web, são também utilizados para indicar os *endpoints* de uma API (CLOUDFLARE, 2022).

As chamadas de API que ocorrem em APIs do tipo REST são feitas através de métodos do protocolo HTTP, sendo os principais utilizados, os métodos GET, POST, PATCH, PUT e DELETE. No projeto SysLyra foi criado dentro da pasta *routes,* os arquivos de definição dos *endpoint*s da API, que apontam para o respectivo método do *controller* que irá tratar as requisições e as respostas de cada solicitação.

A Figura 10 mostra o arquivo de rotas de cursos e a Figura 11 mostra o método no arquivo CursoController.js que faz o tratamento da requisição do *endpoint* de atualização de um curso, no qual passa-se como parâmetro da requisição o id do curso, e obtém-se do corpo da requisição as informações que serão atualizadas.

Figura 10 - Exemplo de arquivo de rotas

```
const { Router } = require('express');
const CursoController = require('../controllers/CursoController');

const router = Router()
router
  .get('/cursos', CursoController.listarCursos)
  .get('/cursos/:id', CursoController.listarCursoPorId)
  .post('/cursos', CursoController.criarCurso)
  .put('/cursos/:id', CursoController.atualizarCurso)
  .delete('/cursos/:id', CursoController.excluirCurso)
module.exports = router
```

Fonte: Do autor.

Figura 11 - Método de atualização de curso no *Controller*

```
static async atualizarCurso(req, res){
  const id = req.params.id;
  const infos = req.body;

  try{
    await database.Cursos.update(infos,
      { where: {id: Number(id)}}
    );
    const cursoAtualizado = await database.Cursos.findOne({ where: {id: Number(id)}})
    return res.status(200).json(cursoAtualizado);
  }catch(err){
    return res.status(500).json(err.message);
  }
}
```

Fonte: Do autor.

4.2.3 Models

A camada de *models* é a camada responsável por gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócio estabelecidas (ZUCHER, 2020). Nos arquivos de modelo do projeto que é implementado a definição das entidades com seus respectivos atributos e as possíveis validações de *back-end*, além das definições de associação entre modelos.

A Figura 12 mostra o arquivo de modelo da entidade cursos que detalha a criação do modelo de cursos.

Figura 12 - Arquivo de modelo de cursos

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Aulas extends Model {

    static associate(models) {
      Aulas.hasMany(models.Aulas_matriculas, {
        foreignKey: 'idAula'
      })
      Aulas.belongsTo(models.Turmas, {
        foreignKey: 'idTurma'
      })
      Aulas.belongsTo(models.Funcionarios, {
        foreignKey: 'idFuncionario'
      })
    }
  }
  Aulas.init({
    descricao: DataTypes.STRING,
    tipo: DataTypes.STRING,
    data: DataTypes.DATEONLY,
    planoAula: DataTypes.STRING,
    concluida: DataTypes.BOOLEAN
  }, {
    sequelize,
    modelName: 'Aulas',
    paranoid: true,

  });
  return Aulas;
};
```

Fonte: Do autor.

4.2.4 Controllers

A camada de *controllers* (controladores) é a camada que no projeto tem a responsabilidade de intermediar as requisições enviadas pela camada de view com as respostas fornecidas pelo Model, processando os dados que o usuário informou e repassando para outras camadas (ZUCHER, 2020).

Nos arquivos controladores a classe de cada entidade possui métodos que recebem como parâmetros a requisição e resposta e retornam uma *Promise,* que é descrita na documentação do MDN como uma representação de um proxy para um valor que não é necessariamente conhecido quando a promessa é criada. Uma *Promise* pode estar em um estado pendente (*pending*), que é o estado inicial, realizada (*fulfilled*) ou rejeitada (*rejected*), sendo que quando ela passa do estado

pendente para algum dos outros dois, o método *then* é chamado e ele chama o método de tratamento referente ao estado da *promise* (MDN, 2022).

Os arquivos de controladores do projeto possuem em sua estrutura a criação de uma classe controladora que possui métodos assíncronos e estáticos que são executados pela classe quando chamados pelo *endpoint* respectivo configurado no arquivo de rota. A Figura 13 mostra como exemplo, os métodos listar cursos, criar curso e excluir curso criados no arquivo controlador da classe CursoController.

Figura 13 - *Controller* de cursos

```
class CursoController {

  static async listarCursos(req, res){
    try{
      const cursos = await database.Cursos.findAll();
      return res.status(200).json(cursos);
    }catch(err){
      return res.status(500).json(err.message);
    }
  }

  static async excluirCurso(req, res){
    let id = req.params.id;
    try{
      await database.Cursos.destroy({
        where: {
          id: Number(id)
        }
      })
      return res.status(200).send({
        message: `Curso de Id ${id} deletada`
      });
    }catch(err){
      return res.status(500).json(err.message);
    }
  }

  static async criarCurso(req, res){
    const novoCurso = req.body;
    try{
      const curso = await database.Cursos.create(novoCurso);
      return res.status(201).json(curso);
    }catch (err){
      return res.status(500).json(err.message);
    }
  }
```

Fonte: Do autor.

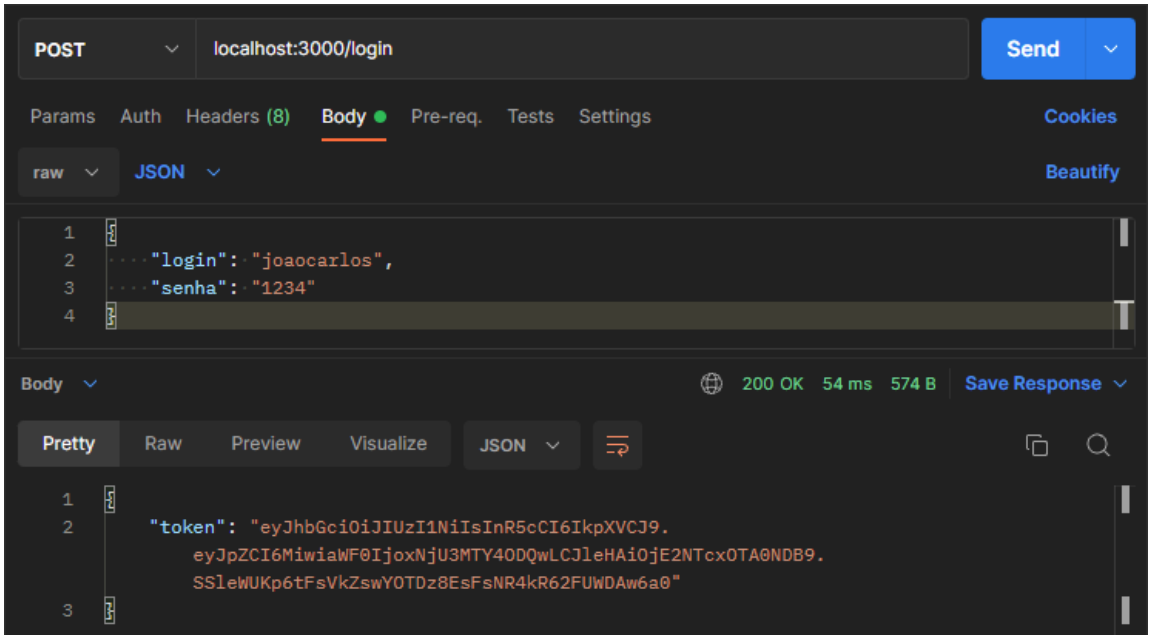
4.2.5 Autenticação

Como forma de promover segurança impedindo o acesso às rotas da API no servidor para usuários não autorizados foi implementado um mecanismo de autenticação para que apenas usuários logados no sistema, tenham acesso às respectivas funcionalidades em detrimento do seu nível de acesso.

Para implementar a autenticação no sistema foi utilizado o JSON Web Token, conhecido pela sigla JWT que é um padrão aberto (RFC 7519) que define uma forma compacta e autocontida para a transmissão segura de informações entre as partes como um objeto JSON, que se tornam seguras e confiáveis por serem assinadas digitalmente através de uma chave secreta com o algoritmo HMAC ou um par de chaves pública/privadas usando RSA ou ECDSA (JWT, 2022).

A autenticação acontece por meio do processo de *login* onde o usuário ao informar seu *username* e senha, e sendo estes validados, na resposta da requisição de *login* é retornado um *token* de acesso que dará acesso aos usuários para realizar as demais requisições do sistema, ao passar este *token* no cabeçalho (*header*) de autorização (*authorization)* como um *Bearer token*. Caso este *token* não seja informado, a resposta para as rotas protegidas é de *token* inexistente ou *token* inválido caso seja informado um *token* que não é validado. Na Figura 14 é mostrado um exemplo de uma requisição *login* de um usuário válido cadastrado no banco de dados.

Figura 14 - Requisição de *login* de usuário válido



Fonte: Do autor.

A implementação para validação do *token* foi feita através da construção de um *middleware* de autenticação que é passado no arquivo de configuração das rotas protegidas, de modo que a requisição toda vez que for chamada irá passar pelo *middleware*, e consequentemente validar se o *token* é válido ou inválido. Caso o *token* seja válido a requisição continua em seu fluxo, mas caso ocorra o contrário e o *token* seja invalidado, uma resposta já é retornada informando a mensagem de erro interrompendo o fluxo da requisição. Na Figura 15 é mostrado como foi implementado o *middleware* de autenticação.

Figura 15 - *Middleware* de autenticação

```
1   const jwt = require('jsonwebtoken');
2   const config = require('../config/jwt-config')
3   const { promisify } = require('util');
4
5   module.exports = async(req, res, next) => {
6
7     const auth = req.headers.authorization;
8     if(!auth){
9       return res.status(401).json({
10        error: true,
11        code: 130,
12        message: "Token inexistente"
13      })
14    }
15
16    const [bearer, token] = auth.split(' ');
17
18    try{
19      const decoded = await promisify(jwt.verify)(token, config.secret);
20      if(!decoded){
21        return res.status(401).json({
22          error: true,
23          code: 130,
24          message: "Token está expirado"
25        })
26      }else{
27        req.id = decoded.id;
28        next()
29      }
30    }catch(err){
31      return res.status(401).json({
32        error: true,
33        code: 130,
34        message: "Token está inválido"
35      })
36    }
37
38  }
```

Fonte: Do autor.

4.3 FRONT-END

O *front-end* de uma aplicação web é a camada do projeto de *software* que está relacionado com a parte visual, por meio do qual os usuários conseguirão interagir, normalmente utilizando as tecnologias base da *web* que são HTML, CSS e JavaScript (SOUTO, 2019). Para a construção do *front-end* da aplicação foi utilizado o *framework* Vue.js, que assim como outras bibliotecas e *frameworks* JavaScript para desenvolvimento *front-end,* como o React e o Angular, é muito utilizado para desenvolver aplicações do tipo *Single Page Applications* (SPA) e para desenvolver vários tipos de interfaces.

O desenvolvimento utilizando o Vue para criação das interfaces se dá por meio da criação de componentes de arquivo único (*Single File Component – SFC*) que possui a extensão *.vue*. A estrutura de um arquivo *.vue* é baseado em três blocos sendo o primeiro o bloco *template*, onde a estrutura da página é definida em código HTML junto com as diretivas do Vue.js que fazem ligação com o JavaScript. O segundo bloco é o *script,* onde é definido o código JavaScript do componente que manipula e interage de modo reativo com os elementos visuais do bloco *template* e o terceiro bloco, *style,* onde é definido a estilização do componente em CSS (GREGÓRIO, 2021).

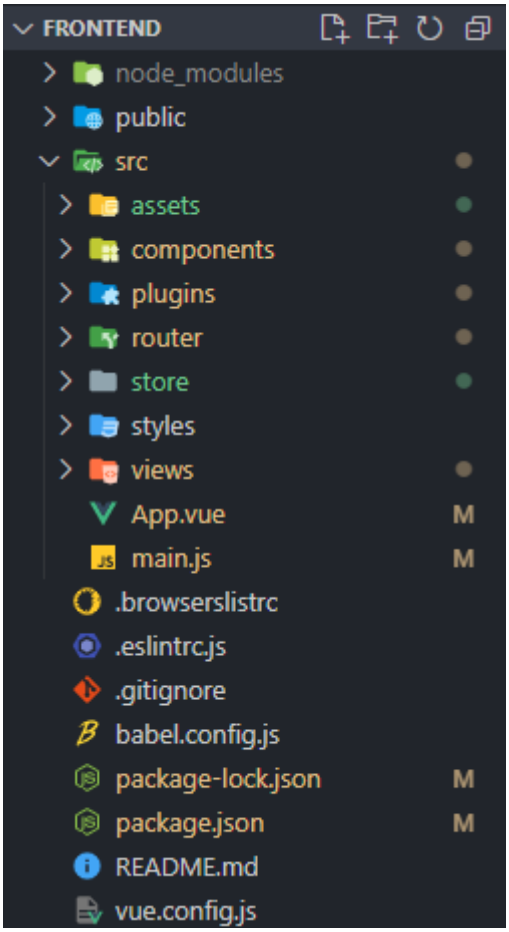
Para auxiliar na estilização do projeto e na reutilização de componentes padronizados foi utilizado o Vuetify que é uma biblioteca de UI baseado no Material Design que possui uma boa gama de componentes trabalhados para ser modular, responsivo e performático (VUETIFY, 2022).

Como encontra-se ilustrado na Figura 16, o projeto *front-end* foi construído com a seguinte estrutura:

- public – contém arquivos públicos que serão publicados na aplicação.
- src/assets – diretório para conter imagens, ícones, sons que serão utilizados no projeto.
- src/components – diretório que contém componentes customizados.
- src/plugins – contém os arquivos de configuração dos plugins utilizados no projeto.
- src/router – contém o arquivo de configuração das rotas da aplicação.

- src/store – contém o arquivo de criação do estado da aplicação e das mutações que é a única maneira de alterar o estado.
- src/styles – diretório que contém arquivo css de estilização global.
- src/views – contém os componentes das páginas da aplicação.
- App.vue – componente pai onde a estrutura da aplicação é renderizada.
- main.js – ponto de entrada onde a instância Vue é criada.

Figura 16 - Estrutura de pastas do projeto *front-end*



Fonte: Do autor.

4.3.1 Comunicação com o servidor

A aplicação *front-end* se comunica com o servidor por meio de chamadas HTTP e para esta finalidade foi utilizado a biblioteca Axios que é um cliente HTTP baseado em *Promises*. A configuração do axios foi feita no arquivo axios.js dentro do diretório de plugins, como mostrado na Figura 17.

Figura 17 - Arquivo de configuração do axios

```
1  import Vue from "vue";
2  import axios from "axios";
3  import provedor from "../store/provedor"
4
5  axios.defaults.baseURL = 'http://127.0.0.1:3000/'
6
7  axios.interceptors.request.use(function(config) {
8    const token = provedor.state.token
9    if(token){
10     config.headers.Authorization = `Bearer ${token}`
11   }
12   return config;
13 }, function (erro){
14   return Promise.reject(erro)
15 })
16
17 Vue.use({
18   install(Vue){
19     Vue.prototype.$http = axios
20   }
21 })
```

Fonte: Do autor.

No arquivo axios.js é informado como uma configuração padrão em axios.defaults.baseURL, o endereço onde as requisições serão realizadas. Também é feito neste arquivo a interceptação das requisições para verificar se o usuário pode ter acesso ao *endpoint* solicitado, sendo esta validação feita caso o usuário ao efetuar *login* tiver um *token* JWT devolvido como resposta. Dessa forma, caso exista um *token* válido nesta interceptação é adicionado à requisição um cabeçalho de autorização com o *token* de autenticação, para que o usuário então consiga ter acesso às rotas protegidas.

A Figura 18 mostra um exemplo de requisição GET que ao obter uma resposta do servidor, em caso de sucesso, atribuirá às determinadas propriedades do componente, os respectivos dados obtidos da chamada.

Figura 18 - Exemplo de requisição GET

```
this.$http.get(`cursos/${this.id}`).then((response) => {
  console.log(response.data);
  this.nome = response.data.nome;
  this.descricao = response.data.descricao;
  this.cargaHoraria = response.data.cargaHoraria;
  this.nivel = response.data.nivel;
});
```

Fonte: Do autor.

Na Figura 19 é mostrado um outro exemplo, porém de uma requisição POST onde é passado o *endpoint* de criação de turma e o objeto data que contém as informações necessárias para ser realizado o cadastro.

Figura 19 - Exemplo de requisição POST

```
let data = {
  descricao: this.descricao,
  idCurso: this.curso,
  horario: this.horario,
  dias: this.selected.toString(),
};

this.$http.post("turmas", data).then(
  (response) => {
    console.log(response);
    alert("Curso cadastrado com sucesso");
    this.$router.push({ name: "Turma" });
  },
  (err) => {
    alert("Não foi possível efetuar o cadastro. Erro: " + err);
  }
);
```

Fonte: Do autor.

4.3.2 Router

Para implementar a navegação entre as páginas da aplicação foi utilizada a biblioteca oficial de roteamento do Vue.js, vue-router. O vue-router provê um componente personalizado para criar *links* que é o *router-link* onde passa-se para o atributo *to*, o nome da rota caso seja nomeada ou o caminho dela. Um outro componente funcional que o vue-router provê, é o *router*-view que é local onde o componente relacionado à rota escolhida vai ser renderizado (VUE.JS, 2022). O *router-view* está localizado no projeto dentro do componente principal App.vue, como ilustra a Figura 20.

Dentro do diretório *router* foi criado o arquivo index.js, onde foi feito a configuração e atribuição de todas as rotas da aplicação que direcionam para uma *view* respectiva atribuída a um componente nomeado. A Figura 21 ilustra como exemplo a configuração das rotas que direcionam para o componente de listagem dos alunos, *ListAlunos*, cujo caminho (*path*) é “/alunos”, para o componente de cadastro

de aluno, *CadastroAlunos*, na opção de inclusão de um novo aluno cujo caminho é “/CadAlunos” e para o mesmo componente *CadastroAlunos*, mas na opção de alteração, diferenciada pelo id que pode ser passado na url, ficando o caminho para a rota nomeada *AlteraAluno*, o *path* “/cadAlunos/:id”.

Figura 20 - Bloco *template* do componente App.vue

```
<template>
  <div @isLogged="logado = $event">
    <div v-if="usuarioEstaLogado">
      <v-app id="inspire" class="overflow-hidden">
        <v-navigation-drawer v-model="drawer" app>
          <MenuLat></MenuLat>
        </v-navigation-drawer>

        <v-app-bar…
        </v-app-bar>

        <v-main>
          <v-slide-x-transition mode="out-in">
            <router-view />
          </v-slide-x-transition>
        </v-main>

        <Rodape />
      </v-app>
    </div>
    <Login v-else />
  </div>
</template>
```

Fonte: Do autor.

Figura 21 - Exemplo de rotas nomeadas do projeto *front-end*

```
  {
    path: "/alunos",
    name: "Aluno",
    component: ListAlunos,
    menu: true
  },
  {
    path: "/CadAlunos",
    name: "CadAluno",
    component: CadastroAlunos,
    menu: false
  },
  {
    path: "/cadAlunos/:id",
    name: "AlteraAluno",
    component: CadastroAlunos,
    menu: false
  },
```

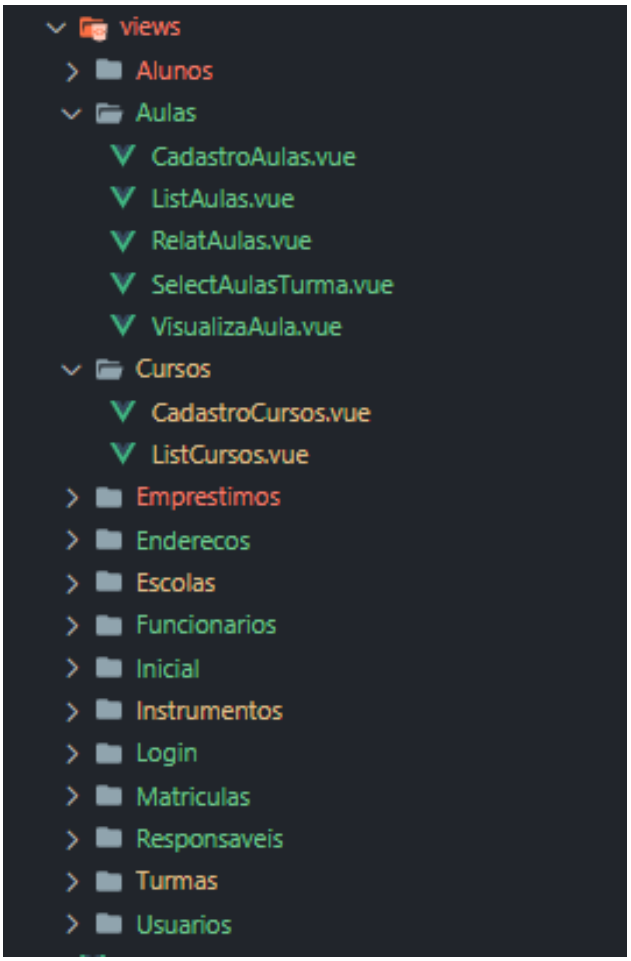
Fonte: Do autor.

4.3.3 Views

As telas do sistema foram implementadas como componentes de arquivos únicos do Vue, de tal modo que para cada componente tornou-se possível obter realce de sintaxe, bloco para execução de *scripts* para manipulação do componente utilizando JavaScript e bloco para definir CSS por componente.

Para cada grupo de cadastro foi criado uma pasta dentro do diretório *views*, onde nessas estão contidos os componentes relacionados ao cadastro, como os componentes de listagem do objeto relacionado, cadastro, componentes de relatório e de visualização como mostrado na Figura 22, onde cada arquivo de *view* criado representa uma interface no sistema.

Figura 22 - Pastas de *views*



Fonte: Do autor.

4.4 INTERFACES

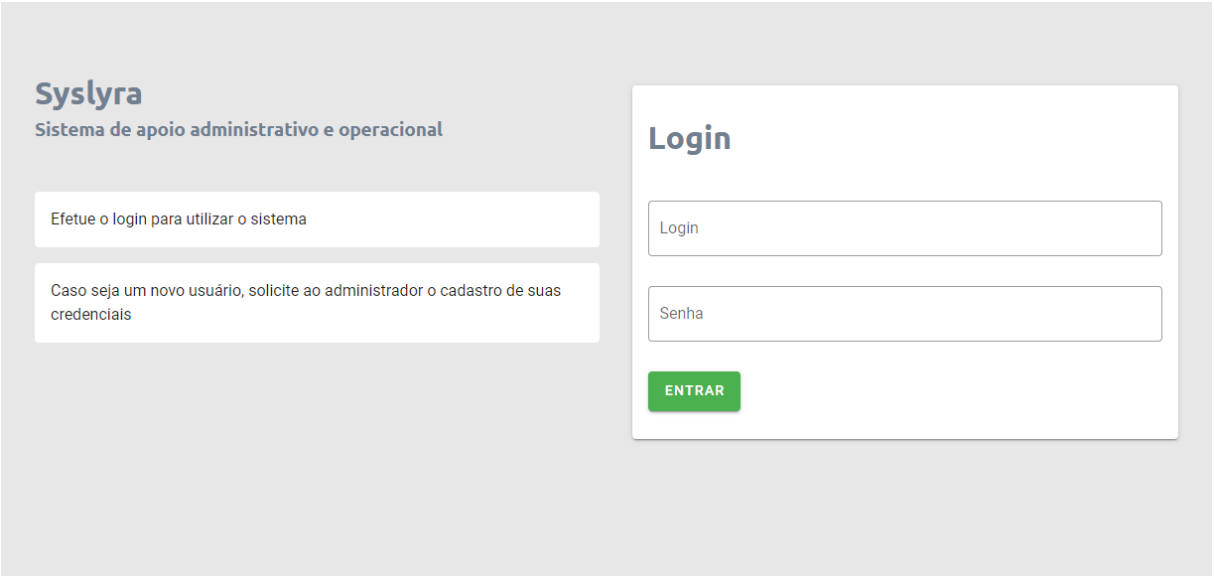
Nesta seção serão apresentadas as principais telas construídas para a aplicação.

4.4.1 Tela de Login

A tela de *login* é a primeira tela apresentada ao usuário ao tentar acessar a aplicação. Nesta tela o usuário deve entrar com o *login* e senha previamente cadastrado no cadastro de usuários para ter acesso às funcionalidades de gestão do sistema.

Quando o usuário insere os seus dados de *login* e senha no formulário de *login*, é feita uma requisição POST enviando esses dados para o servidor que analisará se os dados são válidos ou não. O servidor ao receber esses dados faz uma primeira validação se existe um nome de *login* igual ao recebido na tabela de usuários, sendo que caso não seja encontrado, uma resposta é devolvida, mas caso seja encontrado, uma segunda validação é feita, comparando se a senha recebida e a senha criptografada armazenada na tabela de usuários são iguais. Quando as validações passam, um *token* de autenticação é devolvido para o *client,* permitindo ao usuário o acesso ao sistema. A Figura 23 ilustra a tela de *login* do sistema.

Figura 23 - Tela de *login*

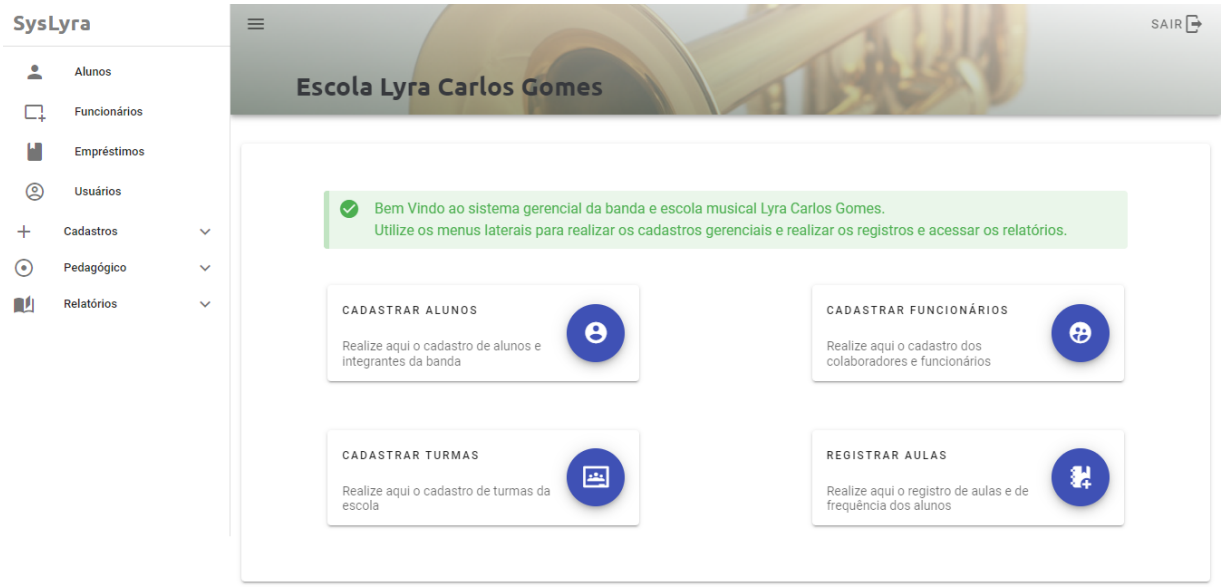


Fonte: Do autor

4.4.2 Tela Inicial

A tela inicial é a tela do sistema que é renderizada logo após o usuário logar no sistema. Nesta tela é disposto na lateral esquerda os menus do sistema. No centro da tela possui um alerta de boas-vindas e quatro *cards* com botões de atalho para quatro tipos de cadastros do sistema, que são os cadastros de alunos, funcionários, turmas e aulas. Na barra de navegação superior, existe um ícone de menu para expandir e esconder o menu lateral e um botão para fazer logout do sistema. A tela inicial é mostrada na Figura 24.

Figura 24 - Tela inicial



Fonte: Do autor.

4.4.3 Tela de Cadastro de Alunos

Na tela de cadastrados de alunos, a primeira visualização obtida pelo usuário é a listagem de alunos previamente cadastrados, exibindo destes alunos as informações de id, nome, telefone e turno e sendo possível realizar as ações de editar, excluir e realizar uma matrícula de um aluno em uma turma. Além disso, na parte superior à direita, há um botão “NOVO” que redireciona a página para a tela de cadastro de um novo aluno.

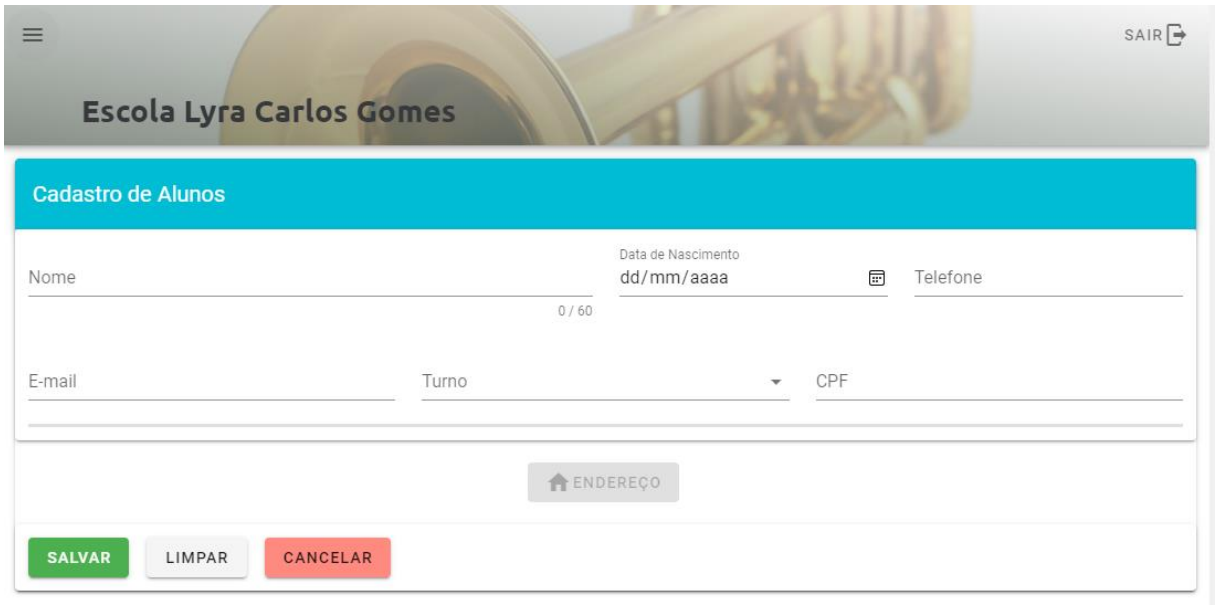
A Figura 25 ilustra a tela de listagem de alunos e a Figura 26 mostra a tela de cadastro de um novo aluno, onde é possível inserir os dados gerais de um aluno e após salvá-lo, ter acesso aos botões que redirecionam para o cadastro de endereço

de um aluno e para o cadastro de responsável e escolar, caso o aluno seja menor de idade ou pertencente à alguma escola.

Figura 25 - Exemplo de tela de listagem de alunos

Fonte: Do autor.

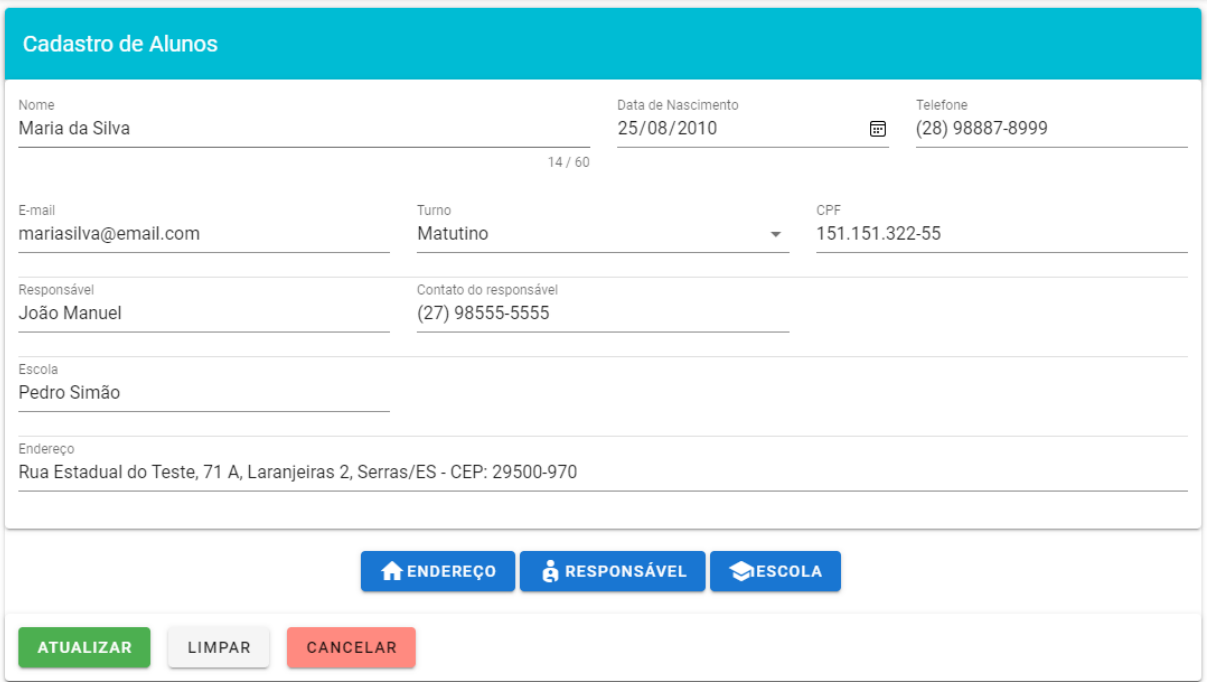
Figura 26 - Tela de Cadastro de um novo aluno



Fonte: Do autor.

A Figura 27 ilustra a tela de cadastro de um aluno na opção de edição quando selecionado o ícone de lápis na tela de listagem. Nesta tela as informações do aluno selecionado são carregadas automaticamente nos campos e exibidas para o usuário.

Figura 27 - Tela de edição de um aluno



Fonte: Do autor.

4.4.4 Tela de Cadastro de Turmas

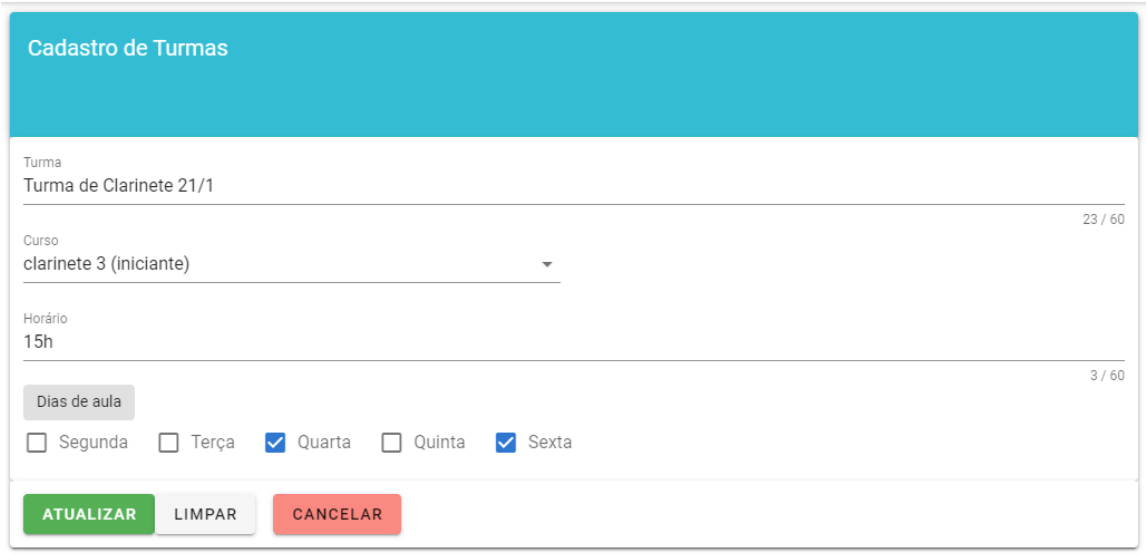
A maior parte das telas de cadastro criadas segue o mesmo padrão de construção, tendo um componente que primeiro renderiza a listagem de todos os registros, permitindo a opção de edição, exclusão e criação de um novo registro. A tela de listagem de turmas exibida na Figura 28 demostra o padrão que foi utilizado no projeto quando é acessado o menu de cadastro de turmas.

Figura 28 - Tela de listagem de turmas

Fonte: Do autor.

A Figura 29 mostra a tela de cadastro de turmas, porém no modo de edição, onde os dados do registro selecionado são exibidos automaticamente nos campos. Para isso, ao invés de ser criado um componente para edição de um registro, foi reaproveitado o componente de cadastro, onde a instância do componente ao ser criada, verifica se foi passado um id por parâmetro quando a rota foi chamada. Desse modo, caso seja identificado no gatilho *created* do Vue, que há um id passado por parâmetro, é feito uma requisição GET por id, que no caso do exemplo supracitado, é uma requisição GET no *endpoint* de turmas por id, assim, obtendo-se os dados da resposta da requisição e atribuindo-os aos campos do componente, como mostra a Figura 30.

Figura 29 - Tela de cadastro de turmas



Fonte: Do autor

Figura 30 - Gatilho *created* do componente de cadastro de turmas

```
created() {
  if (this.id) {
    this.$http.get(`turmas/${this.id}`).then((response) => {
      console.log(response.data);
      this.descricao = response.data.descricao;
      this.horario = response.data.horario;
      this.curso = response.data.idCurso;
      this.selected = response.data.dias.split(",");
    });
  }
  this.$http.get("cursos/").then((response) => {
    this.cursos = response.data;
  });
},
```

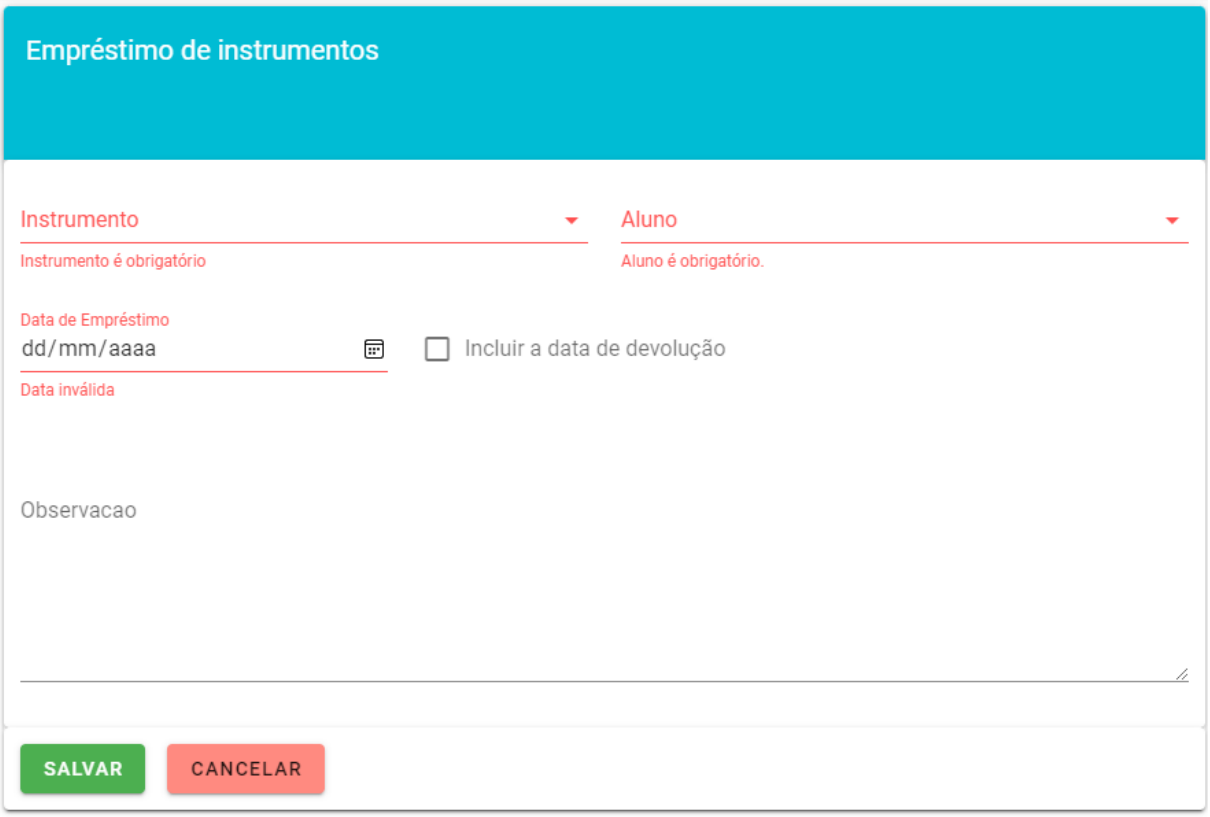
Fonte: Do autor.

Uma observação a ser feita no cadastro de turmas é que o horário das aulas de uma turma é fixo, segundo verificação feita diretamente com um dos diretores da Lyra. Por isso foi implementado apenas um campo fixo para cadastro de horário de aulas da turma.

4.4.5 Tela de Registro de empréstimos de instrumentos

A Figura 31 mostra a tela de cadastro de um novo empréstimo onde o usuário precisa selecionar o instrumento que está sendo emprestado, em seguida o aluno para o qual o empréstimo está sendo feito, e inserir a data que o empréstimo foi feito. Os campos são obrigatórios e validados ao tentar salvar o registro. Caso algum dos campos não sejam informados, é exibido uma mensagem de erro na parte inferior do campo que falhou na validação como mostra também a Figura 31.

Figura 31 - Tela de registro de empréstimo de instrumento



Fonte: Do autor

Foi verificado com um dos diretores musicais, que a data de devolução não é um requisito obrigatório, tendo em vista que o empréstimo de instrumento é feito na

maioria das vezes enquanto o aluno participa da banda ou do projeto de educação musical da Lyra Carlos Gomes. Com isso, foi implementado um *checkbox* para o usuário selecionar caso ele queira registrar uma data de devolução, mas por padrão o *checkbox* vem desmarcado.

Na tela de listagem de empréstimos quando o usuário faz opção por excluir o empréstimo é apresentado um *popup* para confirmação. Uma vez confirmada a devolução do instrumento, o mesmo torna-se disponível para ser emprestado novamente para um outro aluno. A Figura 32 mostra o *popup* exibido ao clicar na opção de excluir um empréstimo.

Figura 32 - Popup de exclusão de empréstimo de instrumento

Fonte: Do autor.

4.4.6 Tela de Cadastro de Matrículas

No sistema há apenas duas interfaces de cadastros que possuem uma interação um pouco diferente da que foi mostrada na subseção 5.4.4. Essas telas são as telas de cadastro de matrículas e cadastro de aulas. Na Figura 33 é mostrado como exemplo a interface exibida quando o usuário seleciona o menu de cadastro de matrículas, onde ele precisa selecionar o aluno para então efetuar ou gerenciar matrículas do aluno selecionado. Por isso, antes de realizar uma nova matrícula é necessário que aluno esteja previamente cadastrado no sistema.

Figura 33 - Tela de Cadastro de Matrículas

Fonte: Do autor.

O mesmo comportamento do cadastro de matrículas acontece na tela de cadastro de aulas, porém ao invés do usuário ter que selecionar um aluno, ele deverá selecionar uma turma. Assim, uma vez selecionada a turma, é possível criar uma aula nova ou visualizar e editar as aulas já existentes para a turma.

A Figura 34 mostra a renderização da tabela que é mostrada quando o usuário seleciona um aluno para realizar uma nova matrícula, ou para visualizar, editar e excluir alguma já existente.

Figura 34 - Tela de Cadastro de Matrículas quando selecionado um aluno

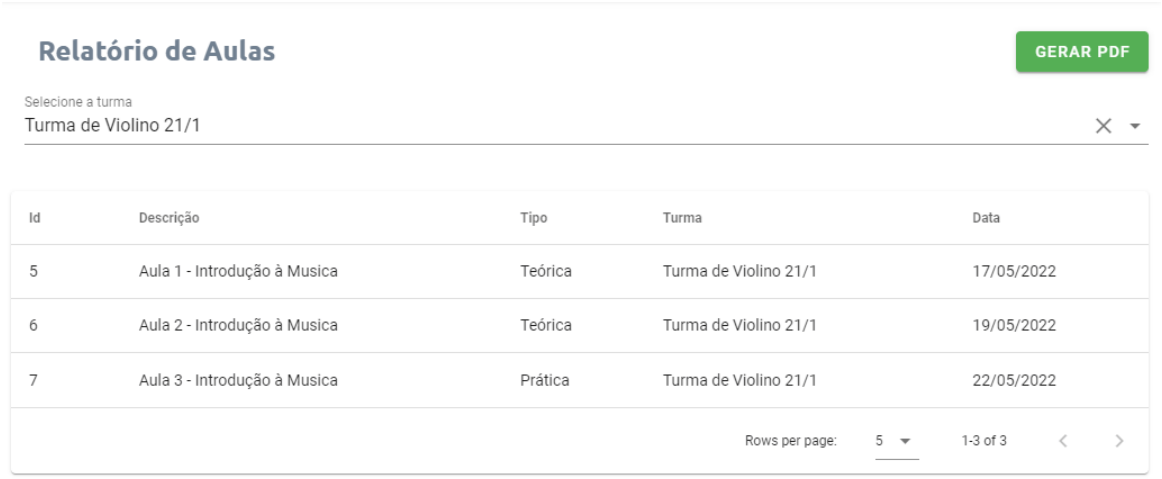
Fonte: Do autor.

4.4.7 Tela de Relatórios

As telas dos relatórios também foram criadas em componentes de arquivo único e possuem funcionamento semelhante às telas de listagem de registro, porém

a diferença é que nas telas de relatório não é possível realizar nenhum tipo de ação nos registros, apenas visualizar e gerar arquivo .pdf ao selecionar o botão “GERAR PDF”. Com exceção no relatório de empréstimo é possível também realizar a ação de emitir um relatório por registro. A Figura 35 mostra a tela de relatório de aulas.

Figura 35 - Tela de relatório de aulas



Fonte: Do autor.

Na Figura 36 é mostrado o relatório gerado em PDF, quando o usuário seleciona o botão de gerar PDF. O mesmo modelo de relatório mostrado na Figura 36 é utilizado também nos relatórios de alunos, empréstimos e instrumentos.

Figura 36 - Relatório de aulas em pdf

Relatório de Aulas 13/07/2022 11:14:57

Turma: Turma de Violino 21/1

Id	Descrição	Tipo	Data	Observações
5	Aula 1 - Introdução à Musica	Teórica	18/05/2022	
6	Aula 2 - Introdução à Musica	Teórica	19/05/2022	Continuado matéria de introdução à musica.
7	Aula 3 - Introdução à Musica	Prática	23/05/2022	Pratica de Introdução realizada com os alunos

Fonte: Do autor.

No relatório de empréstimos há a opção de filtrar os empréstimos por data como é apresentado na Figura 37. No exemplo ilustrado foi feita uma busca dos empréstimos realizados entre os dias 17/07/2022 e 18/07/2022.

Figura 37 - Relatório de empréstimo

Fonte: Do autor.

5 TESTES E VALIDAÇÕES

Os testes de *software* são destinados a mostrar que um programa faz o que é proposto a fazer e para descobrir os defeitos do sistema antes do seu uso, sendo parte de um processo de validação e verificação do sistema, que possuem como objetivo final estabelecer a confiança de que está pronto para seu propósito atendendo aos requisitos especificados e à expectativa do cliente (SOMMERVILLE, 2007).

5.1 INSPEÇÕES

Após finalizada a etapa de desenvolvimento, foi executado a atividade de inspeção como uma das atividades do processo de validação e verificação. Segundo Sommerville (2007), as inspeções servem para analisar e verificar os requisitos do sistema, modelos de projeto, o código fonte e até mesmo os testes de sistema propostos, tendo como objetivo descobrir erros e falhas no sistema. A Tabela 14 descreve as principais inspeções que foram realizadas no sistema.

Tabela 14 - Tabela de inspeções.

Defeito	**Solução**
-Ao tentar realizar *login* os campos digitados caso estivessem inválidos não eram limpos, atrapalhando a usabilidade do sistema.	-Foi adicionado o método de limpar campos caso a resposta da requisição de *login* fosse inválida.
-Ao cadastrar ou atualizar um aluno mesmo quando o campo nome estava preenchido, era mostrada uma mensagem de erro como campo requerido.	-Foi corrigido a validação que estava sendo aplicada sobre o campo nome. Para isso, alterou-se na validação de *name* para nome.
-O campo de telefone no cadastro de alunos permitia a inserção de letras.	-Feita a correção aplicando uma máscara de telefone sobre o campo de telefone no cadastro de alunos e nos demais que também possuem o campo de telefone.

-O cadastro de alunos apresentava inconsistência na hora de cadastrar um aluno.	-Revisado e alterado a forma que o sistema estava se comportando no cadastro de alunos.
-Algumas rotas estavam desprotegidas e acessíveis a qualquer usuário	-Feito a correção atribuindo o *middleware* de autenticação às rotas par aprotegê-las.

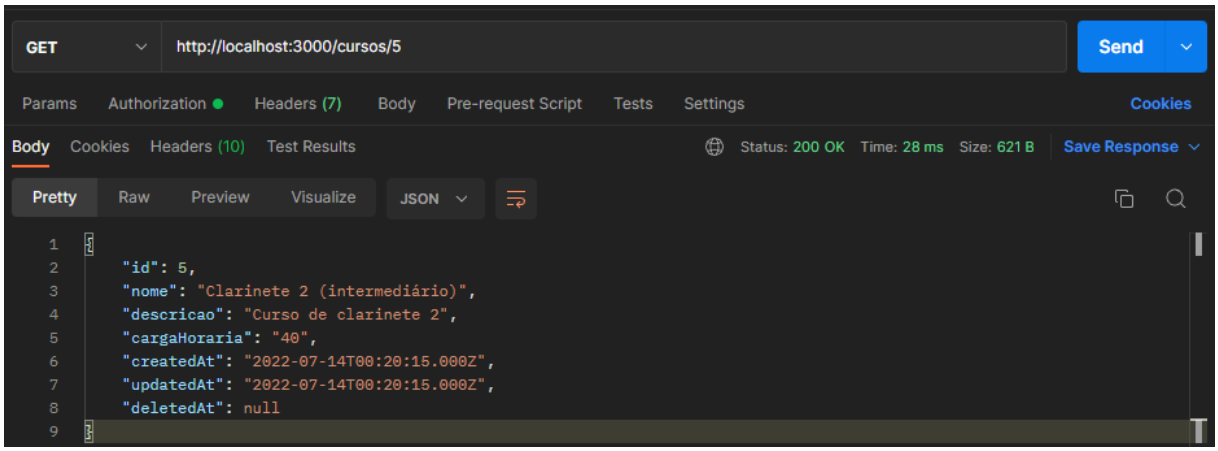
Fonte: Do autor.

5.2 TESTE DE DESENVOLVIMENTO

Durante a etapa de desenvolvimento, foi executado como um tipo de teste de desenvolvimento, os testes de API utilizando a ferramenta Postman para tal finalidade, que é uma plataforma para a construção e uso de APIs, que simplifica cada etapa do ciclo de vida da API e agiliza a colaboração para a criação de APIs com mais agilidade (POSTMAN, 2022).

Os testes de desenvolvimento são testes feitos durante a etapa de desenvolvimento para descobrir *bugs* e defeitos. Durante a construção da API, a medida que foram sendo criadas as rotas dos *endpoints*, utilizou-se do Postman para testar o envio e a resposta das requisições, não deixando esse processo de validação para apenas depois que todas as rotas estivessem criadas. Na Figura 38 é mostrado um teste feito para verificar a resposta de uma requisição GET no *endpoint* de cursos passando o id igual à 5 para testar o método de listar curso por id.

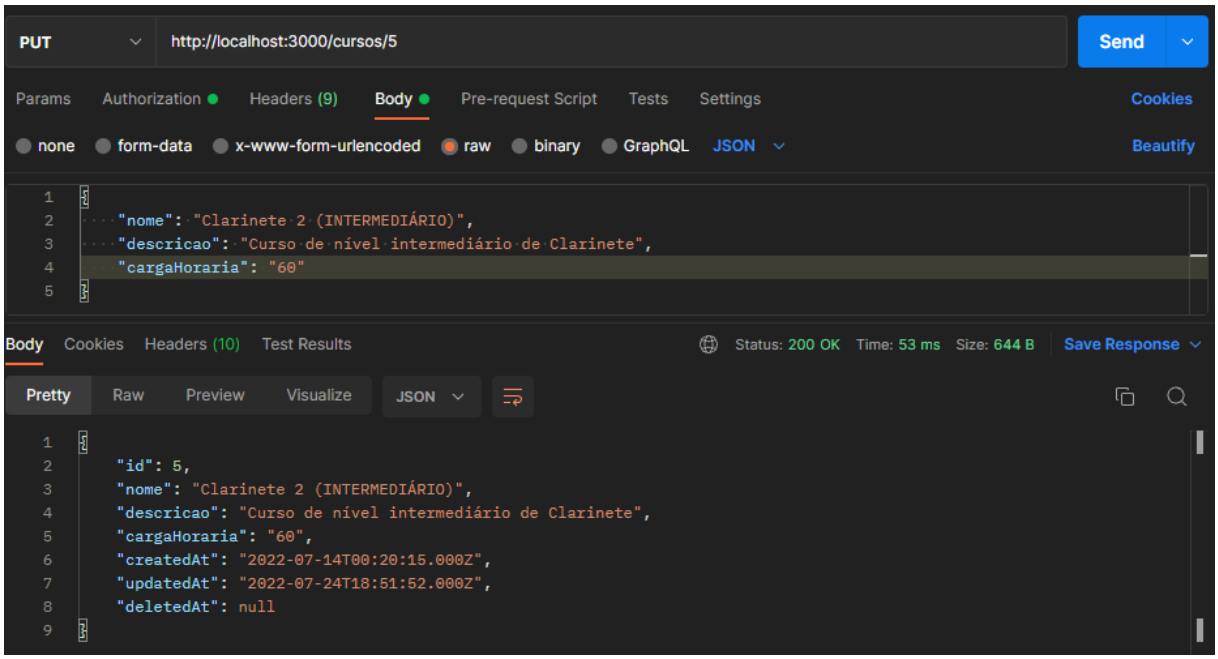
Figura 38 - Teste de requisição GET no *endpoint* de cursos por id



Fonte: Do autor.

A Figura 39 exemplifica o teste de uma requisição PUT no *endpoint* de atualização de um curso. Para esse teste, foi passado no *body* da requisição um objeto JSON com as informações a serem atualizadas.

Figura 39 - Teste de requisição PUT no *endpoint* atualização de curso



Fonte: Do autor.

Com os testes utilizando o Postman foi possível visualizar de modo prático principalmente as respostas obtidas com o envio das requisições e o status que era retornado, auxiliando na validação do processo de implementação dos métodos que foram criados ao longo do desenvolvimento.

Um outro tipo de teste de desenvolvimento realizado foi o teste de sistema onde o principal objetivo era o de testar o sistema como um todo. Segundo Sommervile (2007) a execução dos testes de sistema prioriza-se em verificar se os componentes desenvolvidos são compatíveis, se interagem entre si corretamente e se transferem os dados certos no momento certo.

A escolha por esse tipo de teste, se deu pelo fato de a interface do sistema ser construída sob abordagem de componentes e também pois a arquitetura utilizada é a arquitetura cliente-servidor, onde dois projetos diferentes precisam ser integrados e funcionar corretamente para pleno funcionamento do sistema.

Para auxiliar na execução dos testes foi elaborado o seguinte plano de testes demonstrado na Tabela 15.

Tabela 15 - Plano de testes.

Funcionalidades	**Comportamento esperado**	**Verificações**	**Critérios de aceite**
Login	- Ao digitar o *username* e a senha corretamente o usuário irá logar no sistema podendo ter acesso às demais funcionalidades. - O usuário deverá ser redirecionado para tela inicial do sistema. -Após o *token* de autenticação ser expirado o sistema deverá ser redirecionado para a tela de *login*.	- *Login* no sistema com sucesso. - *Username* ou senha inválidos. - Usuário não preencher os campos obrigatórios. - Aguardar *token* expirar.	- Campos de *username* e senha são obrigatórios. - Em caso de sucesso no *login*, redirecionar para a página principal. - Após *token* expirar, o sistema deve retornar à tela de *login*.
Cadastros	- Deve registrar corretamente os dados que foram informados pelo usuário. - Deve informar ao usuário os campos obrigatórios a ser corrigidos pelo usuário. - Deve informar alertas de sucesso	- Exibe mensagem de confirmação em caso de sucesso no cadastro. - Exibe mensagem de falha. - Em caso de sucesso, redirecionar para a página de listagem referente ao cadastro.	- Exibe a mensagem “Cadastrado com sucesso” ao realizar um cadastro. - Ao realizar o cadastro é redirecionado para a tela de listagem referente ao cadastro feito.

	ou falha na operação. - Deve ser possível alterar e excluir um registro salvo.		
Matrícula	- Deve vincular um aluno a uma turma que o aluno ainda não esteja matriculado.	- Exibe mensagem de confirmação da matrícula. - Matricular um aluno em uma turma na qual já foi matriculado.	-Em caso de sucesso, mostra mensagem de confirmação ao realizar matrícula. -Não permite que um aluno seja cadastrado em uma turma que ele já esteja cadastrado.
Emissão de relatórios	- Será mostrado a visualização dos registros em forma de tabela para o usuário. - Deve permitir a exportação do relatório para um arquivo em pdf ao clicar no botão de “Gerar PDF”.	- Apresenta registros corretos em relação ao que está cadastrado. - Opção de gerar pdf.	- Carrega relatório. - Gera PDF.
Pesquisa	- Ao digitar um texto na barra de pesquisas nos componentes de listagem de registros deverá aparecer os	- Pesquisa retornou os resultados esperados. - Pesquisa não retornou os resultados.	- Mostra na tabela os registros filtrados quando utilizado o campo de pesquisa.

	registros correspondentes que foram filtrados. - Deve ser exibido os registros à medida que o usuário digita no campo de busca.		

Fonte: Do autor.

Com os testes feitos baseados no plano de testes descritos na Tabela 15 foram encontradas algumas falhas que então foram corrigidas como:

- No cadastro de aluno o sistema não se comportava corretamente na hora de salvar um aluno mostrando alertas inconsistentes. Feito a correção na tela e no modo como as chamadas de API estavam sendo feitas para salvar os alunos.
- Alguns campos obrigatórios não estavam sendo explícitos como obrigatórios quando tentava-se salvar um registro, como por exemplo os dados de nome, telefone, data de nascimento e endereço no cadastro de alunos. Foi feito a correção para informar os campos obrigatórios ao tentar atualizar ou criar um registro.
- O sistema não estava redirecionando para a tela de *login* quando o *token* estava expirado. Feito a tratativa no *front-end* para verificar expiração do *token* e redirecionar o usuário para a tela de *login*.

5.3 TESTE DE USUÁRIO

Um outro tipo de teste realizado foi o teste de usuário. Segundo Sommerville (2007), um teste de usuário é um estágio no processo de teste no qual os usuários fornecem entradas e conselhos sobre o sistema, podendo este estágio envolver um teste formal de um sistema que já foi aprovado por um fornecedor externo ou processo informal onde os usuários podem experimentar do produto de *software* novo para ver se gostam e se satisfaz o que precisam.

Ainda de acordo com o autor citado, existem três tipos de teste de usuário, o teste alfa, onde usuários e desenvolvedores trabalham juntos para testar o *software* no local do desenvolvedor; o teste beta, no qual um *release* do *software* é disponibilizado aos usuários para que possam experimentar e levantar os problemas que identificarem, e o teste de aceitação no qual os clientes testam para decidir se o sistema está pronto ou não para ser aceito e implantado no ambiente do cliente.

Para realização do teste foi feita uma reunião com um dos diretores que está à frente do projeto da Lyra Carlos Gomes onde nesta reunião foi realizado um teste alfa, no ambiente de desenvolvimento com o projeto rodando, onde o usuário pôde ter contato com o sistema, conhecendo sobre e tendo a oportunidade de testar suas funcionalidades.

Nesta reunião o cliente identificou alguns campos de determinados cadastros que poderiam ser adicionados, dentre os quais cita-se a opção de ter um campo que no cadastro de aula apontasse para uma página do drive para ter acesso à uma pasta que contém os arquivos que foram ministrados na aula. Além disso o cliente solicitou a possibilidade de no relatório de empréstimo, ter a opção de emitir o relatório por aluno.

Outra observação feita pelo usuário foi sobre o campo “nível” que estava presente no cadastro de curso e que segundo o usuário poderia ser removido, uma vez que o nível está relacionado à obra estudada (música) e não ao curso. Além deste campo, o usuário solicitou a remoção do cadastro de avaliações e de frequências de aulas, uma vez que por ora, a escola pretende dar ênfase à gestão administrativa, restringindo o acesso ao sistema apenas à diretores. Um outro ponto percebido durante o teste, foi que a exclusão de matrícula estava apresentando falha. Diante das observações feitas e das solicitações feitas pelo usuário, os pontos foram anotados e posteriormente, ajustados e implementados.

Durante o teste feito com o usuário, passou-se por momentos de explicação sobre o sistema, resposta às dúvidas do usuário, sugestões para o sistema e *feedback* que por parte do usuário foi positivo, ao comentar que o sistema entrega funcionalidades precisas e funcionais para o estado atual estado da escola.

5.4 TESTE DE USABILIDADE

O teste de usabilidade é um teste que tem como intuito provar a consistência e a solidez de uma interface ao colocá-la para uso em situações comuns, buscando entender como o sistema se comporta no dia a dia, na naturalidade da utilização e se atende aos requisitos estabelecidos. Para esse tipo de teste pode ser utilizado várias abordagens como os testes de benchmark, descoberta de problemas, teste competitivo e teste de aprendizado (WOEBCKEN, 2021).

Para realização deste teste foi feita uma reunião no dia dezenove de julho (19/07/2022) com o orientador Giuliano Prado para validação do sistema e foi utilizado o modelo de teste de descoberta de problemas, onde o principal objetivo do teste é identificar problemas, brechas ou falhas que quebram o produto ou que revelam inconsistência no funcionamento da aplicação. Durante a reunião, as funcionalidades implementadas no sistema foram testadas e as inconsistências e falhas identificadas foram anotadas para passar por correção. A Tabela 16 detalha os resultados do teste de usabilidade.

Tabela 16 - Teste de usabilidade.

Defeitos e observações	Solução	Data da solução
Cor de alerta de boas-vindas da página inicial muito chamativa no tom de vermelho, que geralmente remete a situações de erro.	Alterado a cor do alerta para um tom verde claro.	19/07/2022
Validação dos campos obrigatórios no cadastro de alunos e cadastro de instrumentos.	Foi verificado com o usuário que os campos obrigatórios no cadastro de alunos são nome, telefone, data de nascimento e endereço; e no cadastro de instrumentos, nome e endereço. Após isso, foi feito a implementação das validações.	20/07/2022

No cadastro de turma, foi verificado que não tinha o campo de data de início da turma e na opção de edição, para atualização da turma, não estavam sendo atualizadas todas as informações corretamente.	Adicionado o campo de data de início da turma e corrigido e feito validação dos campos na opção de atualização de turmas.	24/07/2022
Botão de efetuar matrícula presente na tabela da listagem de alunos sem clareza da funcionalidade.	Adicionado *tooltip* que mostra o nome “matrícula” ao passar o mouse sobre o botão.	19/07/2022
Verificado que ao cadastrar o aluno nenhum comportamento acontecia para orientar o cadastro de endereço do aluno.	Alterado o comportado para que ao cadastrar um aluno, automaticamente seja redirecionado para a tela de cadastro de endereço do aluno.	24/07/2022
Depois de cadastrar um novo aluno foi verificado que o sistema permitia o vínculo de um responsável e de uma escola, mesmo o aluno sendo maior de idade.	Foi implementado uma lógica que calcula a idade do aluno no momento do cadastro e apenas mostra os botões de escola e responsável, se o aluno for menor de idade.	21/07/2022
A data de devolução estava como um campo estático no cadastro de empréstimo de instrumento, sendo que segundo o usuário final a data de devolução não é obrigatória.	Foi criado um campo de *checkbox* para o usuário selecionar caso queira registrar uma data de devolução para o empréstimo, mas por *default* o *checkbox* vem desmarcado.	21/07/2022

Foi apresentado erro na hora de registrar uma matrícula.	Feito a correção da lógica de salvar e atualizar matrícula.	23/07/2022
O *login* de usuários é criado de forma estática no cadastro de usuários dentro do sistema. Necessário validar com o cliente este tipo de cadastro.	Verificado com o usuário final que a forma como é feita o cadastro de usuários é funcional tendo em vista que são poucos usuários que terão acesso ao sistema.	21/07/2022
Observado que no cadastro de aulas seria mais semântico se o nome do campo observações fosse plano de aulas.	Alterado o nome do campo de observações para plano de aula.	20/07/2022
No cadastro de matrículas, ao matricular um aluno em uma turma o usuário tem que digitar a data que a matrícula está sendo efetuada. Validado que seria mais usual se esta data fosse automaticamente preenchida com a data atual.	Implementado a lógica para que quando for efetuada uma nova matrícula, a data seja obtida automaticamente com a data do dia.	24/07/2022
As aulas estavam sem nenhuma identificação sobre o seu estado, se concluída ou em aberto.	Criado a coluna “concluida” na tabela de aulas no banco de dados e implementado funcionalidade que atribui a *flag* concluida como verdadeira quando a aula for marcada como finalizada, sendo este um comportamento que reage na listagem de aulas.	23/07/2022

No relatório de aulas, trocar o nome para “Aulas por Turma”.	Feito a alteração do nome do relatório de aulas para “Aulas por Turma”	20/07/2022
No relatório de empréstimos não havia possibilidade de filtrar os empréstimos por data.	Implementado os filtros de data e pesquisa no relatório de empréstimos e a funcionalidade de emitir PDF dos empréstimos filtrados.	23/07/2022

Fonte: Do autor

6 CONSIDERAÇÕES FINAIS E CONCLUSÕES

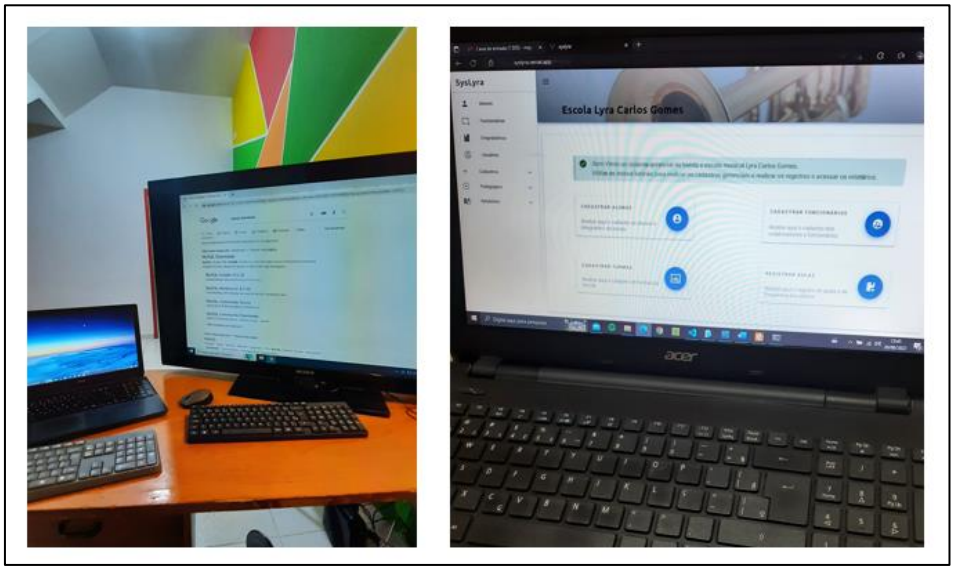
Os estudos sobre a informatização nas entidades que atuam no Terceiro Setor tornam-se bem relevantes, uma vez que revelam problemas existentes em grande parte dessas entidades, que enfrentam dificuldades para se manter de modo eficiente no que diz respeito principalmente à gestão, à comunicação e à organização. Conforme apresentado ao longo do trabalho, pode-se reafirmar a importância que existe no investimento de tecnologias, em meio às instituições filantrópicas, uma vez que a adoção de tecnologia afeta e impacta positivamente na forma como se lida com os dados e as informações na gestão dessas entidades, substituindo o burocrático e lento processo manual do tratamento de dados, por formas que auxiliem, facilite e automatize a maioria dos processos.

O presente trabalho mostrou-se como uma solução para uma dificuldade enfrentada por uma entidade filantrópica da cidade de Alegre, a banda e escola de música Lyra Carlos Gomes. Por meio de um estudo sobre as necessidades desta instituição e através de um sistemático trabalho que envolveu desde atividades de engenharia de *software*, engenharia de requisitos, programação e gestão de qualidade, tornou-se possível a elaboração de um sistema por meio deste projeto, que atende às necessidades gerenciais da escola de música, auxiliando no desenvolvimento desta.

O desenvolvimento deste trabalho foi realizado utilizando tecnologias que não são usualmente ensinadas nas disciplinas do curso, o que promoveu boa parte de tempo dedicado a estudo em cima das tecnologias que foram escolhidas para implementar o sistema, dentre as quais pode-se citar como exemplo, a tecnologia Node.js, o *framework* Vue.js, o consumo de APIs, sendo o acesso à fóruns especializados e à documentações, atividades que foram comuns e de suma importância para ajudar no desenvolvimento da aplicação.

O sistema desenvolvido foi implantado na escola no dia dezenove de agosto de dois mil e vinte e dois (19/08/2022), onde durante um período de duas horas foi feito, uma atividade de instalação e configuração do servidor de dados e da aplicação para tornar a aplicação disponível para a escola e um momento de breve explicação ao usuário sobre a funcionalidade e usabilidade do sistema. A Figura 40 mostra duas fotos do dia da implantação e de entrega do sistema à Lyra Carlos Gomes.

Figura 40 - Fotos da implantação



Fonte: Do autor.

Segundo João (2012) a tecnologia por si só não é capaz de promover às empresas ou organizações mais competitividade, eficiência e torná-las orientadas para a qualidade, sendo preciso, para aproveitar do poder da tecnologia da informação, a realização de mudanças, que podem ser pequenas ou grandes, sendo essas implementadas através de uma gestão de processos de negócios que busca a melhoria contínua desses processos.

Com isso, tendo em vista o projeto desenvolvido neste trabalho, à medida que a escola de música Lyra Carlos Gomes for se desenvolvendo ou à medida que surgirem novos requisitos funcionais, consequentemente novas funcionalidades podem ser adicionadas ao projeto, de modo que ele agregue ainda mais valor na utilização do sistema. Como exemplo de funcionalidades que podem ser incrementadas ao sistema cita-se a implementação de controle de acesso, que possa separar as funcionalidades visíveis por tipo de usuário caso novos tipos de usuários possam ter acesso ao sistema futuramente, além de uma opção de redefinição de senhas por *e-mail* para que usuários já cadastrados no sistema não tenham a necessidade de entrar em contato com um usuário administrador para solicitar alteração de senha. Além disso, o sistema faz um controle de instrumentos que são emprestados apenas atribuindo a *flag* emprestado na tabela de instrumentos igual a 1 se o instrumento estiver emprestado e depois igual a 0 quando é o empréstimo é

finalizado, mas não faz o registro em uma tabela de devoluções, sendo este um requisito que pode ser interessante implementar posteriormente.

O sistema SysLyra desenvolvido neste trabalho pretende atender a instituição no que diz respeito à gestão administrativa, auxiliando no controle, organização e gestão das informações de negócios da escola, porém pode não ser considerado como um sistema que auxilie tanto no âmbito operacional, uma vez que por decisão dos usuários finais, faz mais sentido para a atual situação da escola um sistema que foque mais na gestão do âmbito administrativo, reunindo esses dados em um só lugar. Entretanto, este sistema traz consigo uma nova oportunidade de ajudar uma instituição filantrópica a se organizar internamente auxiliando no enfrentamento das dificuldades gerenciais existentes.

Como parte do projeto de extensão UFES PROTI o desenvolvimento do sistema SysLyra mostrou-se uma importante ferramenta que poderá auxilir a escola Lyra Carlos Gomes em suas atribuições no nível de gestão escolar, pois até então, por utilizar técnicas e métodos manuais e desintegrados, a escola acaba perdendo em agilidade, centralização de informação e até mesmo em perda de informações importante para levantamento de um futuro histórico. A utilização desse sistema pretende-se resolver alguns dos problemas iniciais com a centralização de dados reunindo informações em um banco de dados, promovendo agilidade e precisão no tratamento dos dados. Outro fator que é importante citar é a possibilidade de fornecer à uma instituição filantrópica da cidade de Alegre, uma realidade de inclusão digital, através da utilização de um sistema de informação que foi resultado de desenvolvimento do projeto de extensão que objetiva que alunos do curso de Sistemas de Informação da UFES contribuam com o desenvolvimento regional por meio do projeto.

REFERÊNCIAS

ALVES, William P. **Banco de Dados.** São Paulo: Editora Saraiva, 2014. 9788536518961. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536518961/. Acesso em: 21 fev. 2022.

ANDRADE, Ana Paula de. **O que é o Express.js?** Treina Web, 2021. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-express-js. Acesso em: 26 fev. 2022.

ANTUNES, Ana. **FRONT-END E BACK-END: O QUE É E PRA QUE SERVE?** 2019. Disponível em: https://www.gobacklog.com/blog/front-end-e-back-end/. Acesso em: 29 jun. 2022.

BATISTA, Emerson de Oliveira. **Sistemas de Informação**: o uso consciente da tecnologia para o gerenciamento. 2. ed. São Paulo: Saraiva, 2012.

BORSATO, Eduardo Cesar. **Desenvolvendo migrations utilizando Laravel**. 2017. Disponível em: https://imasters.com.br/back-end/desenvolvendo-migrations-utilizando-laravel. Acesso em: 03 jul. 2022.

CARVALHO, Willian. Construindo um servidor web com Node.js. **DevMedia**, 2015. Disponível em: https://www.devmedia.com.br/construindo-um-servidor-web-com-node-js/32023. Acesso em: 19 fev. 2022.

CLOUDFLARE. **O que é um endpoint de API?** 2022. Disponível em: https://www.cloudflare.com/pt-br/learning/security/api/what-is-api-endpoint/. Acesso em: 04 jul. 2022.

DUFLOTH, Simone Cristina et al. Gestão da informação do ambiente externo em organizações do terceiro setor. 2005.

FRANÇA FILHO, Genauto Carvalho de. **Terceiro setor, economia social, economia solidária e economia popular**: traçando fronteiras conceituais. 2002.

GREGÓRIO, Jorge Luís. **Vue.js 3.x – 02 – Entendendo a estrutura do projeto**. 2021. Disponível em: https://www.jlgregorio.com.br/2021/04/23/vue-js-3-x-02-entendendo-a-estrutura-do-projeto-vue/. Acesso em: 06 jul. 2022.

GUEDES, Gilleanes T. A. **UML 2 : uma abordagem prática**. 3. ed. São Paulo: Novatec Editora, 2018.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. 5. ed. Porto Alegre: Sagra Luzzatto, 2004. xvii, 236 p. (Livros didáticos; Instituto de Informática da UFRGS; 4). ISBN 9788524105906 (broch.).

HIRAMA, Kechi. **Engenharia de software**: qualidade e produtividade com tecnologia. Rio de Janeiro: Elsevier, 2011

INCAU, C. **Vue.js – Construa aplicações incríveis**. Ed. 1. São Paulo: Casa do Código, 2017.

JOÃO, Belmiro N. (org.). **Sistemas de Informação**. São Paulo: Pearson Education do Brasil, 2012.

JÚNIOR, Leopoldo Costa. **Terceiro setor e economia social**. Fundação Getulio Vargas, EAESP - Escola de Administração de Empresas de São Paulo, Centro de Estudos do Terceiro Setor, 1998.

JWT. **Introduction to JSON Web Tokens**. 2022. Disponível em: https://jwt.io/introduction. Acesso em: 05 jul. 2022.

KURZ, Robert. **Os últimos combates**. Ed. 4. Petrópolis, RJ : Vozes, 1997.

LAUDON, Kenneth; LAUDON, Jane. **Sistemas de informação gerenciais**. 9. ed. São Paulo: Pearson Prentice Hall, 2010. Tradução Luciana do Amaral Teixeira; revisão técnica Belmiro Nascimento João.

LYRA CARLOS GOMES. HISTÓRIA. 2020. Disponível em: https://lyracarlosgomes.com/historia/. Acesso em: 30 nov. 2021.

MANZANO, José Augusto N. G.. **MySQL 5.5 interativo**: guia essencial de orientação e desenvolvimento. São Paulo : Érica, 2011.

MDN. **Promise**. 2022. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise. Acesso em: 05 jul. 2022.

ORLANDI, Claudio. **Configurando o ORM Sequelize no NodeJS com ExpressJS**. Blog da Rocketseat, 2018. Disponível em: https://blog.rocketseat.com.br/nodejs-express-sequelize/. Acesso em: 26 fev. 2022.

PEREIRA, C. R. **Node.js: Aplicações web real-time com Node.js**. 1. ed. Editora Casa do Código, 2014.

POSTMAN. **What is Postman?** 2022. Disponível em: https://www.postman.com/product/what-is-postman/. Acesso em: 23 jul. 2022.

RED HAT. **API REST**, 2020. Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em: 25 de fev. 2022.

SALES, José Avando. **A Importância Da Tecnologia No Terceiro Setor**. 2017. Disponível em: https://www.filantropia.ong/informacao/a-importancia-da-tecnologia-no-terceiro-setor. Acesso em: 16 fev. 2022.

SATO, Danilo Toshiaki. **Uso Eficaz de Métricas em Métodos Ágeis de Desenvolvimento de Software**. 2007. 136 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, 2007.

SEBRAE - Serviço Brasileiro de Apoio às Micro e Pequenas Empresas. **Entenda o que são OSCIPs e como elas funcionam**. 2019. Disponível em: https://www.sebrae.com.br/sites/PortalSebrae/bis/oscip-organizacao-da-sociedade-civil-de-interesse-publico,554a15bfd0b17410VgnVCM1000003b74010aRCRD#:~:text=Uma%20OSCIP%20%C3%A9%20uma%20qualifica%C3%A7%C3%A3o,entidades%20t%C3%ADpicas%20do%20terceiro%20setor. Acesso em: 01 fev. 2022.

SEOBILITY. **REST API**. 2022. Disponível em: https://www.seobility.net/en/wiki/REST_API. Acesso em: 04 jul. 2022.

SEQUELIZE. **Model querying - Basics**. 2022. Disponível em: https://sequelize.org/docs/v6/core-concepts/model-querying-basics/. Acesso em: 04 jul. 2022.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. **Sistema de banco de dados**. 7. ed. Rio de Janeiro: Ltc, 2020. Tradução Daniel Vieira ; revisão técnica Daniel Sadoc Menasché.

SOMERVILLE, Ian. **Engenharia de Software, 8ª edição.** 8. ed. São Paulo: Pearson Addison-Wesley, 2007

SOUTO, Mário. **O que é front-end e back-end?** 2019. Disponível em: https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end. Acesso em: 24 jul. 2022.

STAIR, Ralph M.; REYNOLDS, George W.. **Princípios de Sistemas de Informação**. 11. ed. São Paulo: Cengage Learning, 2015. Tradução Noveritis do Brasil; revisão técnica Tânia Fátima Calvi Tait.

TEÓFILO, Romero Batista; DE FREITAS, Lucia Santana. O uso de tecnologia da informação como ferramenta de gestão. **IV Simpósio de Excelência em Gestão e Tecnologia (SEGET).** Resende, Rio de Janeiro, p. 1-12, 2007

UFES PROTI. **Projeto de assessoria socialmente responsável em tecnologia da informação UFES Campus Alegre**. Disponível em: http://proti.alegre.ufes.br/. Acesso em: 30 nov. 2021.

VUE.JS. **Getting Started**. 2022. Disponível em: https://router.vuejs.org/guide/. Acesso em: 24 jul. 2022.

VUETIFY. **Introduction**. 2022. Disponível em: https://vuetifyjs.com/en/introduction/why-vuetify/. Acesso em: 06 jul. 2022.

WOEBCKEN, Cayo. **Entenda o que é teste de usabilidade, para que serve e como é feito**. 2021. Disponível em: https://rockcontent.com/br/blog/teste-de-usabilidade/. Acesso em: 23 jul. 2022.

XAVIER, Lays. **10 melhores sistemas para escolas**. 2022. Disponível em: https://www.galaxpay.com.br/post/10-melhores-sistemas-para-escolas/128. Acesso em: 18 jul. 2022.

ZUCHER, Vitor. **O que é padrão MVC? Entenda arquitetura de softwares!** 2020. Disponível em: https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc#:~:text=A%20camada%20de%20controle%20%C3%A9,e%20repassando%20para%20outras%20camadas.. Acesso em: 24 jul. 2022.