

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**

**JHONY SGANZERLA**

**SISTEMA WEB PARA GESTÃO DOS CURSOS DE EXTENSÃO**

**PATO BRANCO**

**2023**

**JHONY SGANZERLA**

**SISTEMA WEB PARA GESTÃO DOS CURSOS DE EXTENSÃO**

**WEB SYSTEM FOR MANAGEMENT EXTENSION COURSES**

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).  
Orientadora: Andreia Scariot Beulke

**PATO BRANCO**

**2023**



Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

**JHONY SGANZERLA**

**SISTEMA WEB PARA GESTÃO DOS CURSOS DE EXTENSÃO**

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação:19/06/2023

---

Andreia Scariot Beulke  
Mestrado  
Universidade Tecnológica Federal do Paraná

---

Vinicius Pegorini  
Mestrado  
Universidade Tecnológica Federal do Paraná

---

Mariza Miola Dosciatti  
Doutorado  
Universidade Tecnológica Federal do Paraná

**PATO BRANCO**

**2023**

## RESUMO

O Departamento Acadêmico de Informática (DAINF) da Universidade Tecnológica Federal do Paraná (UTFPR) Campus Pato Branco desempenha um papel importante na comunidade ao promover cursos de informática como atividades de extensão. Esses cursos têm como objetivo principal promover a inclusão digital, aumentar as oportunidades de emprego e incentivar os participantes a seguirem carreiras na área de informática ou áreas relacionadas. Considerando esse contexto, como resultado deste trabalho foi desenvolvido um sistema web para a gestão dos cursos de extensão oferecidos pelo DAINF do Campus Pato Branco. A gestão desses cursos envolve dados de usuários e dos cursos, tais como: instrutores (alunos voluntários dos cursos de graduação da UTFPR), professores envolvidos, turmas, data de início e término, cursos ofertados, entre outros. Os dados armazenados auxiliam na gestão das atividades e no controle de presença dos participantes dos cursos para a emissão de certificados, entre outros. Dentre as tecnologias utilizadas no desenvolvimento do sistema estão a linguagem de programação Java, o *framework* Spring para o desenvolvimento do *back-end*, Angular para o desenvolvimento do *front-end* e o Postgres para o banco de dados.

Palavras-chave: sistema; gestão; extensão.

## **ABSTRACT**

The Academic Department of Computer Science (DAINF) at the Federal Technological University of Paraná (UTFPR) Pato Branco Campus plays an important role in the community by promoting computer science courses as extension activities. These courses aim to promote digital inclusion, increase employment opportunities, and encourage participants to pursue careers in the field of computer science or related areas. Considering this context, as a result of this work, a web system was developed for the management of extension courses offered by DAINF at the Pato Branco Campus. The management of these courses involves user and course data, such as: instructors (volunteer students from UTFPR undergraduate courses), involved professors, classes, start and end dates, offered courses, among others. The stored data assists in the management of activities and attendance control of course participants for certificate issuance, among other purposes. Among the technologies used in the development of the system are the Java programming language, the Spring framework for back-end development, Angular for front-end development, and Postgres for the database.

Keywords: system; managment; extension.

## LISTA DE FIGURAS

Figura 1 – Fluxograma do Spring MVC.....	21
Figura 2 – Casos de uso.....	29
Figura 3 – Diagrama de mapeamento do banco de dados.....	33
Figura 4 – Tela de autenticação.....	34
Figura 5 – Cadastro de Usuário.....	35
Figura 6 – Tela de Dashboard.....	36
Figura 7 – Tela de listagem de usuário.....	36
Figura 8 – Cadastro de usuário com campos inválidos.....	37
Figura 9 – Tela de edição de usuário.....	38
Figura 10 – Tela de confirmação de exclusão do registro.....	38
Figura 11 – Tela de listagem de usuários com perfil de instrutor.....	39
Figura 12 – Tela de alteração de usuário com perfil de instrutor.....	39
Figura 13 - Modelo padrão para registros básicos.....	40
Figura 14 – Tela de cadastro de equipe.....	41
Figura 15 – Tela de cadastro de aluno.....	41
Figura 16 – Tela de listagem de turma para chamadas.....	42
Figura 17 – Tela de controle de chamada.....	43
Figura 18 – Tela de chamada sem alunos vinculados.....	59

## LISTA DE QUADROS

<b>Quadro 1 – Lista de ferramentas e tecnologias.....</b>	<b>20</b>
<b>Quadro 2 – Requisitos funcionais.....</b>	<b>26</b>
<b>Quadro 3 – Requisitos não funcionais.....</b>	<b>28</b>
<b>Quadro 4 – Operação para incluir registros no banco de dados.....</b>	<b>29</b>
<b>Quadro 5 – Operação para alterar registros no banco de dados.....</b>	<b>30</b>
<b>Quadro 6 – Operação para excluir registros no banco de dados.....</b>	<b>31</b>
<b>Quadro 7 – Operação de consultar registros no banco de dados.....</b>	<b>32</b>
<b>Quadro 8 – Vincular instrutores a curso.....</b>	<b>32</b>

## LISTA DE CÓDIGO-FONTES

<b>Código-fonte 1 - WebSecurity.....</b>	<b>44</b>
<b>Código-fonte 2 - HTML da tela de Login.....</b>	<b>46</b>
<b>Código-fonte 3 - Classe de Usuário.....</b>	<b>47</b>
<b>Código-fonte 4 - Propriedades da aplicação.....</b>	<b>49</b>
<b>Código-fonte 5 - HTML da lista de usuários.....</b>	<b>50</b>
<b>Código-fonte 6 - Typescript da lista de usuários.....</b>	<b>52</b>
<b>Código-fonte 7 - HTML do cadastro de Habilidades.....</b>	<b>54</b>
<b>Código-fonte 8 - Typescript do cadastro de habilidades.....</b>	<b>55</b>
<b>Código-fonte 9 - HTML da tela de chamada.....</b>	<b>56</b>
<b>Código-fonte 10 - Typescript do cadastro de chamada.....</b>	<b>60</b>



## LISTA DE ABREVIATURAS E SIGLAS

DAINF	Departamento Acadêmico de Informática
FORPROEX	Fórum de Pró-Reitores de Extensão das Universidades Públicas Brasileiras
HTML	<i>Hypertext Markup Language</i>
MVC	<i>Model, View e Controller</i>
SEMVER	<i>Semantic Versioning</i>
SI	Sistemas de Informação
SPA	<i>Single Page Application</i>
TI	Tecnologia da Informação
UTFPR	Universidade Tecnológica Federal do Paraná
URL	<i>Uniform Resource Locator</i>
PNE	Plano Nacional de Educação
RA	Registro Acadêmico

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>13</b>
<b>1.1</b>	<b>Considerações iniciais.....</b>	<b>13</b>
<b>1.2</b>	<b>Objetivos.....</b>	<b>14</b>
1.2.1	Objetivo Geral.....	15
1.2.2	Objetivos Específicos.....	15
<b>1.3</b>	<b>Justificativa.....</b>	<b>15</b>
<b>2</b>	<b>EXTENSÃO UNIVERSITÁRIA.....</b>	<b>17</b>
<b>3</b>	<b>MATERIAIS E MÉTODO.....</b>	<b>20</b>
<b>3.1</b>	<b>Materiais.....</b>	<b>20</b>
3.1.1	Spring Framework.....	21
3.1.2	Spring MVC.....	21
3.1.3	Spring Boot.....	22
3.1.4	Spring Security.....	22
3.1.5	Spring Data.....	22
3.1.6	Angular.....	23
<b>3.2</b>	<b>Método.....</b>	<b>24</b>
<b>4</b>	<b>RESULTADOS.....</b>	<b>25</b>
<b>4.1</b>	<b>ESCOPO DO SISTEMA.....</b>	<b>25</b>
<b>4.2</b>	<b>MODELAGEM DO SISTEMA.....</b>	<b>26</b>
<b>4.3</b>	<b>Apresentação do Sistema.....</b>	<b>34</b>
<b>4.4</b>	<b>Implementação do Sistema.....</b>	<b>43</b>
<b>5</b>	<b>CONCLUSÃO.....</b>	<b>64</b>
	<b>REFERÊNCIAS.....</b>	<b>66</b>

## 1 INTRODUÇÃO

Este capítulo tem como objetivo fornecer uma visão geral do contexto no qual se insere a proposta do sistema desenvolvido como resultado deste trabalho. Nesse contexto, é importante destacar o ambiente e as circunstâncias que motivaram o desenvolvimento do trabalho, a fim de compreender a relevâncias e a necessidade da solução proposta. Em seguida são apresentados os objetivos do trabalho que descrevem as metas a serem alcançadas. Ainda, é apresentada a justificativa de realização do trabalho abordando a importância e relevância da solução proposta. Por fim, são apresentados os capítulos subsequentes que detalham as diferentes seções e aspectos abordados ao longo do trabalho.

### 1.1 Considerações iniciais

A tecnologia *web* surgiu na década de 90 como uma forma de repositório do conhecimento humano (BERNERS-LEE, 1994). Com o passar dos anos ela foi evoluindo e se tornando uma das formas mais utilizadas para acesso aos sistemas de informação. Essa evolução permitiu a incorporação de novos recursos e funções, deixando de ser apenas um repositório de documentos eletrônicos ligados por hipertexto para se tornar uma fonte de geração de novas oportunidades de negócio (ZANETTI JUNIOR; VIDAL, 2004).

De acordo com CETIC.BR (ano), os sistemas *web* permitem acesso aos mais variados tipos de informações e serviços, sendo: sobre a empresa, catálogos de produtos, fornecimento de suporte pós-venda, personalização ou customização de produtos para clientes, lista de preços, sistemas de pedidos ou reservas, pagamento *on-line* (CGI.BR, 2014).Essas atividades envolvem a interação entre clientes e fornecedores nas suas mais variadas combinações (clientes e fornecedores, clientes e clientes, fornecedores e fornecedores, sejam eles pessoas físicas ou jurídicas), permitindo a implementação de processos de negócio (JACHYNTO, 2008).

Pesquisa realizada em 2017 pelo CETIC.BR, sobre a existência de departamento de Tecnologia da Informação (TI) nas empresas indica que 35% das empresas de pequeno porte possuem esse tipo de departamento, esse percentual sobe para 66% nas empresas de médio porte e nas empresas de grande porte são 89% delas que contam com esse tipo de departamento (CETIC.BR, 2017).

Assim como nas empresas, nas universidades são produzidos diversos tipos de informação, seja no meio acadêmico ou na gestão. Essas informações estão relacionadas ao ensino (atividades de aula), pesquisa (atividades científicas) e extensão (atividades que envolvem a comunidade). E para cada atividade são produzidas informações correspondentes. O controle e o gerenciamento dessas informações devem ser realizados por Sistemas de Informação (SI) para que seja prático e ágil armazenar, recuperar e utilizar dados. A extensão universitária permite compartilhar com o público externo o conhecimento gerado na universidade. Geralmente essas atividades são de cunho social, objetivando promover o desenvolvimento dos diversos segmentos da sociedade, principalmente os mais necessitados, realizar projetos e programas que possibilitem assegurar valores e igualdades sociais.

O Departamento Acadêmico de Informática (DAINF) da Universidade Tecnológica Federal do Paraná (UTFPR) Campus Pato Branco realiza atividades de extensão promovendo cursos de informática para a comunidade. Esses cursos visam promover inclusão digital e melhoria na empregabilidade, além de incentivar os participantes a ingressarem em cursos de graduação na área de informática ou na área de exatas de um modo geral.

Considerando esse contexto, este trabalho visa desenvolver um sistema *web* para a gestão dos cursos de extensão oferecidos pelo DAINF do Campus Pato Branco. A gestão desses cursos envolve dados de usuários e dos cursos como: instrutores (alunos voluntários dos cursos de graduação da UTFPR), professores envolvidos, turmas, data de início e término, cursos ofertados, entre outros. Os dados armazenados auxiliam na gestão das atividades e no controle de presença dos participantes dos cursos para a emissão de certificados, entre outros.

## **1.2 Objetivos**

O objetivo geral visa estabelecer a meta principal ou o propósito central da realização desse trabalho. Os objetivos específicos são metas ou ações intermediárias que auxiliam na realização do objetivo geral detalhando as etapas ou as tarefas necessárias para atingi-lo.

### 1.2.1 Objetivo Geral

Desenvolver um sistema *web* para gestão dos cursos de extensão ofertados pelo DAINF da UTFPR Campus Pato Branco.

### 1.2.2 Objetivos Específicos

- Contribuir para o suporte ao professor responsável na formação das equipes de instrutores e no gerenciamento dos cursos.
- Aprimorar o processo de gestão dos cursos de extensão, facilitando o acompanhamento e controle das informações necessárias para a emissão de certificados aos participantes.
- Possibilitar o controle de presença para um acompanhamento detalhado e efetivo dos alunos participantes dos cursos.

## 1.3 Justificativa

Para cada atividade desenvolvida na universidade é utilizada infraestrutura, equipamentos e ambientes. Além disso, há a participação direta e indireta de funcionários, professores e/ou alunos. Para realizar os cursos de extensão oferecidos pelo DAINF são utilizados laboratórios com computadores e projetores multimídia, kits como os de Arduino e de Lego Mindstorms e outros itens como materiais para montagem e manutenção de computadores, alunos (em sua maioria são voluntários) para ministrar os cursos e professores para coordenar cada um dos cursos, entre outros recursos.

Os instrutores dos cursos são alunos de graduação da UTFPR e possuem habilidades distintas. Assim, os alunos se candidatam para ministrar os cursos para os quais eles consideram que possuem conhecimento e/ou habilidades.

Os participantes devem atender determinadas condições para manter a vaga no curso, como, por exemplo, não faltar ao primeiro dia de aula, e não mais que três vezes durante o período de realização do curso. Ao final do curso os instrutores recebem um certificado de participação para que possam cumprir com as horas de atividades complementares exigidas para complementação da formação do aluno constante na matriz curricular dos cursos da UTFPR.

Esses e outros dados têm sido gerenciados no DAINF por meio de planilhas eletrônicas compartilhadas no Google Drive com os envolvidos para que a informação esteja ao alcance de todos. A comunicação com os instrutores para verificar as habilidades para cada curso é realizada por *e-mail* o que torna o processo de comunicação sujeito a falhas, pois nem todos acessam com frequência os *e-mails*.

Assim, a proposta desse trabalho visa possibilitar que a informação seja centralizada, tornando a gestão desses cursos organizada, proporcionando acesso mais rápido e confiável aos dados.

Para o desenvolvimento do sistema proposto foi utilizada a linguagem de programação Java, o *framework* Spring para o *back-end* e o Angular para o *front-end*. A escolha na utilização dessas tecnologias proporcionou uma arquitetura sólida, sendo capaz de lidar com demandas complexas e escalável, pois permite que o sistema se adapte e cresça conforme necessário, sem comprometer seu desempenho ou estabilidade.

## 2 EXTENSÃO UNIVERSITÁRIA

A extensão universitária pode ser definida como ação da universidade junto à comunidade visando o compartilhamento, com o público externo, do conhecimento produzido pelas atividades de ensino e de pesquisa desenvolvidas na instituição (UNIVERSIDADE..., 2019).

A extensão proporciona a articulação do conhecimento científico oriundo do ensino e da pesquisa de forma a contribuir para atender necessidades e interesses e colaborar na solução de problemas da comunidade na qual a universidade se insere. A extensão é uma forma de a universidade, por meio da ação do seu corpo docente, discente e técnicos administrativos, transformar a realidade social da sua comunidade. Fazendo, ainda, com que as pessoas que não participam das atividades efetivas de ensino, pesquisa e prestação de serviços também estejam integradas com a vida acadêmica.

A extensão também contribui para que os docentes e discentes estejam mais inteirados dos problemas, das necessidades e dos interesses da comunidade que os cerca. Assim, soluções para esses problemas e formas de atender interesses, necessidades e demandas podem ser providas por meio de ações de pesquisa e de ensino, como atividade acadêmica.

Silva (1997) ressalta que essa inserção ao expressar que a extensão universitária é, na verdade, uma forma de interação que deve existir entre a universidade e a comunidade na qual está inserida. Funciona como uma via de duas mãos, em que a universidade leva conhecimento e/ou assistência à comunidade e recebe informações sobre as reais necessidades da comunidade, anseios, aspirações, além do aprendizado proveniente do saber dessa comunidade. Há uma troca de conhecimento. A universidade aprende com a própria comunidade sobre os valores e a cultura dessa comunidade. Por fim, Silva (1997) destaca que a universidade, por meio da extensão, influencia e é influenciada pela comunidade, ou seja, possibilita troca de valores, conhecimento, saberes entre a universidade e o meio (SILVA, 1997).

Como forma de efetivar a extensão, o Plano Nacional de Educação (PNE), PNE 2001 – 2010, estabeleceu metas para o desenvolvimento da Extensão Universitária. Dessas metas destacam-se (BRASIL, 2001):

- Meta 21: Garantir, nas instituições de educação superior, a oferta de cursos de extensão, para atender as necessidades da educação continuada de adultos, com ou sem formação superior, na perspectiva de integrar o necessário esforço nacional de resgate da dívida social e educacional.
- Meta 23: Implantar o Programa de Desenvolvimento da Extensão Universitária em todas as Instituições Federais de Ensino Superior no quadriênio 2001-2004 e assegurar que, no mínimo, 10% do total de créditos exigidos para a graduação no ensino superior no país será reservado para a atuação dos alunos em ações extensionistas.

O Projeto de Lei nº 8.035 (BRASIL, 2011) propõe o Plano Nacional de Educação para o decênio 2011-2020 e contém novamente a Meta 23. Esse projeto foi transformado na Lei Ordinária 13.005/2014 (BRASIL, 2014) que aprova o Plano Nacional de Educação - PNE tem como uma de suas metas (BRASIL, 2014):

- Meta 12: elevar a taxa bruta de matrícula na educação superior para 50% (cinquenta por cento) e a taxa líquida para 33% (trinta e três por cento) da população de 18 (dezoito) a 24 (vinte e quatro) anos, assegurada a qualidade da oferta e expansão para, pelo menos, 40% (quarenta por cento) das novas matrículas, no segmento público.

Uma das estratégias para implementar essa meta é (BRASIL, 2014): “12.7 - assegurar, no mínimo, 10% (dez por cento) do total de créditos curriculares exigidos para a graduação em programas e projetos de extensão universitária, orientando sua ação, prioritariamente, para áreas de grande pertinência social”.

Essa estratégia desafia as instituições de ensino superior brasileiras a repensarem suas concepções e práticas extensionistas, o currículo dos seus cursos e a própria universidade (IMPERATORE; PEDDE; IMPERATORE, 2015).

Ainda em 2012, o Fórum de Pró-Reitores de Extensão das Universidades Públicas Brasileiras (FORPROEX) ressaltou a necessidade de indissociabilidade entre ensino, pesquisa e extensão e o impacto da extensão na formação do estudante. Nesse Fórum ficou evidenciado que a participação dos estudantes nas ações de extensão universitária deve estar sustentada em iniciativas que viabilizem a flexibilização curricular e a integralização dos créditos obtidos nas ações de extensão universitária (FORUM..., 2012). Esse Fórum foi criado em 1987.

Como forma de contribuir para a discussão de como implementar e efetivar a curricularização da extensão, docentes da Universidade Tecnológica Federal do



Paraná vêm propondo ações e realizado projetos. Uma dessas propostas é de autoria de Teleginski e Porto Alegre (2014). Essa proposta de curricularização de extensão tem como base as próprias disciplinas dos cursos. Nessa proposta, os coordenadores dos cursos são os responsáveis por identificar os conteúdos das disciplinas que possuem algum viés extensionista. Esses são os conteúdos que podem ser desenvolvidos em benefício da comunidade ou da instituição. O professor seria o responsável por identificar em comunidades ou instituições demandas que possam ser supridas a partir dos conhecimentos que os alunos adquirem na sua disciplina de graduação.

### 3 MATERIAIS E MÉTODO

Este capítulo está subdividido, inicialmente, em duas seções, sendo uma para os materiais, que se refere às ferramentas e tecnologias, e outra para o método, que é a sequência das principais atividades realizadas para a modelagem e o desenvolvimento do sistema. As principais ferramentas utilizadas no desenvolvimento do projeto são o *Framework* Spring e o *Framework* Angular.

**Quadro 1 - Lista de ferramentas e tecnologias**

<b>Ferramenta / Tecnologia</b>	<b>Versão</b>	<b>Disponível em:</b>	<b>Finalidade</b>
Angular	14.3.0	<a href="https://angular.io/">https://angular.io/</a>	Desenvolvimento <i>web</i> .
Angular Material	8.0.2	<a href="https://material.angular.io/">https://material.angular.io/</a>	Biblioteca de componentes.
Intellij Idea Ultimate	2022.3	<a href="https://www.jetbrains.com/idea">https://www.jetbrains.com/idea</a>	IDE de desenvolvimento.
Java	19	<a href="http://www.java.com/pt_BR/download/win10.jsp">http://www.java.com/pt_BR/download/win10.jsp</a>	Linguagem de Programação.
pgAdmin 4	4.14	<a href="https://www.pgadmin.org">https://www.pgadmin.org</a>	Ferramenta para administração do banco de dados.
PostgreSQL	10	<a href="https://www.postgresql.org">https://www.postgresql.org</a>	Banco de dados.
Spring Boot	2.7.1	<a href="https://projects.spring.io/spring-boot/">https://projects.spring.io/spring-boot/</a>	<i>Framework</i> para criação de aplicações.
Spring Data	2.0.0	<a href="https://projects.spring.io/spring-data-jpa/">https://projects.spring.io/spring-data-jpa/</a>	<i>Framework</i> de implementação de repositórios.
Spring Security	4.2.3	<a href="https://projects.spring.io/spring-security/">https://projects.spring.io/spring-security/</a>	<i>Framework</i> de autenticação e segurança da aplicação.
Visual Paradigm	15	<a href="https://www.visual-paradigm.com/">https://www.visual-paradigm.com/</a>	Modelagem do sistema.

**Fonte: Autoria própria (2023).**

#### 3.1 Materiais

Essa seção apresenta uma breve descrição sobre as principais ferramentas que foram utilizadas no desenvolvimento do sistema proposto.

##### 3.1.1 Spring Framework

O Spring é um conjunto abrangente de projetos que oferecem soluções para uma ampla gama de desafios enfrentados pelos programadores no desenvolvimento

de aplicações Java com facilidade e versatilidade, e atualmente o Spring conta com diversos projetos, principalmente utilizados em aplicações web.

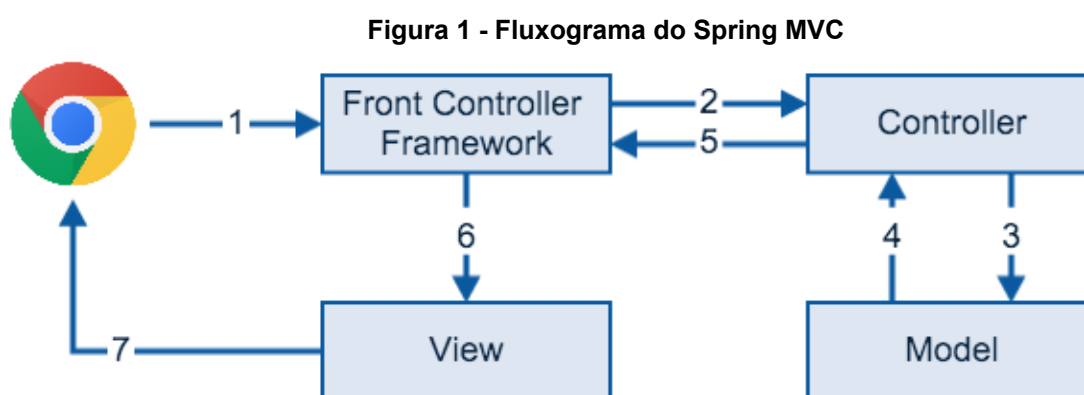
Com o Spring, é possível ter acesso a um ecossistema rico de ferramentas e bibliotecas que simplificam o desenvolvimento de aplicações. Ele oferece recursos como injeção de dependência, controle transacional, segurança, acesso a banco de dados, gerenciamento de cache e outras vantagens.

O Spring possui diversos projetos que atendem a diferentes necessidades, sendo amplamente utilizado em aplicações web. Entre os principais projetos do Spring estão: Spring Boot, Spring MVC, Spring Data e Spring Security. A seguir, será apresentada uma breve descrição sobre alguns dos principais projetos do Spring.

### 3.1.2 Spring MVC

O Spring MVC – sendo MVC de *Model*, *View* e *Controller* - é o *framework* que auxilia no desenvolvimento de aplicações web mais robustas, flexíveis e com uma clara separação de responsabilidades nos papéis do tratamento da requisição.

A ideia do MVC é separar a responsabilidade de cada componente dentro de uma aplicação para facilitar a manutenção do código. O *controller* trata da requisição dos dados, o *model* é responsável pela persistência e consulta de dados no banco de dados e a *view* transforma o HTML em dados para que o usuário visualize a aplicação. A Figura 1 apresenta um esquema de como essas três partes interagem com o navegador e o *framework* de controle do *front-end* da aplicação.



Fonte: Afonso (2017, p.s.n.).

### 3.1.3 Spring Boot

Afonso e Normandes (2017) relatam que o Spring Boot trouxe agilidade e a possibilidade de focar nas funcionalidades da aplicação com o mínimo de configuração. Isso porque o Spring Boot analisa o projeto e o configura automaticamente.

### 3.1.4 Spring Security

O Spring Security é um *framework* de controle de acesso, que permite decidir quem acessará o quê no sistema.

O controle de acesso do Spring Security pode ser aplicado em dois níveis: nas requisições que verificam aplicação, no caso do projeto web e na invocação de métodos *beans* gerenciados pelo contexto Spring.

O Spring Security é composto pelos seguintes módulos (AFONSO, 2017):

- spring-security-core
- spring-security-config
- spring-security-web
- spring-security-taglibs
- spring-security-remoting
- spring-security-ldap
- spring-security-cas
- spring-security-openid

### 3.1.5 Spring Data

O Spring Data é um projeto que visa simplificar o acesso às tecnologias de armazenamento de dados, sejam elas relacionais (MySQL, PostgreSQL, etc.) ou não (MongoDB, Redis, etc.). O Spring Data já possui vários projetos incorporados, como (AFONSO, 2017):

- Spring Data Commons
- Spring Data JPA
- Spring Data Gemfire
- Spring Data KeyValue

- Spring Data LDAP
- Spring Data MongoDB
- Spring Data REST
- Spring Data Redis
- Spring Data for Apache Cassandra
- Spring Data for Apache Solr

O Spring Data JPA auxilia os desenvolvedores padronizando o uso de algumas funcionalidades e, assim, reduzindo o esforço de implementação. Um exemplo disso é a implementação padrão para repositórios, incluindo métodos como *save*, *delete*, *findOne*, entre outros.

### 3.1.6 Angular

O Angular é um *framework* criado pelo Google, escrito em JavaScript e projetado para a criação de aplicações *web*, *mobile* ou *desktop* do modelo *Single Page Application* (SPA), utiliza a linguagem TypeScript, Dart ou JavaScript para desenvolvimento. No caso das linguagens TypeScript e Dart elas são transformadas automaticamente em JavaScript pelo Angular.

A plataforma do Angular é *open-source* e possui uma grande comunidade. A primeira versão gerada foi o AngularJS, mas, a partir da segunda versão Angular 2 a plataforma foi totalmente reformulada para adaptar-se às novas tecnologias do mercado. Atualmente o Angular está em sua versão 8 e o projeto AngularJS está sendo descontinuado (AFONSO, 2019).

A partir do Angular 2 o versionamento começou a ser trabalhado no modelo *Semantic Versioning* (SEMVER). Os principais elementos do Angular são: componentes, *templates*, diretivas, roteamento, módulos, serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas, como a de executar os testes unitários de uma aplicação. Atualmente as principais bibliotecas de componentes para o Angular são: PrimeNG, Angular Material e Ng-Bootstrap. O Angular facilita a criação de bibliotecas por ser baseado em componentes (AFONSO, 2019).

### **3.2 Método**

O método consiste nas atividades de levantamento e modelagem de requisitos. O método utilizado neste trabalho foi o processo sequencial linear de Pressman (2008) que define as fases de levantamento de requisitos, análise e projeto e desenvolvimento do sistema . Neste trabalho foram desenvolvidas as duas primeiras etapas, portanto, são as apresentadas a seguir.

Para o levantamento dos requisitos foi realizada entrevista com professores do DAINF que trabalham na organização e na coordenação dos cursos de extensão. A partir disso, foram levantadas as principais informações que os professores necessitam para fazer a gestão dos cursos ofertados e, assim, definir as funcionalidades do sistema.

Na sequência, foi realizada a análise e projeto do sistema, definindo os requisitos como funcionais e não funcionais, os casos de uso e o diagrama de entidade e relacionamento do banco de dados.

## 4 RESULTADOS

Este capítulo apresenta os resultados da realização deste trabalho abrangendo a definição do escopo, a modelagem do sistema, a apresentação e a implementação do sistema. A definição do escopo refere-se à delimitação das funcionalidades e características desenvolvidas no sistema. A modelagem do sistema consiste na criação dos diagramas de casos de uso e de entidade e relacionamento do banco de dados. Essas representações ajudam a compreender e comunicar de forma clara e precisa as diferentes partes e interações do sistema, como os componentes, as funcionalidades e os relacionamentos entre eles. A apresentação do sistema consiste na exposição das telas contendo aspectos e funcionalidades do sistema. A implementação do sistema refere-se à codificação e desenvolvimento das funcionalidades de acordo com as especificações definidas anteriormente.

### 4.1 ESCOPO DO SISTEMA

O sistema é *web* e visa auxiliar na gestão dos cursos de extensão oferecidos pelo DAINF da UTFPR, Campus Pato Branco.

O sistema tem o objetivo principal de gerenciar a realização de cursos de extensão, no sentido de divulgar esses cursos para a comunidade acadêmica para que alunos possam candidatar-se para serem instrutores. Um curso tem um professor responsável e pode ter mais de um instrutor, compondo uma equipe para a realização de cada curso. Acadêmicos cadastrados no sistema podem contatar o professor externamente para solicitar a candidatura para serem instrutores dos cursos e o professor responsável pelo referido curso faz a validação dos candidatos que efetivamente comporão a equipe. Critérios como habilidades e participação em outros cursos podem ser adotados para a composição da equipe entre os candidatos.

O sistema possibilitará cadastrar e manter dados de professores e de instrutores, além de cursos, equipes e turmas. Um curso inicialmente será ministrado por uma equipe (que deve ser compostas por um ou mais professores responsáveis e instrutores), ocorre de uma data inicial até uma data final, em determinado local e ambiente.

O professor cadastrado no sistema poderá cadastrar cursos, turmas, alunos, habilidades e vínculos de instrutores a cursos. Porém, o cadastro de novos professores deve ser validado por um usuário administrador, diferentemente dos instrutores, que podem cadastrar-se sem validação.

Ao formar uma equipe para ministrar o curso, o professor pode indicar o tipo de transporte que a equipe irá utilizar para deslocar-se até o local do curso. Os cursos de extensão que são ministrados para a comunidade pelo DAINF têm sido, em sua grande maioria, realizados aos sábados. A indicação do meio de transporte tem como objetivo simplificar a organização de sistema de carona, que sejam providenciados vales-transportes ou reembolso com despesas de combustível para aqueles que optarem por utilizar veículos particulares.

O sistema também possui um controle de chamada dos alunos que realizam os cursos, facilitando, assim, a obtenção dos dados para a emissão de certificados. Além do acompanhamento das presenças para contato com responsáveis ou outras ações.

Uma tela de listagem por turma realizará a mostragem de controle de presença por aluno, para que vejam os que atingiram o percentual mínimo de presença para a obtenção do certificado. No cadastro do curso são informados os campos de nome e o conteúdo do curso, da carga horária, da data de início e final. Esses dados podem ser utilizados para a emissão dos comprovantes de participação.

## 4.2 MODELAGEM DO SISTEMA

Os requisitos para o sistema são apresentados nos quadros a seguir. Os requisitos funcionais que estão no Quadro 2 se referem às funcionalidades oferecidas pelo sistema resultantes da interação do usuário.

**Quadro 1 – Requisitos funcionais**

	<b>Requisito</b>	<b>Descrição</b>
0 1	Manter usuário	Usuários do sistema, com os perfis. a) administrador – pré-cadastrado no sistema. Pode alterar a sua senha e possui permissão de manter usuários e habilitar usuários professor. b) professor - usuário vinculado como responsável por cursos. Pode cadastrar cursos, áreas e outros. Ao realizar o seu cadastro



		indica que é professor e o cadastro precisa ser validado pelo administrador. c) instrutor – responsável por ministrar os cursos. Faz o seu próprio cadastro indicando que é instrutor. Esse usuário pode candidatar-se a ser instrutor de cursos e realizar as ações (registro de presença e outros) relacionadas às turmas nas quais ele está vinculado.
0 2	Manter professor	Cadastro dos professores no sistema, realizado pelo próprio usuário, que indica o tipo como professor e é validado pelo administrador.
0 3	Manter instrutor	Cadastro de instrutores. Os instrutores realizam o seu próprio cadastro e não necessitam de validação.
0 4	Validar usuário	Administrador valida cadastro de professor e pode excluir cadastro de usuários.
0 5	Manter habilidades	Cadastro de habilidades que se referem a conhecimentos que indicam quais cursos os instrutores se consideram aptos a ministrar. Exemplos de conhecimento: programação C, Arduino e <i>HyperText Markup Language</i> (HTML).
0 6	Manter habilidades instrutores	Aluno indica habilidades para compor o seu perfil. Essas habilidades indicam, por exemplo, em quais cursos ele pode atuar.
0 7	Manter cursos	Cursos que serão ofertados para a comunidade.
0 8	Manter equipes cursos	A equipe que ministrará um determinado curso. Essa equipe é composta por um responsável pelo curso (um usuário que é professor) e um ou mais usuários alunos.
0 9	Manter turmas dos cursos	Um curso será ministrado por uma equipe e ocorrerá entre uma data inicial e final, em um determinado horário, local e ambiente (sala).
1 0	Manter tipos de cursos	Um curso é de um tipo ou uma área de conhecimento. Exemplos: HTML, linguagem Java, Arduino, etc. O nível do curso é indicado no cadastro de realização do curso. Os níveis podem ser pré-definidos ou um campo texto com a sugestão de nível, exemplo: básico, intermediário, avançado. Um campo texto também estará disponível para permitir o usuário informar outra classificação que não as sugeridas.
1 1	Manter alunos	Alunos que pertencem às turmas dos cursos.
1 2	Manter alunos e cursos	Compor as equipes com alunos que será adicionada a uma turma de um determinado curso.
1 3	Manter chamada	Realização do controle de presença e conteúdo ministrado no curso.
1 4	Manter tipo transporte	O tipo de transporte que será utilizado pela equipe. Exemplos: ônibus circular (indica a necessidade de vale transporte), ônibus (indica a necessidade de pagamento de passagem, deslocamento para outra cidade, por exemplo), carro próprio, moto própria, sem despesa (deslocamento a pé, bicicleta ou outro), carona.
1 5	Manter equipes e tipo de transporte	A equipe e o tipo de transporte utilizado para deslocamento até o local do curso para uma determinada turma.
1 6	Vincular instrutores a curso	O professor responsável pelo respectivo curso pode incluir instrutores como parte da equipe de um curso. Nesse momento o

		instrutor é vinculado à equipe que ministrará o curso.
1 7	Manter cursos de graduação	Os cursos de graduação que os instrutores selecionam ao cadastrar-se. Um instrutor, normalmente, é aluno de um curso de graduação.

**Fonte: Autoria própria (2023).**

Os requisitos não funcionais, apresentados no Quadro 3, se referem às regras de negócio, segurança e outros que atuam sobre os requisitos funcionais e sobre o sistema como um todo.

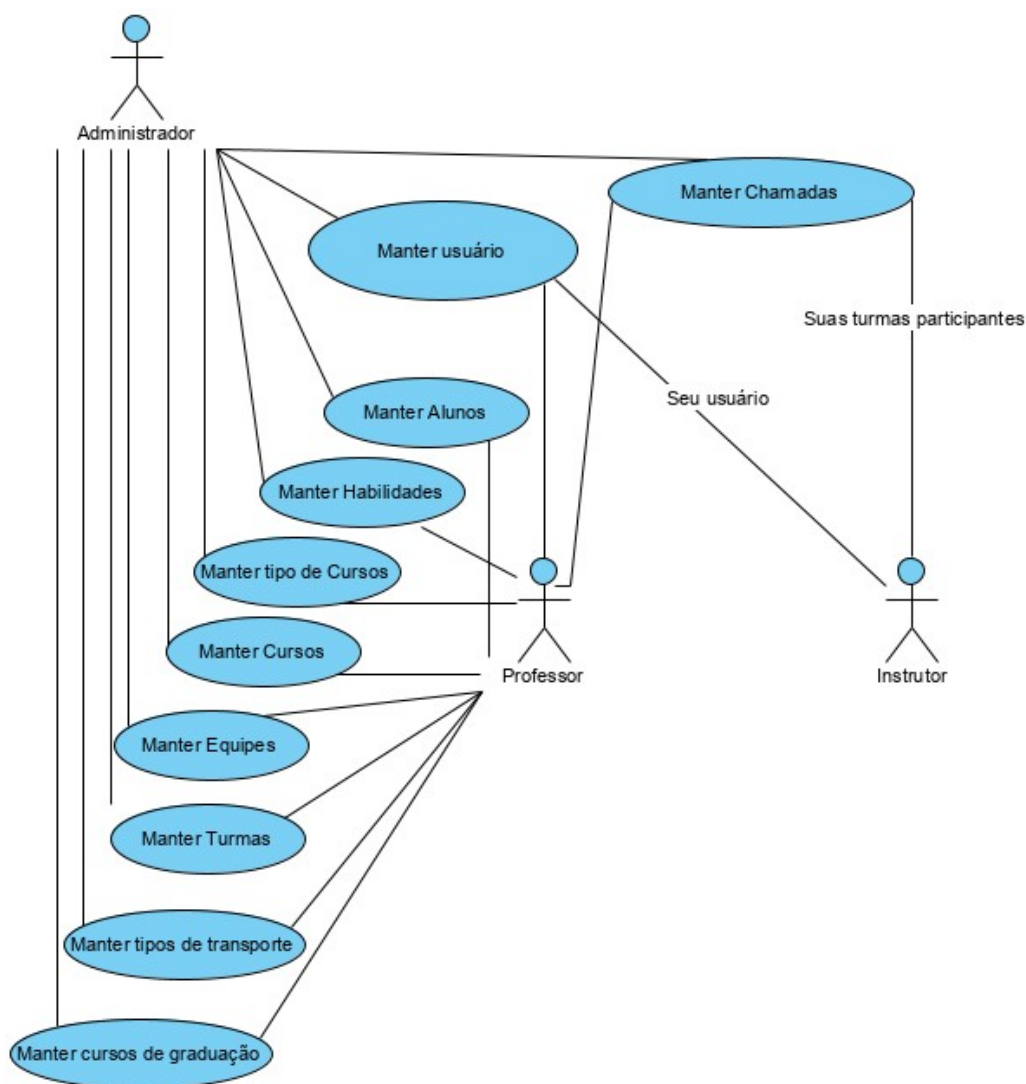
**Quadro 1 – Requisitos não funcionais**

	<b>Nome</b>	<b>Descrição</b>
01	Efetuar <i>login</i>	O sistema validará a autenticação para acesso do usuário no sistema por meio de <i>login</i> (sendo esse o <i>e-mail</i> ) e senha válidos a partir do cadastro do usuário no sistema.
02	Campos de preenchimento obrigatório	Os campos que são de preenchimento obrigatório serão validados por meio de funções do sistema.
03	Campos com máscaras de entrada	Os campos que possuem caracteres especiais, e que não serão armazenados no banco de dados, serão validados por meio de máscaras de entrada nos respectivos campos.
04	Vínculo entre dados	O sistema também não deve permitir a exclusão de dados vinculados. Por exemplo: excluir tipo de transporte que esteja vinculado a cadastro de instrutores e meio de transporte que utilizam para deslocar-se ao local do curso.
05	Autenticação de usuários	O usuário professor somente poderá autenticar-se no sistema após o seu cadastro ter sido validado.

**Fonte: Autoria própria (2023).**

Os requisitos funcionais estão organizados em casos de uso apresentados na Figura 2. Pela figura e descrição do primeiro requisito funcional, o sistema possuirá três perfis de acesso, que definem os usuários. O Administrador com a função de manter usuários, permitindo incluir, excluir, alterar e validar usuários que fizeram o próprio cadastro. Na Figura 2, o rótulo “seu usuário” indica que o referido ator (instrutor ou professor) faz apenas o seu cadastro de usuário. No caso do ator professor, o cadastro precisa ser validado pelo administrador do sistema. O usuário instrutor não precisa ter o seu cadastro validado. Ele se candidata para ser instrutor de curso, mas para efetivamente fazer parte da equipe que ministrará o curso é necessário que haja validação do professor responsável pelo referido curso.

Figura 1 - Casos de uso



Fonte: Autoria própria (2023).

A seguir é apresentada a expansão dos casos de uso apresentados na Figura 2. Os casos de uso indicados como “manter”, são expandidos em conjunto e por operação, ou seja, a expansão “Inserir” se refere à operação de inclusão de todos os casos de uso “manter”.

#### Quadro 1 – Operação para incluir registros no banco de dados

**Caso de uso:**

Inserir (refere-se à operação de inclusão nos casos de uso identificados como “manter” e “validar” para o usuário administrador).

**Descrição:**

Inclusão dos dados cadastrais no sistema.

**Atores:**

Administrador, professor ou instrutor, no que couberem as suas permissões.

**Pré-condição:**

Não há.

<p><b>Sequência de Eventos:</b></p> <ol style="list-style-type: none"> <li>1. Ator acessa a tela para realizar os cadastros necessários.</li> <li>2. Ator preenche os dados e solicita a inclusão dos dados no sistema. O sistema inclui o registro no banco de dados e apresenta uma mensagem de confirmação da operação.</li> </ol> <p><b>Pós-Condição:</b> Registro inserido no banco de dados.</p>	
<b>Nome do fluxo alternativo (extensão)</b>	<b>Descrição</b>
1. Campos obrigatórios não preenchidos.	<ol style="list-style-type: none"> <li>1.1. O ator não preenche campos obrigatórios e clica na opção para salvar.</li> <li>1.2. O sistema valida as informações e exibe uma mensagem informando que campos obrigatórios não foram preenchidos.</li> <li>1.3. O sistema permanece na tela de inclusão, com os campos que já haviam sido preenchidos e destacando os campos sem preenchimento.</li> </ol>
2. Campos preenchidos com formato inválido.	<ol style="list-style-type: none"> <li>2.1. O ator preenche os campos de forma incorreta e clica na opção para salvar.</li> <li>2.2. O sistema valida as informações e exibe uma mensagem informando que os dados foram preenchidos de forma incorreta.</li> <li>2.3. O sistema permanece na tela de inclusão com os dados que já haviam sido informados e destacando os campos com dados no formato inválido.</li> </ol>

**Fonte: Autoria própria (2023).**

O Quadro 5 apresenta a expansão do caso de uso para alterar um registro do banco de dados.

#### **Quadro 1 – Operação para alterar registros no banco de dados**

<p><b>Caso de uso:</b> Alterar (refere-se à operação de alteração nos casos de uso identificados como “manter”).</p> <p><b>Descrição:</b> Alteração dos dados cadastrais no sistema.</p> <p><b>Atores:</b> Administrador, professor ou instrutor, no que couberem as suas permissões.</p> <p><b>Pré-condição:</b> Registro estar cadastrado no sistema.</p> <p><b>Sequência de Eventos:</b></p> <ol style="list-style-type: none"> <li>1. O ator acessa a tela para visualização de dados do registro.</li> <li>1. O sistema apresenta o registro selecionado para alteração.</li> <li>2. Ator altera os dados do registro e clica no botão Salvar.</li> <li>3. O sistema valida as informações e as salva no mesmo registro da tabela do banco de dados.</li> </ol> <p><b>Pós-Condição:</b> Registro alterado no banco de dados.</p>	
<b>Nome do fluxo alternativo (extensão)</b>	<b>Descrição</b>
1. Campos obrigatórios não	1.1. O ator não informa dados obrigatórios e clica na opção para salvar.

preenchidos.	1.2. O sistema valida que não foram informados dados obrigatórios e exibe uma mensagem para o usuário sem salvar os dados no banco de dados. 1.3. O sistema permanece na tela de alteração mantendo os dados informados anteriormente.
2. Campos preenchidos com formato inválido.	2.1. O ator altera os dados com um valor em formato inválido e na opção para salvar. 2.2. O sistema valida informando que os dados não estão no formato esperado e exibe uma mensagem ao usuário sem salvar o registro no banco de dados. 2.3. O sistema permanece na tela de inclusão mantendo as alterações realizadas.

**Fonte: Aatoria própria (2023).**

O Quadro 6 apresenta a expansão do caso de uso para excluir um registro do banco de dados.

#### Quadro 1 – Operação para excluir registros no banco de dados

<p><b>Caso de uso:</b> Excluir (refere-se à operação de exclusão nos casos de uso identificados como “manter”).</p> <p><b>Descrição:</b> Exclusão de registros no sistema.</p> <p><b>Atores:</b> Administrador, professor ou instrutor, no que couberem as suas permissões.</p> <p><b>Pré-condição:</b> Registro estar incluso no sistema.</p> <p><b>Sequência de Eventos:</b></p> <ol style="list-style-type: none"> <li>Ator acessa a tela para a exclusão do registro.</li> <li>Ator seleciona o registro para excluí-lo e clica no botão Excluir.</li> <li>O sistema exclui o registro do banco de dados e exibe a lista de cadastros atualizada.</li> </ol> <p><b>Pós-Condição:</b> Registro excluído do banco de dados.</p>	
<b>Nome do fluxo alternativo (extensão)</b>	<b>Descrição</b>
1. Exclusão de registro com vínculos no sistema.	1.1. Ator seleciona um registro que possui vínculos no sistema e clica na opção para excluir. 1.2 O sistema verifica que o registro possui vínculos, não exclui e exibe mensagem de alerta ao usuário.

**Fonte: Aatoria própria (2023).**

O Quadro 7 apresenta a expansão do caso de uso para consultar informações do banco de dados.

#### Quadro 1 – Operação de consultar registros no banco de dados

<p><b>Caso de uso:</b> Consultar (refere-se à operação de consulta nos casos de uso identificados como</p>
--

“manter” e “Buscar serviço” e se aplica às consultas oferecidas pelo sistema e que não estão vinculadas a casos de uso, como a de verificar os serviços que um determinado usuário presta ou o usuário consultar os serviços que estão a ele vinculados).

**Descrição:**

Consultar informações cadastradas no sistema.

**Atores:**

Administrador, professor ou instrutor, no que couberem as suas permissões.

**Pré-condição:**

Registro estar incluso no sistema.

**Sequência de Eventos:**

1. Ator acessa a tela para visualização de dados já cadastrados.
1. Ator indica quais dados ele pretende consultar por meio de filtros.
2. O sistema exibe os dados da consulta ao usuário.

**Pós-Condição:**

Dados apresentados para o usuário.

**Fonte: Aatoria própria (2023).**

A vinculação de instrutores a cursos, realizada pelo professor responsável pelo referido curso, fará com que aqueles instrutores componham a equipe que realizará o curso.

**Quadro 1 – Vincular instrutores a curso**

**Caso de uso:**

Vincular instrutores

**Descrição:**

O professor responsável pelo curso realizará o vínculo de uma equipe composta de instrutores ao curso. Esses instrutores comporão a equipe que realizará o curso. No cadastro da equipe o professor responsável adicionará instrutores para sua composição e após isso adicionará essa equipe formada para o curso.

**Atores:**

Administrador/Professor.

**Pré-condição:**

Haver usuários cadastrados para compor uma equipe

**Sequência de Eventos:**

1. Ator realiza vínculos de usuários para compor uma equipe.
2. Ator acessa a tela para visualização do curso adicionado.
3. Ator seleciona a equipe para ministrar o curso.
4. O sistema registra os dados no curso.

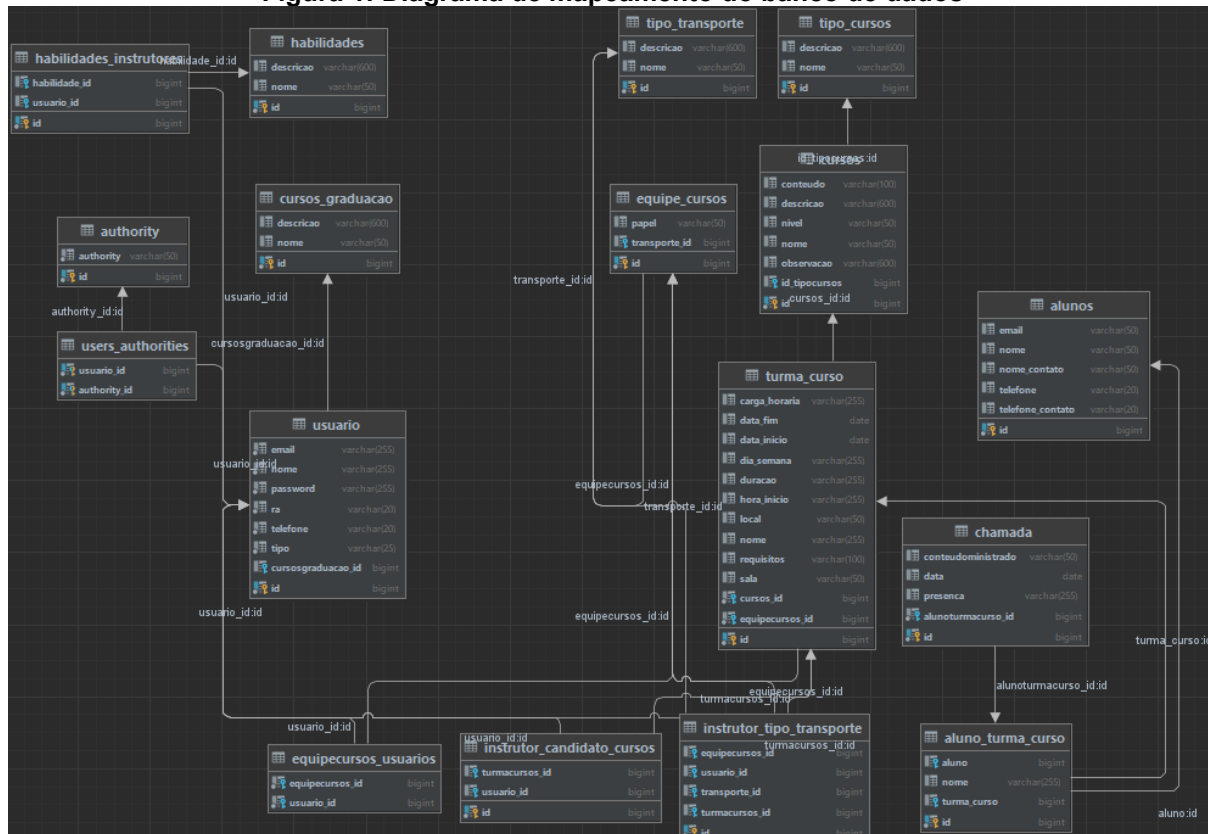
**Pós-Condição:**

Dados inseridos no banco de dados do sistema.

**Fonte: Aatoria própria (2023).**

As tabelas (entidades do banco de dados) e os relacionamentos entre elas são apresentados na Figura 3.

Figura 1: Diagrama de mapeamento do banco de dados



Fonte: Autoria própria (2023).

Os usuários cadastrados podem ser instrutores ou professores, como o sistema possui um único usuário com permissões de administrador, não há um cadastro. O campo tipo na tabela de Usuários indica se o usuário é administrador, professor ou instrutor. Os campos chave estrangeira dessa tabela estão ou apenas um preenchido (se o usuário é professor ou instrutor) ou nenhum se o usuário é administrador.

A tabela de instrutores possui um campo do tipo chave estrangeira para indicar o curso de graduação do instrutor. Esse campo pode não ser preenchido no caso de o instrutor não pertencer a um curso de graduação, ser técnico administrativo ou professor, por exemplo. A tabela que vincula instrutores e habilidades indica as habilidades que o instrutor escolhe para definir o seu perfil, indicando suas competências e auxiliando na escolha dos instrutores que se candidatam para compor a equipe de realização de um curso.

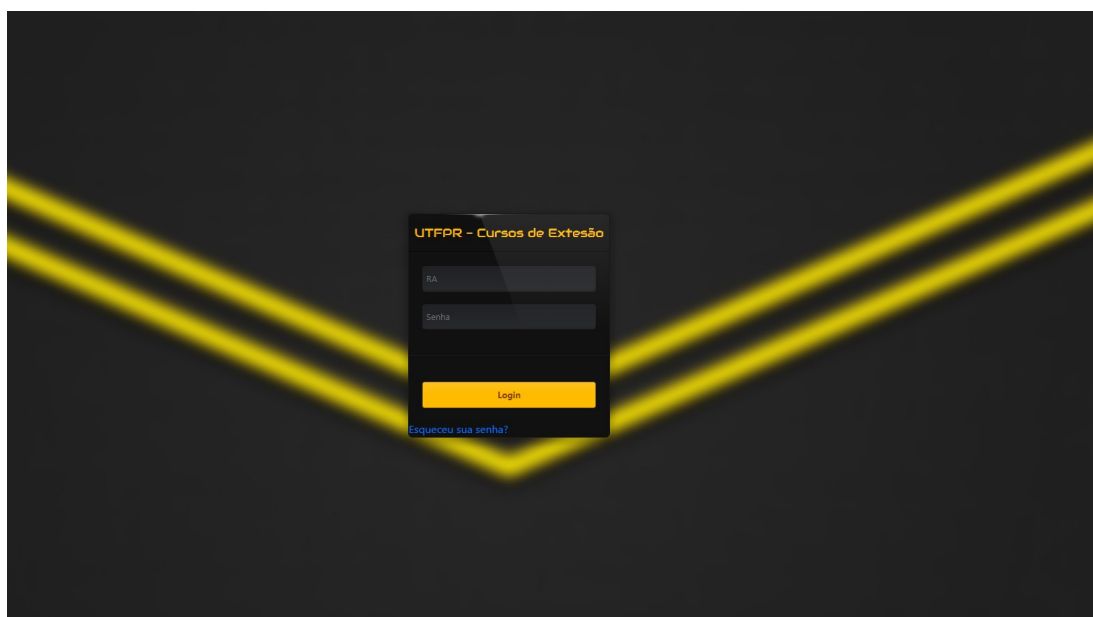
A indicação do tipo de transporte auxiliará na definição dos recursos para prover o deslocamento dos instrutores até o local de realização dos cursos. Indicando, por exemplo, se é necessário providenciar vale-transporte ou valor para o combustível.

As turmas dos cursos indicam os alunos de cada curso, sendo, assim, possível realizar controle de frequência nos cursos, podendo ser esses dados utilizados para a emissão de certificados e comprovantes, entre outros

### 4.3 Apresentação do Sistema

Para se ter acesso ao sistema, o usuário deve autenticar-se fornecendo seu Registro Acadêmico(RA) e a senha cadastrada. Esse processo é realizado por meio da tela de autenticação apresentada na Figura 4.

Figura 1 - Tela de autenticação



Fonte: Autoria própria (2023).

Caso o usuário seja um instrutor ou professor e ainda não tiver realizado no sistema o cadastro de usuário para autenticação, ele poderá solicitar a criação de uma conta para outro professor ou administrador do sistema. A conta deverá ser criada utilizando a tela de cadastro apresentada na Figura 5. Nessa tela, o usuário



deverá informar os dados do seu RA, senha, nome, e-mail, contato e o curso de graduação que frequenta, e seu tipo (administrador, professor ou instrutor).

**Figura 2 - Cadastro de Usuário**

Home | Usuários | Turmas | Equipes | Tipo de Cursos | Graduações | Habilidades | Transportes | Alunos | Chamada | Logout

RA\* | Senha | Tipo\* ▼

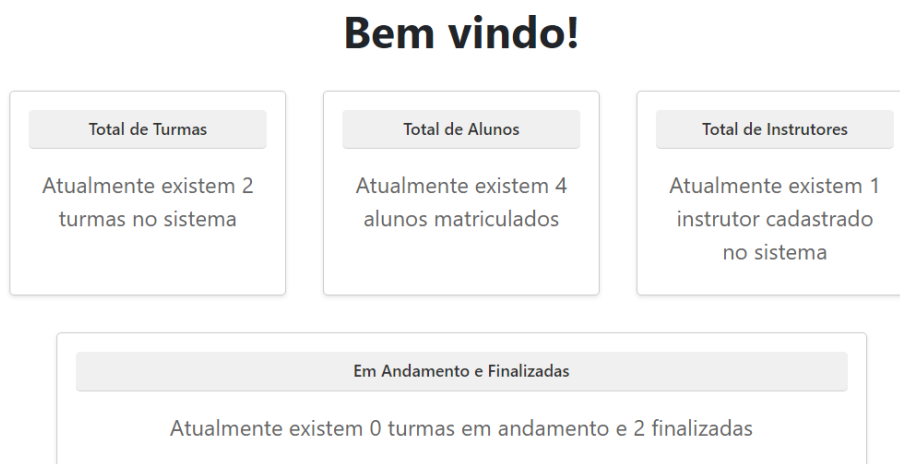
Nome\* | E-mail\* | Contato\* | Curso Graduação\* ▼

Cadastrar | Cancelar

**Fonte: Autoria própria (2023).**

Após autenticar-se no sistema, o usuário será redirecionado para a tela inicial, apresentada na Figura 6, onde terá acesso a uma interface gráfica que apresenta de forma visual e resumida as principais informações, métricas e dados relevantes do sistema. Essas informações incluem o total de turmas, de alunos matriculados, de instrutores cadastrados bem como o número de turmas em andamento e finalizadas. Essas informações fornecem um panorama geral das atividades e progresso do sistema.

**Figura 3 – Tela de Dashboard**



**Fonte: Autoria própria (2023).**

O leiaute padrão do sistema é composto de seções distintas. Essas partes incluem o corpo do *leiaute*, representado pelo elemento “body”, que exibe o conteúdo principal da página. Outro fragmento é a barra de navegação, representada pelo elemento “*navbar*” que contém um menu superior com *links* para acesso às diferentes páginas do sistema, o botão de saída (*logout*) e a opção de retornar para a página principal (*home*) ao clicar no primeiro título da barra. A Figura 7 apresenta a estrutura do *leiaute* padrão e também lista os usuários cadastrados no sistema.

**Figura 4 - Tela de listagem de usuário**

The screenshot shows a user management interface. At the top is a dark navigation bar with links: Home, Usuarios, Turmas, Equipes, Tipo de Cursos, Graduações, Habilidades, Transportes, Alunos, Chamada, and a Logout button. Below the navigation bar is the title 'Usuário' and a '+ Adicionar novo' button. A 'Filtro' input field is present. The main content is a table with the following data:

No	Nome	Email	RA	Tipo	Editar	Deletar
1	Jhony Sganzerla	admin@admin.com	123456789	administrador		
3	Instrutor	instrutor@instrutor.com	a12356465	instrutor		
2	Professor	professor@professor.com	123123346	professor		

**Fonte: Autoria própria (2023).**

A tela de listagem de usuários apresentada na Figura 7, pode ser acessada por meio do menu superior, ao clicar no link “Usuário” do menu superior. Nessa tela, todos usuários podem alterar seu perfil, sendo que, usuários com perfil de professores ou de administradores podem cadastrar novos usuários e fazer a manutenção dos existentes. No campo “Filtro”, localizado nessa tela, é possível realizar buscas utilizando qualquer dado presente na listagem, facilitando a localização de usuários específicos.

No canto superior esquerdo do corpo da página, encontra-se o botão “Adicionar novo”, que permite que usuários com perfil de professor ou administrador possam cadastrar novos usuários. Ao clicar nesse botão, o usuário é redirecionado para a tela de cadastro de usuários, como mostrado na Figura 5.

A tela de cadastro de usuários é submetida a uma validação de seus campos, garantindo que todos os campos obrigatórios sejam preenchidos. Se algum campo obrigatório não for preenchido, o cadastro não é considerado válido e uma mensagem de erro será exibida abaixo do campo, como mostrado na Figura 8. A validação dos campos é importante para garantir que todas as informações essenciais sejam fornecidas e que o cadastro seja concluído com sucesso.

**Figura 5 - Cadastro de usuário com campos inválidos**

A imagem mostra a interface de usuário para o cadastro de um novo usuário. No topo, há um menu de navegação com links para Home, Usuários, Turmas, Equipes, Tipo de Cursos, Graduações, Habilidades, Transportes, Alunos e Chamada, além de um botão Logout. O formulário principal contém os seguintes campos:

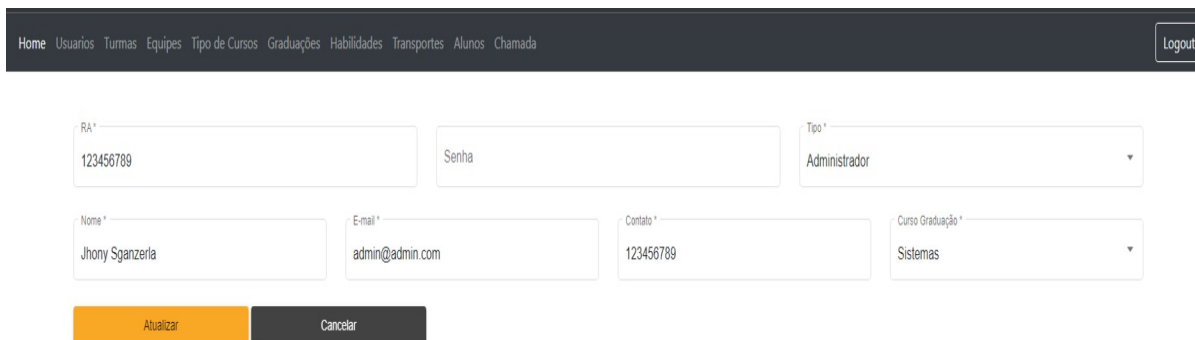
- RA \*: Campo de texto com o valor "123".
- Senha \*: Campo de texto com caracteres ocultos por pontos.
- Tipo \*: Menu suspenso com a opção "Administrador" selecionada.
- Nome \*: Campo de texto com uma borda vermelha e a mensagem "Campo obrigatório" abaixo dele.
- E-mail \*: Campo de texto com uma borda vermelha e a mensagem "Campo obrigatório" abaixo dele.
- Contato \*: Campo de texto com uma borda vermelha e a mensagem "Campo obrigatório" abaixo dele.
- Curso Graduação \*: Menu suspenso com uma borda vermelha e a mensagem "Campo obrigatório" abaixo dele.

Na base do formulário, há dois botões: "Cadastrar" (em amarelo) e "Cancelar" (em cinza escuro).

**Fonte: Autoria própria (2023).**

As telas que exibem uma listagem de informações, como a tela de usuário mostrada na Figura 7, também incluem um botão de edição representado pelo ícone de um lápis. Ao clicar nesse botão, o usuário é redirecionado para a tela de cadastro de usuário apresentada na Figura 5. Nessa tela, os campos são preenchidos automaticamente com os valores do usuário em edição, permitindo que sejam realizadas atualizações conforme a necessidade como exemplificado na Figura 9.

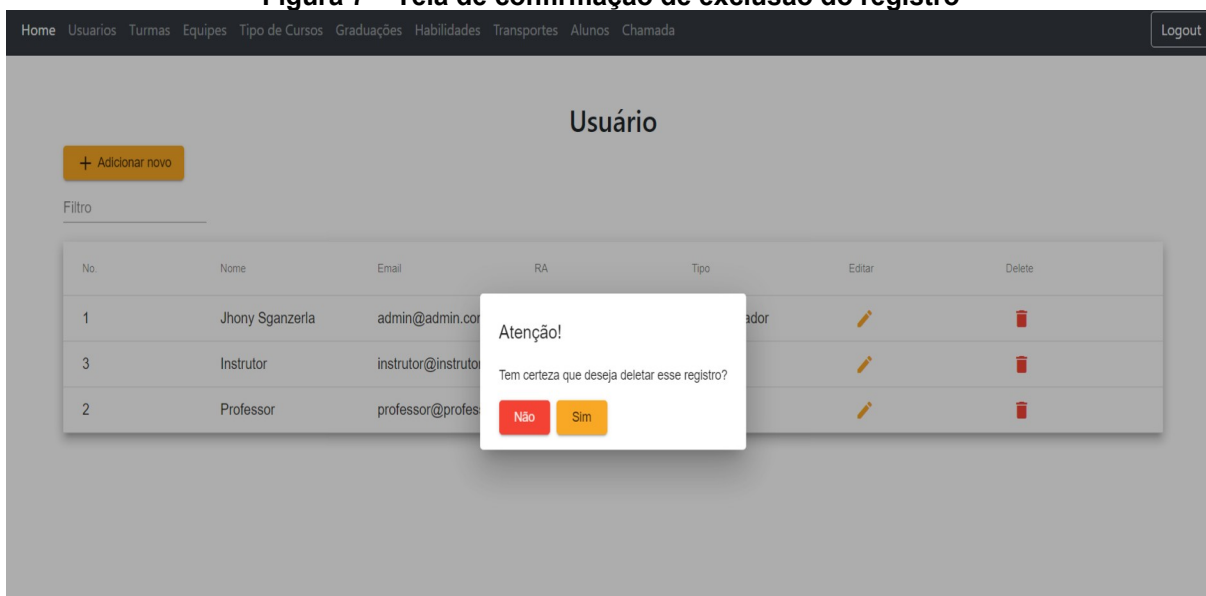
**Figura 6 – Tela de edição de usuário**



**Fonte: Autoria própria (2023).**

Ao selecionar a opção de exclusão, indicada pelo ícone de uma lixeira na tela de listagem de usuário, exibido na Figura 7, o sistema abrirá uma janela modal que solicitará ao usuário a confirmação da exclusão do cadastro, conforme ilustrado na Figura 10.

**Figura 7 – Tela de confirmação de exclusão do registro**



No.	Nome	Email	RA	Tipo	Editar	Delete
1	Jhony Sganzerla	admin@admin.com		Administrador		
3	Instrutor	instrutor@instrutor.com				
2	Professor	professor@professor.com				

**Fonte: Autoria própria (2023).**

Da mesma forma, todas as outras telas que envolvem cadastros possuem uma listagem que abrange as funcionalidades principais de um formulário. Essas funcionalidades incluem a capacidade de inserir novos registros, manter os

existentes e excluí-los, quando necessário. Essa abordagem permite que o usuário realize as operações básicas de um cadastro.

Algumas telas, como a tela de listagem de usuário (Figura 7), possuem comportamentos distintos com base nas permissões do usuário que está autenticado no sistema, como exemplificado na Figura 11.

**Figura 8 – Tela de listagem de usuários com perfil de instrutor**

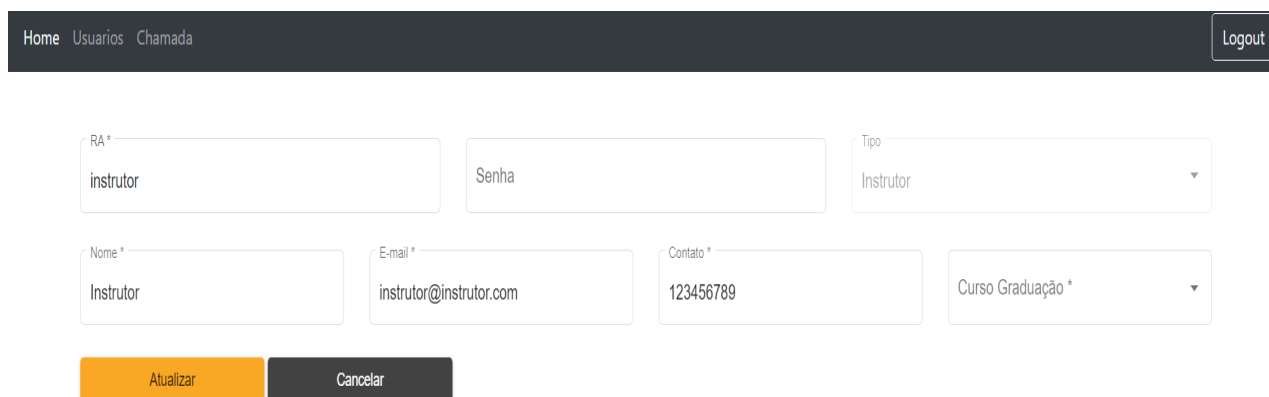


No.	Nome	Email	RA	Tipo	Editar	Delete
3	Instrutor	instrutor@instrutor.com	instrutor	instrutor		

**Fonte: Autoria própria (2023).**

Um usuário com perfil de instrutor tem permissões limitadas dentro do sistema. Ele só pode visualizar e realizar manutenções em seu próprio perfil, não tendo a capacidade de alterar seu tipo de permissão. Além disso, o instrutor não possui permissão para inserir novos registros no sistema, conforme ilustrado na Figura 12. Essas restrições foram implementadas para garantir que cada usuário tenha acesso e controle adequado de acordo com a sua função dentro do sistema.

**Figura 9 – Tela de alteração de usuário com perfil de instrutor**



RA \*  
instrutor

Senha

Tipo  
Instrutor

Nome \*  
Instrutor

E-mail \*  
instrutor@instrutor.com

Contato \*  
123456789

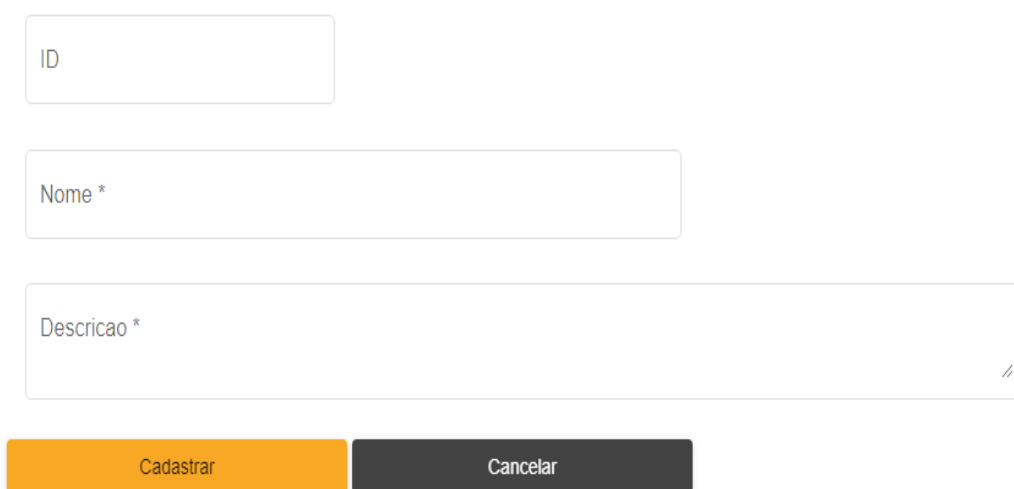
Curso Graduação \*

Atualizar Cancelar

**Fonte: Autoria própria (2023)**

As telas de cadastro simples do sistema são compostas por três campos básicos, sendo eles: *ID*, *Nome* e *Descrição*, como ilustrado na Figura 13. Alguns exemplos de telas que seguem esse padrão são os cadastros de tipos de cursos, graduações, habilidades e transportes. Nessas telas, o usuário pode inserir as informações necessárias nos campos correspondentes para cadastrar novos itens ou atualizar os existentes.

**Figura 10 - Modelo padrão para registros básicos**



O formulário apresenta três campos de entrada de texto empilhados verticalmente. O primeiro campo, rotulado 'ID', é o menor. O segundo campo, rotulado 'Nome \*', é de tamanho médio. O terceiro campo, rotulado 'Descricao \*', é o maior e possui uma barra de rolagem no canto inferior direito. Abaixo dos campos, há dois botões: 'Cadastrar' em um fundo laranja e 'Cancelar' em um fundo cinza escuro.

**Fonte: Autoria própria (2023).**

Algumas telas incluem listas que precisam ser preenchidas para associar cadastros. Um exemplo disso é a tela de cadastro de equipe, que possui um campo de listagem para selecionar os professores e/ou instrutores que farão parte da equipe, como pode ser visto na Figura 14.

**Figura 11 – Tela de cadastro de equipe**

ID  
1

Papel \*  
Equipe para aula de Java 20/06

Transporte

Novo	ID	Nome	Email	Remove
+	1	Jhony Sganzerla	admin@admin.com	
+		Usuario		

Atualizar Cancelar

**Fonte: Aatoria própria (2023).**

Da mesma forma, a tela de cadastro de alunos permite associar uma lista de turmas às quais o aluno pertencerá, conforme ilustrado na Figura 15.

**Figura 12 – Tela de cadastro de aluno**

ID  
1

Nome \*  
Jhony Sganzerla

Nome do Contato \*  
Jhony Sganzerla

Telefone do Contato \*  
(12) 34564-7897

Telefone

Email

Lista de Turmas Participadas

Nome	Deletar
Turma Turma de Java Sábado manhã	

Adicionar Curso

Atualizar Cancelar

**Fonte: Aatoria própria (2023).**

Além de gerenciar os cadastros de turmas, equipes, instrutores e alunos, o sistema inclui a funcionalidade de fazer chamadas para o controle de presença nas turmas. Essa ferramenta possibilita um acompanhamento eficiente da presença dos alunos durante as aulas. Na tela de chamada (Figura 16), são exibidas as turmas disponíveis para registro de presença. Caso o usuário que esteja realizando a chamada seja um instrutor, apenas as turmas em que ele está vinculado serão exibidas.

**Figura 13 – Tela de listagem de turma para chamadas**

Chamada

Filtro \_\_\_\_\_

Chamada	No.	Curso	Data Inicio	Dia da Semana	Local
	1	Turma de Java	Sábado	manhã	
	2	Turma de C#			

**Fonte: Autoria própria (2023).**

Ao clicar no botão de acesso à chamada, localizado ao lado do número da turma, conforme ilustrado na Figura 16, o usuário é redirecionado para a tela de chamada, como pode ser observada na Figura 17. Nessa tela, para cada aluno, é possível selecionar uma das seguintes opções: presente, ausente, feriado, cancelado. O total de presença é contabilizado na lista localizada abaixo da chamada. É importante ressaltar que os dias marcados como “feriado” não são incluídos no cálculo da porcentagem de frequência dos alunos.



Figura 14 – Tela de controle de chamada

Nome	01/05/2023	02/05/2023	03/05/2023	04/05/2023	05/05/2023	06/05/2023	07/05/2023	08/05/2023	09/05/2023	10/05/2023	11/05/2023	12/05/2023	Nova
Jhony Sganzerla	Presente	Presente	Presente	Presente	Presente	Presente	Presente	Feriado	Presente	Presente	Presente	Presente	
Jhony Sganzerla3	Presente	Presente	Presente	Ausente	Ausente	Presente	Presente	Feriado	Presente	Presente	Presente	Presente	
Jhony Sganzerla4	Presente	Ausente	Presente	Presente	Presente	Presente	Presente	Feriado	Presente	Presente	Presente	Presente	
Jhony Sganzerla5	Presente	Ausente	Ausente	Presente	Presente	Presente	Presente	Feriado	Presente	Ausente	Presente	Presente	

Nome	Presente	Ausente	Feriado	Cancelado	% de Presença
Jhony Sganzerla	11	0	1	0	100%
Jhony Sganzerla3	9	2	1	0	82%
Jhony Sganzerla4	10	1	1	0	91%
Jhony Sganzerla5	8	3	1	0	73%

Atualizar
Voltar

Fonte: Autoria própria (2023).

#### 4.4 Implementação do Sistema

O sistema foi desenvolvido utilizando a linguagem Java, e aproveitando a ampla gama de projetos oferecidos pelo *framework* Spring incluindo o Spring Boot, Spring Data e Spring Security em conjunto com a IDE IntelliJ para realização do *back-end*. O *framework* Angular para um desenvolvimento simples e rápido para a implementação do *front-end*, utilizando a ferramenta Visual Studio Code e para persistência de dados, o banco PostgreSQL.

O sistema pode ser acessado por meio da autenticação do usuário que foi implementada com o auxílio do Spring Security. No Código-fonte 1 é possível visualizar como foi realizada a implementação da autenticação no sistema e a definição de direitos de acesso às páginas.

Na Linha 17 percebe-se que a classe `WebSecurityConfig` utiliza a anotação “`@EnableWebSecurity`” para ativar o suporte à segurança do Spring Security e, ainda, fornecer a integração com o Spring MVC. Essa classe estende também a classe `WebSecurityConfigurerAdapter` que é responsável por definir alguns métodos que são responsáveis pela configuração da segurança na aplicação.

## Código-fonte 1 - WebSecurity

```

17. @EnableWebSecurity
18. @Configuration
19. public class WebSecurity {
20.
21.     private final AuthUserService authUserService;
22.     private final AuthenticationEntryPoint authenticationEntryPoint;
23.
24.     public WebSecurity(AuthUserService authUserService,
25.                       AuthenticationEntryPoint
authenticationEntryPoint) {
26.         this.authUserService = authUserService;
27.         this.authenticationEntryPoint = authenticationEntryPoint;
28.     }
29.     @Bean
30.     @SneakyThrows
31.     public SecurityFilterChain filterChain(HttpSecurity http) {
32.         AuthenticationManagerBuilder authenticationManagerBuilder =
33.
http.getSharedObject(AuthenticationManagerBuilder.class);
34.
authenticationManagerBuilder.userDetailsService(authUserService)
35.         .passwordEncoder(passwordEncoder());
36.         AuthenticationManager authenticationManager =
authenticationManagerBuilder.build();
37.
38.         http.headers().frameOptions().disable();
39.
40.         http.cors()
41.             .and().csrf().disable()
42.             .exceptionHandling()
43.                 .authenticationEntryPoint(authenticationEntryPoint)
44.             .and()
45.             .authorizeRequests()
46.                 .antMatchers("/error/**").permitAll()
47.                 .antMatchers("/auth/**").permitAll()
48.
49.                 .antMatchers(
"/users/**").hasAnyRole("INSTRUTOR", "PROFESSOR", "ADMIN")
50.                 .antMatchers(
"/alunoturmacurso/**").hasAnyRole("INSTRUTOR", "PROFESSOR", "ADMIN")
51.                 .antMatchers(
"/chamada/**").hasAnyRole("INSTRUTOR", "PROFESSOR", "ADMIN")
52.
53.                 .antMatchers(
"/cursosgraduacao/**").hasAnyRole("INSTRUTOR", "PROFESSOR", "ADMIN")
54.                 .antMatchers(
"/cursosgraduacao/**").hasAnyRole("PROFESSOR", "ADMIN")
55.
56.                 .antMatchers("/cursos/**").hasAnyRole("PROFESSOR",
"ADMIN")
57.                 .antMatchers("/equipe/**").hasAnyRole("PROFESSOR",
"ADMIN")
58.                 .antMatchers(
"/habilidades/**").hasAnyRole("PROFESSOR", "ADMIN")
59.                 .antMatchers("/turmas/**").hasAnyRole("PROFESSOR",
"ADMIN")
60.                 .antMatchers(
"/tipocursos/**").hasAnyRole("PROFESSOR", "ADMIN")

```

```

61.         .antMatchers (
62.             "/tipotransporte/**").hasAnyRole ("PROFESSOR", "ADMIN")
63.         .antMatchers ("/h2-console/**",
64.             "/swagger-resources/**",
65.             "/swagger-ui.html",
66.             "/swagger-ui/**",
67.             "/v2/api-docs",
68.             "/webjars/**").permitAll ()
69.         .anyRequest ().authenticated ()
70.         .and ()
71.         .authenticationManager (authenticationManager)
72.         //Filtro da Autenticação
73.         .addFilter (new
74.             JWTAuthenticationFilter (authenticationManager, authUserService) )
75.         //Filtro da Autorização
76.         .addFilter (new
77.             JWTAuthorizationFilter (authenticationManager, authUserService) )
78.         .sessionManagement ().sessionCreationPolicy (SessionCreationPolicy.STATE
79.             LESS) ;
80.         return http.build () ;
81.     }
82.     @Bean
83.     public PasswordEncoder passwordEncoder () {
84.         return new BCryptPasswordEncoder () ;
85.     }
86. }

```

Fonte: Autoria própria (2023).

No Código-fonte 1 nas Linha 53 e 54, é possível observar um trecho de código que define as validações globais de acesso com base nas permissões do usuário. Nesse trecho, é definido que todos os usuários com permissão podem realizar requisições do tipo GET para os cursos de graduação. No entanto, somente os usuários com perfil de professor e administrador têm permissão para realizar todos os tipos de requisições. Isso significa que todos os usuários autenticados podem ler as informações, mas apenas professores e administradores têm permissão para realizarem ações como inserir, atualizar ou excluir cursos. Essa configuração permite um controle granular de acesso, garantindo que apenas usuários autorizados possam executar determinadas operações que impactam diretamente os dados dos cursos de graduação. Dessa forma, a segurança e a integridade das informações são preservadas, enquanto a leitura das informações é aberta a todos os usuários autenticados.

Quando um usuário acessa o sistema, ele é redirecionado para a tela de autenticação, caso não esteja autenticado. Nessa tela, são solicitados os dados de usuário e senha, como pode ser visto no Código-fonte 2.

A estilização da tela de autenticação foi feita utilizando a biblioteca Bootstrap. Além disso, foram criadas algumas novas classes para personalizar o estilo, conforme observado nas linhas 1 e 2.

#### Código-fonte 2 - HTML da tela de Login

```

1. <div class="background-wrap">
2. <div class="background"></div>
3. </div>
4.
5. <form id="accesspanel" action="">
6. <h1 id="litheader">UTFPR - Cursos de Extensão</h1>
7. <div class="inset">
8. <p>
9. <input
10. type="text"
11. #ra="ngModel"
12. required
13. name="ra"
14. [(ngModel)]="usuarioLogin.ra"
15. id="ra"
16. placeholder="RA"
17. (change)="clearMessage()"
18. />
19. </p>
20. <p>
21. <input
22. type="password"
23. #senha="ngModel"
24. required
25. name="password"
26. [(ngModel)]="usuarioLogin.password"
27. id="password"
28. placeholder="Senha"
29. (change)="clearMessage()"
30. />
31. </p>
32. </div>
33.
34. <div class="text-center inset text-white">
35. <div *ngIf="(ra.dirty || ra.touched)" >
36. <div *ngIf="ra.errors?.['required']">Nome é obrigatório.</div>
37. </div>
38. <div *ngIf="(senha.dirty || senha.touched)" >
39. <div *ngIf="senha.errors?.['required']">Senha é obrigatório.</div>
40. </div>
41. <p *ngIf="!!this.message">
42. {{ this.message }}
43. </p>

```

```
44. </div>
45.
46. <p class="p-container">
47. <input
48. type="submit"
49. name="Login"
50. id="go"
51. value="Login"
52. (click)="doLogin()"
53. />
54. </p>
...

```

Fonte: Autoria própria (2023).

Para que o usuário possa se autenticar no sistema é necessário fornecer o RA solicitado na linha 14 e a senha na linha 26 do Código-fonte 2. Caso isso não ocorra, será exibida uma mensagem informando que os campos são de preenchimento obrigatório.

Em caso de ocorrência de erro, a variável contida na linha 42 do Código-fonte 2, será mostrada, contendo informações sobre o erro ocorrido. Por exemplo, se a senha fornecida não coincidir com a cadastrada, a mensagem “Usuário ou Senha inválido” será exibida.

Após isso, ao clicar no botão para realizar a autenticação, a função “doLogin” da linha 52 do Código-fonte 2 será acionada, realizando uma tentativa de autenticação. Essa função compara o RA do aluno e a senha codificada com uma busca no banco de dados, utilizando a classe “Usuario”, presente no Código-fonte 3.

### Código-fonte 3 - Classe de Usuário

```
1. package com.sganzerla.model;
2.
3. import lombok.Data;
4. import lombok.ToString;
5. import org.springframework.security.core.GrantedAuthority;
6. import org.springframework.security.core.userdetails.UserDetails;
7.
8. import javax.persistence.*;
9. import javax.validation.constraints.*;
10.     import java.util.ArrayList;
11.     import java.util.Collection;
12.     import java.util.Set;
13.
14.     @Entity
15.     @Data
16.     @ToString
17.     public class Usuario implements UserDetails {
18.         @Id

```

```

19.     @GeneratedValue(strategy=GenerationType.IDENTITY)
20.     private Long id;
21.
22.     @NotNull
23.     @NotEmpty(message = "O email de usuário não pode ser vazio")
24.     @NotBlank
25.     @Email(message = "Email inválido",
26.     regexp = "^(?=.{1,64}@)[A-Za-z0-9_-]+(\\.[A-Za-z0-9_-]+)*@[^-]
[A-Za-z0-9_-]+(\\.[A-Za-z0-9_-]+)* (\\. [A-Za-z]{2,})$" )
27.     private String email;
28.     @NotNull(message = "A senha não pode ser nula")
29.     private String password;
30.
31.     @NotNull(message = "O nome de usuário não pode ser nulo")
32.     @NotEmpty(message = "O nome de usuário não pode ser vazio")
33.     @NotBlank
34.     @Size(min = 4, max = 255, message = "O tamanho deve ser entre
{min} e {max}")
35.     private String nome;
36.
37.     @NotNull(message = "O tipo não pode ser nulo")
38.     @NotEmpty(message = "O tipo não pode ser vazio")
39.     @NotBlank
40.     @Size(max = 25)
41.     private String tipo; //administrador, professor ou instrutor
42.
43.     @Column(length = 20)
44.     @NotNull(message = "O telefone não pode ser nulo")
45.     @NotEmpty(message = "O telefone não pode ser vazio")
46.     @NotBlank
47.     @Size(max = 20, message = "O telefone deve ser entre 0 e
{max}")
48.     private String telefone;
49.
50.     @Column(length = 20)
51.     @NotNull(message = "O RA não pode ser nulo")
52.     @NotEmpty(message = "O RA de usuário não pode ser vazio")
53.     @NotBlank
54.     @Size(max = 20, message = "O RA deve ser entre 0 e {max}")
55.     private String ra;
56.
57.     @ManyToOne
58.     @JoinColumn(name = "cursosgraduacao_id" , referencedColumnName
= "id")
59.     private CursosGraduacao cursoGraduacao;
60.
61.     @ManyToMany(fetch = FetchType.EAGER)
62.     @JoinTable(name = "users_abilities",
63.     joinColumns = @JoinColumn(
64.     name = "usuario_id", referencedColumnName = "id"),
65.     inverseJoinColumns = @JoinColumn(
66.     name = "authority_id", referencedColumnName = "id"))
67.     private Set<Authority> userAuthorities;
68.
69.     @Override
70.     public Collection<? extends GrantedAuthority> getAuthorities()
{
71.     return new ArrayList<GrantedAuthority>(this.userAuthorities);
72.     }
73.
74.     @Override

```

```

75.     public String getUsername () {
76.         return this.getRa ();
77.     }
78.
79.
80.     @Override
81.     public boolean isAccountNonExpired () {
82.         return true;
83.     }
84.
85.     @Override
86.     public boolean isAccountNonLocked () {
87.         return true;
88.     }
89.
90.     @Override
91.     public boolean isCredentialsNonExpired () {
92.         return true;
93.     }
94.
95.     @Override
96.     public boolean isEnabled () {
97.         return true;
98.     }
99. }
100.

```

Fonte: Autoria própria (2023).

No Código-fonte 3, é possível ver o uso da biblioteca Lombok nas anotações das linhas 15 e 16, simplificando a criação de métodos *getters*, *setters* e construtores. Além disso, validações de valores nulos e em branco são aplicadas, como evidenciado nas anotações “@NotNull”, “@NotBlank” e “@NotEmpty”. Também é utilizada uma coluna relacional para associar o usuário às suas permissões.

Para realizar a consulta do usuário autenticado, é necessário acessar o banco de dados, cujas configurações estão registradas no arquivo “application.properties”. Dessa forma, o Java consegue estabelecer a conexão com o banco de dados Postgres, conforme demonstrado no Código-fonte 4. Na linha 4 é especificado o nome da configuração, enquanto que nas linhas 9,10 e 11 são definidas a *Uniform Resource Locator* (URL) de conexão, com o nome do usuário e senha do banco de dados, respectivamente.

#### Código-fonte 4 - Propriedades da aplicação

```

1. #Application properties
2. spring.mvc.pathmatch.matching-strategy=ant_path_matcher
3. spring.config.name=CursosExtensao

```

```

4. server.port=8080
5.
6. # Datasource
7. spring.datasource.driver-class-name=org.postgresql.Driver
8. spring.datasource.url=jdbc:postgresql://localhost:5432/cursosextencao
9. spring.datasource.username=postgres
10. spring.datasource.password=123
11.
12. #JPA
13. spring.jpa.show-sql=true
14. spring.jpa.database=POSTGRESQL
15. spring.jpa.hibernate.ddl-auto=create-drop
16. spring.jpa.properties.hibernate.dialect =
    org.hibernate.dialect.PostgreSQL9Dialect
17. spring.jpa.properties.hibernate.globally_quoted_identifiers=true

```

**Fonte: Autoria própria (2023).**

Para possibilitar a adição de novos usuários foram desenvolvidos formulários de inserção e listas. Essas listas seguem um padrão consistente em todas as telas existentes, o qual pode ser verificado no HTML do Código-fonte 5, e no arquivo Typescript do Código-fonte 6. Esses formulários e listas facilitam a adição de novos usuários e a visualização dos dados. O código HTML define a estrutura e o leiaute dos formulários, enquanto o arquivo TypeScript contém a lógica necessária para manipular os dados, como adicionar novos usuários e exibir a lista atualizada.

#### Código-fonte 5 - HTML da lista de usuários

```

1. <h2 class="text-center">Usuário</h2>
2.
3. <button mat-raised-button color="primary" routerLink="/usuario/novo"
   *ngIf="allowNew">
4.   <mat-icon>add</mat-icon>
5.   Adicionar novo
6. </button>
7.
8. <div>
9.   <mat-form-field>
10.     <mat-label>Filtro</mat-label>
11.     <input
12.       matInput
13.       (blur)="applyFilter(input.value)"
14.       placeholder="Ex. Jhony Sganzerla"
15.       #input
16.     />
17.   </mat-form-field>
18. </div>
19.
20.   <div class="overflow-auto height-mat-table mat-elevation-z8">
21.     <table
22.       mat-table
23.       #table
24.       [dataSource]="dataSource"
25.       class="w-100"

```



```

26.     >
27.     <ng-container matColumnDef="id">
28.     <th mat-header-cell *matHeaderCellDef>No.</th>
29.     <td mat-cell *matCellDef="let element">{{ element.id }}</td>
30.     </ng-container>
31.
32.     <ng-container matColumnDef="nome">
33.     <th mat-header-cell *matHeaderCellDef>Nome</th>
34.     <td mat-cell *matCellDef="let element">{{ element.nome }}</td>
35.     </ng-container>
36.
37.     <ng-container matColumnDef="email">
38.     <th mat-header-cell *matHeaderCellDef>Email</th>
39.     <td
40.     mat-cell
41.     *matCellDef="let
42.     element">{{ element.email }}</td>
43.     </ng-container>
44.
45.     <ng-container matColumnDef="ra">
46.     <th mat-header-cell *matHeaderCellDef>RA</th>
47.     <td mat-cell *matCellDef="let element">{{ element.ra }}</td>
48.     </ng-container>
49.
50.     <ng-container matColumnDef="tipo">
51.     <th mat-header-cell *matHeaderCellDef>Tipo</th>
52.     <td mat-cell *matCellDef="let element">{{ element.tipo }}</td>
53.     </ng-container>
54.
55.     <ng-container matColumnDef="editar">
56.     <th mat-header-cell *matHeaderCellDef>Editar</th>
57.     <td mat-cell *matCellDef="let element">
58.     <button
59.     mat-icon-button
60.     matTooltip="Clique para editar"
61.     class="iconbutton"
62.     color="primary"
63.     (click)="edit(element.id) "
64.     >
65.     <mat-icon>edit</mat-icon>
66.     </button>
67.     </td>
68.     </ng-container>
69.
70.     <ng-container matColumnDef="deletar">
71.     <th
72.     mat-header-cell
73.     *matHeaderCellDef
74.     mat-sort-
75.     header>Delete</th>
76.     <td mat-cell *matCellDef="let element">
77.     <button
78.     mat-icon-button
79.     matTooltip="Clique para Deletar"
80.     class="iconbutton"
81.     (click)="apagar(element.id) "
82.     color="warn"
83.     >
84.     <mat-icon>delete</mat-icon>
85.     </button>
86.     </td>
87.     </ng-container>
88.
89.     <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
90.     <tr
91.     mat-row
92.     *matRowDef="let
93.     row;
94.     columns:
95.     displayedColumns"></tr>

```

```

84.
85.     <tr class="mat-row" *matNoDataRow>
86.     <td class="mat-cell" colspan="4">
87.         Sem dados no filtro "{{ input.value }}"
88.     </td>
89.     </tr>
90. </table>
91.</div>
92.

```

Fonte: Autoria própria (2023).

Também é possível ver no Código-fonte 5, nas linhas 4 a 7, o botão que redireciona ao “/novo”, que é responsável pelo cadastro de novos usuários. O *datasource* carregado na linha 25, faz com que a tabela receba os dados a serem exibidos em cada coluna e linha. Esse comportamento pode ser visualizado nas linhas 43 a 45 onde é mostrado o RA do aluno é cadastrado e exibido.

No Código-fonte 6, é exibida a validação de permissão de usuário na linha 24. Após isso, é importante observar que a “displayedColumns”, que é uma variável de lista de texto que recebe os títulos do cabeçalho é criada, recebendo o nome dos campos que serão exibidos no cabeçalho da lista. Essa definição permite ler o *datasource* para que a tabela criada na linha 21 do Código-fonte 5 separe cada cabeçalho de seus valores e os exiba na tela.

#### Código-fonte 6 - Typescript da lista de usuários

```

1. import { Component, OnInit } from '@angular/core';
2. import { MatDialog } from '@angular/material/dialog';
3. import { Router } from '@angular/router';
4. import { ConfirmDialogComponent } from
   './../shared/confirmDialog.component';
5. import { UsuarioService } from '../usuario.service';
6. import { AuthService } from 'src/app/auth/auth.service';
7.
8. @Component({
9.   selector: 'app-usuario',
10.  templateUrl: './usuario.component.html',
11.  styleUrls: ['./usuario.component.css']
12. })
13. export class UsuarioComponent implements OnInit {
14.
15.   dataSource:any;
16.   filtro: string = '';
17.   allowNew: boolean = true;
18.
19. constructor(private usuarioService: UsuarioService, private
   router:Router, public dialog: MatDialog, private authService:
   AuthService) { }
20.
21.   ngOnInit(): void {
22.     this.loadTable();

```

```

23.
24.     this.allowNew = !
    this.authService.temPermissao('ROLE_INSTRUTOR');
25.
26.     }
27.
28.     displayedColumns: string[] = ['id', 'nome', 'email', 'ra',
    'tipo', 'editar', 'deletar'];
29.
30.
31.     edit(id: number) {
32.         this.router.navigate(['usuario/alterar/' + id]);
33.     }
34.
35.     apagar(id: number) {
36.         const dialogRef = this.dialog.open(ConfirmDialogComponent, {
37.             data: 'Tem certeza que deseja deletar esse registro?'
38.         });
39.         dialogRef.afterClosed().subscribe(result => {
40.             if (result) {
41.                 this.usuarioService.delete(id).subscribe(() => {
42.                     this.loadTable();
43.                 });
44.             }
45.         });
46.     }
47.
48.     applyFilter(filterValue: string) {
49.         this.filtro = filterValue;
50.         this.loadTable();
51.     }
52.
53.     loadTable() {
54.         this.usuarioService.getUsuarios().subscribe((users: any) => {
55.             this.dataSource = users.filter((item: any) => {
56.                 return this.displayedColumns.some((key) => {
57.                     item[key] = item[key] == null ? '' : item[key];
58.                 return item[key]
59.                     .toString()
60.                     .toLowerCase()
61.                     .includes(this.filtro.toLowerCase());
62.                 });
63.             });
64.         });
65.     }
66. }

```

Fonte: Autoria própria (2023).

No Código-fonte 6, na linha 54, é possível visualizar a função padrão presente em todas as listas de formulários, responsável por realizar o filtro da tabela. Quando um valor é inserido é feita uma nova requisição ao servidor para buscar todos os usuários, aplicando um filtro para buscar apenas valores que contenham o texto digitado em algum dos campos.

Ao clicar no botão “Novo” das telas de cadastro de registros básicos (Figura 13), é seguido um código padrão utilizado por todas as telas com o mesmo estilo. Nesse código padrão são criados os campos de *id* (linhas 4 a 6), nome (linhas 10 a 16) e descrição (linhas 19 a 22) do Código-fonte 7. Cada um dos campos é controlado por um formulário que permite a validação de campos obrigatórios. Um exemplo dessa validação pode ser observado na linha 13, em que o código solicita a verificação da validade do campo “nome” do formulário. Caso o campo não esteja preenchido corretamente, a mensagem “Campo Obrigatório” será exibida em vermelha indicando que um erro ocorreu.

### Código-fonte 7 - HTML do cadastro de Habilidades

```

1. <form [formGroup]="form" (ngSubmit)="onSubmit()" class="col-12">
2. <div >
3. <mat-form-field appearance="outline" class="col-2">
4. <mat-label>ID</mat-label>
5. <input matInput type="number" readonly FormControlName="id" />
6. </mat-form-field>
7.
8. </div>
9.
10. <mat-form-field appearance="outline" class="col-4">
11. <mat-label>Nome</mat-label>
12. <input matInput type="nome" FormControlName="nome" />
13. <mat-error *ngIf="form.get('nome')!['invalid']"
14. >Campo obrigatório</mat-error
15. >
16. </mat-form-field>
17.
18. <br />
19. <mat-form-field appearance="outline" class="col-6">
20. <mat-label>Descricao</mat-label>
21. <textarea matInput type="descricao"
22. FormControlName="descricao"></textarea>
23. </mat-form-field>
24.
25. <div class="actions col-12 row">
26. <button mat-raised-button color="primary" type="submit"
27. class="col-2 mr-1">
28.   {{ isNew ? "Cadastrar" : "Atualizar" }}
29. </button>
30. <button
31.   class="col-2"
32.   mat-raised-button
33.   color="accent"
34.   type="button"
35.   (click)="onCancel()"
36. >
37.   Cancelar
38. </button>
39. </div>
40. </form>

```

Fonte: Autoria própria (2023).

Se o usuário estiver realizando uma operação de inserção ou edição de um registro, o texto “Salvar” do botão que permite adicionar o registro no banco de dados, será alterado “Cadastrar” ou “Atualizar”, como pode ser observado na linha 26 do Código-fonte 7. Essa mudança é controlada por meio da variável “isNew” que é criada na linha 16 do Código-fonte 8 e atualizada ao entrar na tela (linha 36). Nessa etapa, é verificado se o campo “ID” existe. Se o campo existe significa que o registro está sendo alterado e não criado. Essa informação é utilizada para controlar o registro e enviar os dados para o servidor.

#### Código-fonte 8 - Typescript do cadastro de habilidades

```
1. import { Component, OnInit } from '@angular/core';
2. import { FormBuilder, FormControl, FormGroup, Validators } from
  '@angular/forms';
3. import { ActivatedRoute, Router } from '@angular/router';
4. import { HabilidadesService } from '../habilidades.service';
5. import { Habilidades } from '../model/habilidades';
6.
7. @Component({
8. selector: 'app-habilidades-crud',
9. templateUrl: './habilidades-crud.component.html',
10. styleUrls: ['./habilidades-crud.component.css'],
11. })
12. export class HabilidadesCrudComponent implements OnInit {
13.
14. form: FormGroup
15.
16. isNew: boolean = true;
17. habilidadesId: number;
18. habilidadesOptions: Array<Habilidades>;
19.
20. constructor(
21. private formBuilder: FormBuilder,
22. private habilidadesService: HabilidadesService,
23. private route: ActivatedRoute,
24. private router: Router
25. ) {}
26.
27. ngOnInit() {
28.
29. this.form = this.formBuilder.group({
30. id: [''],
31. nome: ['', Validators.required],
32. descricao: ['', Validators.required],
33. });
34.
35. this.route.params.subscribe((params) => {
36. if (params['id']) {
37. this.isNew = false;
38. this.habilidadesId = +params['id'];
39. thi
```

```

s.habilidadesService.getHabilidade(this.habilidadesId).subscribe((ha
bilidades) => {
40.   this.form.patchValue({
41.     id: habilidades.id,
42.     nome: habilidades.nome,
43.     descricao: habilidades.descricao,
44.   });
45. });
46. }
47. });
48. }
49.
50.   onSubmit() {
51.     const habilidades: Habilidades = this.form.value;
52.     if (!this.form.valid) return;
53.     this.habilidadesService.save(habilidades).subscribe(() => {
54.       this.router.navigateByUrl('/habilidades');
55.     });
56.   }
57.
58.   onCancel() {
59.     this.router.navigateByUrl('/habilidades');
60.   }
61. }

```

Fonte: Autoria própria (2023).

Além disso, o Código-fonte 8 demonstra que ao acessar uma tela de cadastro que contenha registros básicos (que são registros compostos por id, nome e descrição), se houver um *id* para o registro, esse valor é carregado do banco de dados para os campos receberem atualizações (linha 39). Essa funcionalidade permite que o usuário visualize e atualize os dados existentes. Ao enviar o registro para o servidor (linha 53), é realizada uma validação dos dados preenchidos no formulário para garantir que todos os campos que são de preenchimento obrigatórios contenham os valores esperados (linha 52).

Em algumas telas com funcionalidades mais complexas, como a de controle de chamada (figuras 16 e 17), são encontrados códigos mais complexos que são os que contém listas, mapas, cálculos e não somente objetos simples para persistência, os quais usam alguns recursos do Angular Material, como pode ser visto no Código-fonte 9. Nesse código são definidos os elementos e estrutura da interface relacionados às operações de criação, leitura, atualização e exclusão de chamadas.

#### Código-fonte 9 - HTML da tela de chamada

```

1. <form (ngSubmit)="onSubmit()" class="col-12">
2. <div *ngIf="isTurmaVazia; else turma">
3. <h2>Nenhum aluno encontrado para esta turma</h2>
4.

```

```

5. <button
6.   class="col-2"
7.   mat-raised-button
8.   color="accent"
9.   type="button"
10.    (click)="onCancel()"
11.  >
12.    Voltar
13.  </button>
14. </div>
15.
16.   <ng-template #turma>
17.     <table
18.       class="table table-striped table-bordered table-hover table-sm
19.         text-center table-responsive"
20.     >
21.       <thead>
22.         <tr>
23.           <th scope="col" style="width: fit-content">Nome</th>
24.           <th scope="col" *ngFor="let data of listaDeDatas; let i =
25.             index">
26.             <mat-form-field style="width: 100px">
27.               <b>
28.                 <input
29.                   matInput
30.                   type="date"
31.                   [ngModelOptions]="{ standalone: true }"
32.                   [ngModel]="listaDeDatas[i] | date : 'yyyy-MM-dd'"
33.                   (ngModelChange)="changeData($event, i)"
34.                 />
35.               </b>
36.             </mat-form-field>
37.           </th>
38.           <th>
39.             <button
40.               mat-raised-button
41.               color="primary"
42.               type="button"
43.               class="col-2 mr-1"
44.               (click)="adicionarData()"
45.               {{ "Nova" }}
46.             </button>
47.           </th>
48.         </tr>
49.       </thead>
50.       <tbody>
51.         <tr
52.           *ngFor="
53.             let aluno of pegarNomesAlunoFromMap();
54.             let i = index;
55.             trackBy: trackByFn
56.           ">
57.           <td>
58.             <b>{{ aluno }}</b>
59.           </td>
60.           <td
61.             *ngFor="
62.               let data of pegaChamadasFromMap(aluno);
63.               let i = index;

```

```

64.     trackBy: trackByFn
65.     "
66.     [class]="pegaChamadasFromMap(aluno)![i].presenca"
67.     >
68.     <mat-select
69.     [(ngModel)]="pegaChamadasFromMap(aluno)![i].presenca"
70.     [(ngModelOptions)]="{ standalone: true }"
71.     >
72.     <mat-option value="Presente">Presente</mat-option>
73.     <mat-option value="Ausente">Ausente</mat-option>
74.     <mat-option value="Feriado">Feriado</mat-option>
75.     <mat-option value="Cancelado">Cancelado</mat-option>
76.     </mat-select>
77.     </td>
78.     </tr>
79.     </tbody>
80.     </table>
81.     <br />
82.
83.     <table
84.     class="table table-striped table-bordered table-hover table-sm
85.     text-center table-responsive"
86.     style="width: fit-content"
87.     >
88.     <thead>
89.     <tr>
90.     <th style="width: fit-content">Nome</th>
91.     <th>
92.     <b>
93.     {{ "Presente" }}
94.     </b>
95.     </th>
96.     <th>
97.     <b>
98.     {{ "Ausente" }}
99.     </b>
100.    </th>
101.    <th>
102.    <b>
103.    {{ "Feriado" }}
104.    </b>
105.    </th>
106.    <th>
107.    <b>
108.    {{ "Cancelado" }}
109.    </b>
110.    </th>
111.    <th>
112.    <b>
113.    {{ "% de Presença" }}
114.    </b>
115.    </th>
116.    </tr>
117.    </thead>
118.
119.    <tbody>
120.    <tr
121.    *ngFor="
122.    let aluno of pegarNomesAlunoFromMap();
123.    let i = index;
124.    trackBy: trackByFn

```



```
124.     "  
125.     >  
126.     <td>  
127.     <b>{{ aluno }}</b>  
128.     </td>  
129.     <td>  
130.     *ngFor="  
131.     let total of totaisPresencaPorAluno(aluno) ;  
132.     let i = index;  
133.     trackBy: trackByFn  
134.     "  
135.     >  
136.     <b>{{ total }}</b>  
137.     </td>  
138.     </tr>  
139.     </tbody>  
140.     </table>  
141.     <br />  
142.     [...]
```

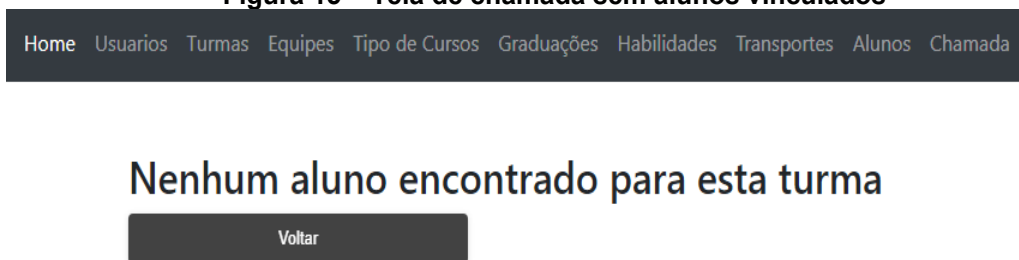
Fonte: Autoria própria (2023).

No Código-fonte 9, a na linha 2, é utilizada a diretiva *NgIf* para verificar se há alunos vinculados em uma turma. Caso não haja alunos matriculados será exibida a tela apresentada na Figura 18. O código da linha 3 representa a mensagem que será exibida quando não houver alunos encontrados para a turma, conforme definido na linha 2. Essa mensagem informa ao usuário que nenhum aluno foi encontrado para a turma selecionada.

Ainda o Código-fonte 9 faz com que, caso existam alunos, seja exibida a listagem dos alunos sendo possível registrar se o aluno está presente ou ausente, se está com a matriculada cancelada ou se em um determinado dia é feriado (linhas 68 a 76). Além disso, é disponibilizada a opção de escolher as datas em que ocorrem as aulas (linhas 26 a 34). Após selecionar uma data uma nova coluna na tabela de registro de presença é criada com a data atual para que seja registrada a presença dos alunos naquela data. É possível alterar a data, caso seja necessário.

Nas linhas 84 a 140, é criada a tabela referente ao total de presenças, na qual é exibida a porcentagem de presenças do aluno excluindo feriados.

**Figura 15 – Tela de chamada sem alunos vinculados**



**Fonte: A autoria própria (2023).**

Para que o leiaute da Figura 18 possa existir, no arquivo `chamada-crud.component.ts`, ocorrem diversas funções de agrupamento e controle de chamada (Código-fonte10). Alguns objetos foram criados para agrupar os registros, os quais podem ser visualizados nas linhas 15,16 e 22. Na linha 15 é criado o objeto “listaDeDatas”, que armazena o total de colunas das chamadas agrupadas por data. Na linha 16, o objeto “listaDeChamada” é utilizado para salvar as chamadas agrupados por aluno. Por fim, na linha 22 está o objeto “listaDeChamadasMap”, responsável por agrupar as chamadas por aluno e por data, a fim de possibilitar a exibição dos dados e posterior envio de todas as chamadas para o servidor.

A partir da linha 218, inicia-se a coleta de dados para criar a tabela de totais de presença por aluno. Nesse processo ocorre a soma de todos os valores presentes na chamada do aluno incluindo presenças, ausências, cancelados e feriados.

Em determinados pontos do código, como nas linhas 129 e 130, é realizado um tratamento para o horário. No banco de dados somente o dia é armazenado, contudo ao converter em data no *front-end*, é utilizado o horário padrão GMT-3, porém, presume-se que a data venha em GMT+0 e para que não sejam descontados 3 horas do dia ao carregar o valor, há uma conversão de compensação que adiciona 3 horas ao valor normal.

**Código-fonte 10 - Typescript do cadastro de chamada**

```

14. export class ChamadaCrudComponent implements OnInit {
15. listaDeDatas: Array<Date>;
16. listaDeChamada: Array<Chamada> = [];
17.
18. hojeFormatado = new Date().toLocaleDateString()

```

```

19.
20. isTurmaVazia = false;
21.
22. listaDeChamadasMap: Map<String, Array<Chamada>> = new Map<
23. string,
24. Array<Chamada>
25. >();
26.
27. formatarData = (data: Date) =>{
28. return ` ${data.getDate()}/${data.getMonth()+1}/${
    {data.getFullYear()} `
29. }
30.
31. idTurma: any;
32.
    [...]
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89. onSubmit() {
90.
91. this.listaDeChamada = [];
92.
93. for(let i = 0; i < this.pegarNomesAlunoFromMap().length; i++){
94. const aluno = this.pegarNomesAlunoFromMap()[i];
95. const chamadasDoAluno = this.listaDeChamadasMap.get(aluno);
96.
97. if(chamadasDoAluno){
98. this.listaDeChamada = this.listaDeChamada.concat(chamadasDoAluno);
99. }
100.}
101.
102.
103.this.listaDeChamada.forEach((chamada) => {
104.if(chamada.data.toString().length > 10){
105.
106.let data = chamada.data.toLocaleDateString().split('/');
107.let dataFormatada = ` ${data[2]}-${data[1]}-${data[0]} `
108.chamada.data = dataFormatada;
109.}
110.});
111.
112.
113.//verifica se nao há chamadas repetidas
114.for(let i = 0; i < this.pegarNomesAlunoFromMap().length; i++){
115.const aluno = this.pegarNomesAlunoFromMap()[i];
116.const chamadasDoAluno = this.listaDeChamadasMap.get(aluno);
117.
118.
119.if(chamadasDoAluno){
120.for(let j = 0; j < chamadasDoAluno.length; j++){
121.const chamada = chamadasDoAluno[j];
122.let cont = 0;
123.for(let k = 0; k < chamadasDoAluno.length; k++){
124.const chamada2 = chamadasDoAluno[k];
125.
126.let data1Convertida = new Date(chamada.data);
127.let data2Convertida = new Date(chamada2.data);

```

```

128.
129.data1Convertida.setTime (          data1Convertida.getTime ()          +
    data1Convertida.getTimezoneOffset ()*60*1000);
130.data2Convertida.setTime (          data2Convertida.getTime ()          +
    data2Convertida.getTimezoneOffset ()*60*1000);
131.
132.
133.if(this.formatarData(data1Convertida)                               ==
    this.formatarData(data2Convertida)){
134.cont++;
135.}
136.}
137.
138.if(cont > 1){
139.let d = new Date(chamada.data);
140.d.setTime( d.getTime () + d.getTimezoneOffset ()*60*1000 );
141.this.alertService.errorList (['Há mais de uma chamada com a data $
    {this.formatarData(d)}']) // para o aluno ${aluno}
142.return;
143.}
144.}
145.}
146.
147.}
148.
149.
150.this.chamadaService.save(this.listaDeChamada).subscribe(() => {
151.this.router.navigateByUrl('/chamada');
152.});
153.}
154.
155.onCancel () {
156.this.router.navigateByUrl('/chamada');
157.}
158.
159.trackByFn(index: any, item: any) {
160.return index;
161.}
162.
163.adicionarData () {
164.let dataAtual = new Date ();
165.this.listaDeDatas.push (dataAtual);
166.
167.for (let i = 0; i < this.pegarNomesAlunoFromMap ().length; i++) {
168.const aluno = this.pegarNomesAlunoFromMap () [i];
169.
170.let chamadasDoAluno = this.listaDeChamadasMap.get (aluno);
171.
172.if (chamadasDoAluno) {
173.let chamadaNova: Chamada = JSON.parse (
174.JSON.stringify (chamadasDoAluno [chamadasDoAluno.length - 1])
175.) ;
176.chamadaNova.id = undefined;
177.chamadaNova.data = dataAtual;
178.chamadaNova.conteudoministrado = '';
179.chamadaNova.presenca = 'Presente';

```

```
180.chamadasDoAluno.push(chamadaNova);
181.}
182.}
183.}
184.
185.pegarNomesAlunoFromMap() {
186.return Array.from(this.listaDeChamadasMap.keys());
187.}
188.
189.pegarChamadasFromMap(aluno: String) {
190.return this.listaDeChamadasMap.get(aluno);
191.}
192.
193.changeData(event: any, index: number) {
194.
195.this.alertService.clear();
196.
197.let a = new Chamada();
198.a.data = event;
199.
200.this.listaDeChamada[index] = a;
201.
202.for (let i = 0; i < this.pegarNomesAlunoFromMap().length; i++) {
203.const aluno = this.pegarNomesAlunoFromMap()[i];
204.
205.let chamadasDoAluno = this.listaDeChamadasMap.get(aluno);
206.
207.if (chamadasDoAluno) {
208.let chamadaNova: Chamada = JSON.parse(
209.JSON.stringify(chamadasDoAluno[index])
210.) ;
211.chamadaNova.data = this.listaDeChamada[index].data;
212.chamadasDoAluno[index] = chamadaNova
213.}
214.}
215.}
216.
217.
218.totaisPresencaPorAluno(aluno: String) {
219.let relatorio: Map<String, any> = new Map<String, any>();
220.let totais = [0,0,0,0,""];
221.
222.let chamadasDoAluno = this.listaDeChamadasMap.get(aluno);
223.
224.if (chamadasDoAluno) {
225.let total: Map<String, number> = new Map<String, number>();
226.total.set('Presente', 1);
227.total.set('Ausente', 1);
228.total.set('Feriado', 1);
229.total.set('Cancelado', 1);
230.
231.chamadasDoAluno.forEach((chamada) => {
232.let cont = total.get(chamada.presenca);
233.if (!!cont) {
234.total.set(chamada.presenca, cont + 1);
235.}
236.});
```

```
237.  
238.  
239.relatorio.set(aluno, { total: total });  
240.}  
241.  
242.totais[0] = relatorio.get(aluno)?.total.get('Presente')-1;  
243.totais[1] = relatorio.get(aluno)?.total.get('Ausente')-1;  
244.totais[2] = relatorio.get(aluno)?.total.get('Feriado')-1;  
245.totais[3] = relatorio.get(aluno)?.total.get('Cancelado')-1;  
246.  
247.totais[4] = Math.round(totais[0] * 100 / (this.listaDeDatas.length  
- totais[2])) + '%';  
248.  
249.return totais;  
250.}  
251.}
```

**Fonte: Aatoria própria (2023).**

## 5 CONCLUSÃO

O objetivo principal deste trabalho de conclusão de curso foi desenvolver um sistema web para gerenciamento dos cursos de extensão da UTFPR, Campus Pato Branco. Para isso, foram identificados os requisitos necessários para desenvolver as funcionalidades essenciais do sistema.

Atualmente a UTFPR não possui um sistema que permita controlar e gerenciar as atividades de extensão na forma de cursos de aperfeiçoamento para a comunidade. Essas atividades envolvem o controle de equipes de instrutores, turmas, horários e chamadas. Normalmente a gestão dos dados dos cursos depende principalmente do uso de planilhas, o que pode ser ineficiente e complicado.

Para solucionar essas questões, nesse trabalho foi desenvolvido um sistema com o objetivo de aprimorar a administração dos cursos. Esse sistema permite o cadastramento de alunos, professores, instrutores, habilidades, transporte e possibilita o controle de chamada dos alunos. Além disso, oferece recursos para o controle de turmas e equipes, tanto para os professores quanto para os administradores do sistema.

Durante o desenvolvimento desse projeto foram empregadas diversas ferramentas. Para o desenvolvimento do *front-end* e do *back-end*, foram utilizadas as IDEs IntelliJ e VSCode, que oferecem bom suporte, facilitando a criação e a manutenção do código, pois oferecem recursos que automatizam as tarefas comuns. Os *frameworks* Bootstrap e Angular Material, foram utilizados para a estilização das interfaces com recursos que permitem uma estilização simplificada e intuitiva. Além disso, ao utilizar o Angular, foi possível realizar telas reutilizáveis de forma mais eficiente e rápida. Isso facilitou a implementação de componentes que podem ser facilmente reaproveitados em diferentes páginas, proporcionando a reutilização de código. Com essa abordagem, fragmentos de diversas páginas podem ser combinados para formar um leiaute padronizado.

Outra ferramenta essencial para o desenvolvimento do trabalho foi o Spring pois oferece uma variedade de recursos para auxiliar na implementação do sistema. O Spring permite a injeção de dependências, simplificando a configuração e o gerenciamento. Além disso, ele oferece recursos avançados de segurança, possibilitando restringir o acesso de acordo com as permissões do usuário.

Durante o desenvolvimento do trabalho, algumas dificuldades foram encontradas, principalmente em relação à persistência e envio objetos para o *back-end*, devido à presença de telas complexas que s objetos não totalmente mapeados no *back-end*, e para sua resolução foram criados objetos temporários para o tratamento do objeto antes do mesmo ser enviado junto à requisição.

Para uma compreensão mais aprofundada dos requisitos do trabalho, o autor participou como instrutor de um curso de extensão. Essa experiência proporcionou uma visão clara e objetiva das principais dificuldades enfrentadas no controle dos cursos. Dessa forma é possível concluir que o objetivo principal definido para este trabalho foi alcançado, atendendo as necessidades principais dos professores e dos instrutores.

Adequação da aplicação para que atenda o controle individual de transporte por usuário, filtragens por colunas nas listas para facilitar a busca por propriedades, gerar relatórios por equipes, por aluno, por curso, por ano, uma opção para o aluno que deseja ser instrutor se voluntariar, além de controle de notas de trabalho para alunos são indicações de trabalhos futuros para complementar o que foi desenvolvido e apresentado neste trabalho.



## REFERÊNCIAS

- AFONSO, Alexandre. **O que é Angular?** Disponível em: <https://blog.algaworks.com/o-que-e-angular/> Acesso em: 28 ago. 2019.
- AFONSO, Alexandre. **O que é Spring MVC?** Disponível em: <https://blog.algaworks.com/spring-mvc/>. Acesso em: 05 nov. 2019.
- AFONSO, Alexandre; NORMANDES, Junior. **Produtividade no desenvolvimento de aplicações web com Spring Boot**. 2 ed. São Paulo, 2017.
- BERNERS-LEE, T. The World Wide Web — **past, present and future**. **Journal of Digital Information**, [S.R.], v.1, n.1, Aug. 1996. ISSN: 1368-7506.
- BRASIL. Câmara dos Deputados. **Projeto de Lei nº 8.035/2010**. Aprova o Plano Nacional de Educação para o decênio 2011-2020 e dá outras providências. Projetos de Leis e Outras Proposições. 2011. Disponível em: <http://www.camara.gov.br/proposicoesWeb/fichadetramitacao?idProposicao=490116>. Acesso em: 20 out. 2019.
- BRASIL. Congresso Nacional. **Lei nº 10.172**, de 9 de janeiro de 2001. Aprova o Plano Nacional de Educação (PNE) e dá outras providências. Diário Oficial da União, de 10 de janeiro de 2001, p. 128. Disponível em: [http://legislacao.planalto.gov.br/legisla/legislacao.nsf/Viw\\_Identificacao/lei%2010.172-2001?OpenDocument](http://legislacao.planalto.gov.br/legisla/legislacao.nsf/Viw_Identificacao/lei%2010.172-2001?OpenDocument). Acesso em: 20 out. 2019.
- BRASIL. **LEI Nº 13.005**, de 25 de junho de 2014. Aprova o Plano Nacional de Educação - PNE e dá outras providências. 2014. Disponível em: [http://www.planalto.gov.br/ccivil\\_03/\\_ato2011-2014/2014/lei/l13005.htm](http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2014/lei/l13005.htm). Acesso em: 20 out. 2019.
- CETIC.BR. **Panorama setorial da Internet: presença das empresas na web. Tecnologia e empresas**. Ano 6, n.2, 2014. Disponível em: [https://cetic.br/media/docs/publicacoes/6/Panorama\\_Setorial\\_7\\_FINAL\\_2-1.pdf](https://cetic.br/media/docs/publicacoes/6/Panorama_Setorial_7_FINAL_2-1.pdf). Acesso em: 27 ago. 2019.
- CETIC.BR. Pesquisa sobre o uso das tecnologias de informação e comunicação nas empresas brasileiras. **TIC empresas 2017**, 2019. Disponível em <https://cetic.br/tics/empresas/2017/empresas/A1A/expandido>. Acesso em 27 ago. 2019.
- CGI.BR. **TIC Empresas 2012 revela que metade das empresas de grande porte já usam as redes sociais**. 2014. Disponível em: < <http://www.cgi.br/noticia/tic-empresas-2012-revela-que-metade-das-empresas-de-grande-porte-ja-usam-as-redes-sociais/>>. Acesso em: 18 nov. 2019.
- FORUM DE PRÓ-REITORES DE EXTENSÃO DAS INSTITUIÇÕES DE EDUCAÇÃO SUPERIOR PÚBLICAS BRASILEIRAS. **Política nacional de extensão universitária**. 2012.
- IMPERATORE, Simone Loureiro Brum; PEDDE, Valdir; IMPERATORE, Jorge Luis Ribeiro. **Curricularizar a extensão ou extensionalizar o currículo? Aportes**

**teóricos e práticas de integração curricular da extensão ante a estratégia 12.7** do PNE. XV COLÓQUIO INTERNACIONAL DE GESTÃO UNIVERSITÁRIA – CIGU. Desafios da Gestão Universitária no Século XXI. Mar del Plata – Argentina. 2015, p. 1-16.

JACYNTHO, Mark Douglas de Azevedo. **Processos para desenvolvimento de aplicações web**. Monografias em Ciência da Computação, n. 23/09, 2008. Disponível em: [ftp.inf.puc-rio.br > pub > docs > techreports > 09\\_23\\_jacyntho](ftp.inf.puc-rio.br/pub/docs/techreports/09_23_jacyntho). Acesso em: 27 ago. 2019.

PRESSMAN, Roger. **Engenharia de software**. São Paulo: Makron Books, 2008.

SILVA, Oberdan Dias da. **O que é extensão universitária?** Integração III, v.9, p. 148-149, maio/97. Disponível em: <https://www.ecientificocultural.com/ECC3/oberdan9.htm>. Acesso em: 08 out. 2019.

TELEGINSKI, Diego Estevam; PORTO ALEGRE, Laíze Márcia. **A curricularização da extensão nos cursos da Universidade Tecnológica Federal do Paraná**. Seminário de extensão e inovação da UTFPR – 4º SEI-UTFPR, 2014, p. 1-7.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO. **O que é a extensão universitária**. Disponível em: <http://www.proex.ufes.br/o-que-%C3%A9-extens%C3%A3o-universit%C3%A1ria>. Acesso em: 08 out. 2019.

ZANETI JUNIOR, Luiz Antonio; VIDAL, Antonio Geraldo da Rocha. Construção de sistemas de informação baseados na tecnologia web. **R. Adm.**, São Paulo, v.41, n.3, 2004.