



PEDRO HENRIQUE PEREIRA SOARES

**DESENVOLVIMENTO DE UM SISTEMA DE EGRESSOS DO CURSO  
TADS DO CAMPUS CAMPINAS DO IFSP**

CAMPINAS

2024

PEDRO HENRIQUE PEREIRA SOARES

**DESENVOLVIMENTO DE UM SISTEMA DE EGRESSOS DO CURSO TADS DO  
CAMPUS CAMPINAS DO IFSP**

Trabalho de Conclusão de Curso apresentado  
como parte dos requisitos para obtenção do  
diploma do Curso Análise e Desenvolvimento  
de Sistemas do Instituto Federal de Educação,  
Ciência e Tecnologia Campus Campinas.

Orientador: Prof.<sup>a</sup> Zady Castañeda Salazar.

CAMPINAS

2024

Ficha Catalográfica  
Instituto Federal de São Paulo – Campus Campinas  
Biblioteca “Pedro Augusto Pinheiro Fantinatti”  
Rosângela Gomes - CRB8/8461

S676d	<p>Soares, Pedro Henrique Pereira</p> <p>Desenvolvimento de um sistema de egressos do curso TADS do campus Campinas do IFSP. / Pedro Henrique Pereira Soares. Campinas, SP: [s.n.], 2024.</p> <p>51 f. : il.</p> <p>Orientadora: Ma. Zady Castaneda Salazar</p> <p>Trabalho de Conclusão de Curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas. Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, 2024.</p> <p>1. Acompanhamento de egressos. 2. Desenvolvimento de sistemas web. 3. Node.JS. 4. React. 5. Typescript. I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas, Curso de Análise e Desenvolvimento de Sistemas. II. Título.</p>
-------	---

ATA N.º 23/2024 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - TADS

Na presente data, realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado DESENVOLVIMENTO DE UM SISTEMA DE EGRESSOS DO CURSO TADS DO CAMPUS CAMPINAS DO IFSP, apresentado(a) pelo(a) aluno(a) Pedro Henrique Pereira Soares (CP3002659) do CURSO DE TECNÓLOGO EM ANÁLISE DE SISTEMAS (campus Campinas). Os trabalhos foram iniciados às 20h:00 pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	Instituição	Presença (Sim/Não)
Zady Castaneda Salazar (Presidente/Orientador)	IFSP	sim
Andreiwid Sheffer (Examinador 1)	IFSP	sim
Fabio Feliciano de Oliveira(Examinador 2)	IFSP	sim

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do(a) candidato(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo(a) aluno(a), tendo sido atribuído o seguinte resultado:

☒ [ X ] Aprovado(a)                      ☐ [ ] Reprovado(a)

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino em nome dos demais membros da banca examinadora.

IFSP Campus Campinas - 05/12/2024

Documento assinado eletronicamente por:

- Andreiwid Sheffer Correa, PROFESSOR ENS BASICO TECN TECNOLÓGICO, em 05/12/2024 20:49:08.
- Zady Castaneda Salazar, PROFESSOR ENS BASICO TECN TECNOLÓGICO, em 06/12/2024 12:03:16.
- Fabio Feliciano de Oliveira, PROFESSOR ENS BASICO TECN TECNOLÓGICO, em 20/12/2024 17:57:26.

Este documento foi emitido pelo SUAP em 05/12/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifsp.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 856552  
Código de Autenticação: dd8cb79217



## RESUMO

O presente trabalho apresenta o desenvolvimento de um sistema para o acompanhamento de egressos do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do IFSP Campus Campinas. A criação de um mecanismo eficiente para monitorar o progresso dos ex-alunos tem como objetivo impactar positivamente a avaliação institucional realizada pelo Ministério da Educação, especialmente nos critérios estabelecidos pelo Sistema Nacional de Avaliação da Educação Superior (Sinaes). O sistema proposto visa coletar informações relevantes dos egressos por meio de formulários online, armazenando os dados em um banco de dados seguro e disponibilizando-os para análise por meio de gráficos e tabelas. A metodologia envolve a pesquisa das políticas de acompanhamento de egressos do IFSP, o desenvolvimento de uma aplicação web utilizando tecnologias como *React*, *Node.js* e *MySQL*. Espera-se que o sistema contribua para a melhoria da qualidade do ensino, permita a identificação de falhas no processo de formação, fortaleça o relacionamento entre a instituição e seus egressos e atenda aos critérios de avaliação do Ministério da Educação, elevando o conceito institucional.

**Palavras-chave:** acompanhamento de egressos; desenvolvimento de sistemas *web*; Node.JS; React; Typescript.

## ABSTRACT

*This paper presents the development of a system for monitoring graduates of the Technology in Systems Analysis and Development program at the Federal Institute of São Paulo (IFSP), Campinas Campus. The creation of an efficient mechanism to track the progress of former students aims to positively influence the institutional evaluation conducted by the Ministry of Education, particularly regarding the criteria established by the National Higher Education Evaluation System (Sinaes). The proposed system seeks to collect relevant information from alumni through online forms, store the data in a secure database, and make it available for analysis using graphs and tables. The methodology involves researching IFSP's alumni monitoring policies and developing a web application utilizing technologies such as React, Node.js, and MySQL. It is expected that the system will contribute to improving the quality of education, enable the identification of shortcomings in the training process, strengthen the relationship between the institution and its alumni, and meet the Ministry of Education's evaluation criteria, thereby enhancing the institution's reputation.*

**Keywords:** *alumni; web systems development; Node.js; React; TypeScript.*

## LISTA DE FIGURAS

<b>Figura 1</b> - Arquitetura geral do sistema.....	18
<b>Figura 2</b> - Diagrama da estrutura do banco de dados .....	20
<b>Figura 3</b> - Estrutura de pastas do back-end .....	25
<b>Figura 4</b> - Trecho de código que realiza a conexão com o banco de dados .....	26
<b>Figura 5</b> - Exemplo de função com consulta SQL parametrizada.....	26
<b>Figura 6</b> - Arquivo userRoutes.ts .....	27
<b>Figura 7</b> - Validações de dados no userController.ts .....	28
<b>Figura 8</b> - Função de criptografia utilizando a biblioteca bcrypt .....	28
<b>Figura 9</b> - Login do usuário no userController.ts .....	29
<b>Figura 10</b> - Exemplo de validação de permissão.....	29
<b>Figura 11</b> - Estrutura de pastas do front-end .....	33
<b>Figura 12</b> - Roteamento do sistema no arquivo routes.tsx .....	34
<b>Figura 13</b> - UserProvider do useContext.tsx .....	35
<b>Figura 14</b> - getUserData do userService.ts.....	35
<b>Figura 15</b> – Checagem de dependências do componente Form .....	37
<b>Figura 16</b> – Validação de campos na criação de questão .....	37
<b>Figura 17</b> - Tela de Login.....	39
<b>Figura 18</b> - Tela de Registro.....	40
<b>Figura 19</b> - Dashboard do Egresso .....	41
<b>Figura 20</b> - Preenchimento do Formulário .....	42
<b>Figura 21</b> - Página de Análise de Dados .....	43
<b>Figura 22</b> - Página de Formulários .....	44
<b>Figura 23</b> - Página de Edição de Formulários .....	44
<b>Figura 24</b> - Modal de adição de questão .....	45

## LISTA DE ABREVIATURAS E SIGLAS

<b>API</b>	<i>Application Programing Interface</i>
<b>IFSP</b>	<b>Instituto Federal de Educação, Ciência e Tecnologia de São Paulo</b>
<b>MVC</b>	<i>Model-View-Controller</i>
<b>JSON</b>	<i>JavaScript Object Notation</i>
<b>JWT</b>	<i>JSON Web Tokens</i>
<b>TADS</b>	<b>Tecnologia em Análise e Desenvolvimento de Sistemas</b>
<b>SUAP</b>	<b>Sistema Unificado de Administração Pública</b>
<b>LDAP</b>	<i>Lightweight Directory Access Protocol</i>
<b>SQL</b>	<i>Structured Query Language</i>



## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>10</b>
<b>1.1 Justificativa .....</b>	<b>11</b>
<b>1.2 Objetivos.....</b>	<b>12</b>
<i>1.2.1 Objetivos específicos .....</i>	<i>12</i>
<b>2 FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>13</b>
<b>2.1 Egresso.....</b>	<b>13</b>
<i>2.1.1 Pesquisa com egressos.....</i>	<i>13</i>
<i>2.1.2 Programa de acompanhamento de egressos do IFSP.....</i>	<i>13</i>
<b>2.2 Bancos de dados.....</b>	<b>14</b>
<i>2.2.1 Sistema de banco de dados .....</i>	<i>14</i>
<i>2.2.2 SQL.....</i>	<i>14</i>
<u><i>2.2.2.1 MySQL .....</i></u>	<u><i>15</i></u>
<b>2.3 Javascript .....</b>	<b>15</b>
<i>2.3.1 Typescript .....</i>	<i>15</i>
<i>2.3.2 Node .....</i>	<i>15</i>
<i>2.3.3 React.....</i>	<i>16</i>
<b>3 METODOLOGIA.....</b>	<b>17</b>
<b>3.1 Projetos semelhantes .....</b>	<b>17</b>
<b>3.2 Arquitetura do sistema.....</b>	<b>18</b>
<b>3.3 Desenvolvimento do banco de dados.....</b>	<b>19</b>
<i>3.3.1 Modelagem dos dados.....</i>	<i>19</i>
<b>3.4 Desenvolvimento do backend .....</b>	<b>23</b>
<i>3.4.1 Bibliotecas utilizadas .....</i>	<i>23</i>
<i>3.4.2 Estrutura do backend .....</i>	<i>24</i>
<i>3.4.4 Integração com o banco de dados .....</i>	<i>25</i>
<i>3.4.3 Implementação das funcionalidades principais .....</i>	<i>27</i>
<u><i>3.4.3.1 Autenticação e autorização .....</i></u>	<u><i>27</i></u>
<u><i>3.4.3.2 Gerenciamento de formulários com dependências .....</i></u>	<u><i>30</i></u>
<u><i>3.4.3.3 Outras funcionalidades .....</i></u>	<u><i>31</i></u>
<b>3.5 Desenvolvimento do frontend.....</b>	<b>32</b>
<i>3.5.1 Escolha das tecnologias.....</i>	<i>32</i>

3.5.2 Estrutura do frontend.....	32
3.5.3 Implementação das funcionalidades principais .....	33
<u>3.5.3.1 Roteamento</u> .....	33
<u>3.5.3.2 Fluxo de login e registro:</u> .....	33
<u>3.5.3.3 Autenticação de usuários</u> .....	34
<u>3.5.3.4 Dashboard do egresso e administrador</u> .....	36
4 RESULTADOS .....	39
4.1 Fluxos de uso do sistema .....	39
4.1.1 Fluxos do usuário egresso.....	39
<u>4.1.1.1 Tela de login e registro</u> .....	39
<u>4.1.1.2 Preenchimento de formulários</u> .....	41
4.1.2 Fluxos do usuário administrador.....	42
<u>4.1.2.1 Página de análise de dados</u> .....	43
<u>4.1.2.2 Página de criação e edição de formulários</u> .....	44
5 CONCLUSÕES.....	47
REFERÊNCIAS .....	50

## 1 INTRODUÇÃO

A educação apresenta inúmeras variáveis que precisam estar em equilíbrio para a sua qualidade. Concepções pedagógicas utilizadas atualmente não são as mesmas de dez ou vinte anos atrás. Além disso, entendimentos passados foram substituídos por noções atualizadas no que diz respeito às responsabilidades de instituições de educação, seja no âmbito das normas internas das próprias instituições ou por força legal.

Uma vez formado, o aluno de instituições escolares no passado não era visto como responsabilidade da instituição, hoje essa prática está ultrapassada, tendo em vista as mudanças recentes e novos entendimentos educacionais, isso obriga os cursos das IES (Instituições de ensino superior) a revisar suas práticas pedagógicas. Pesquisas que inicialmente são realizadas esporadicamente, se colocam com uma boa fonte para auxiliar tomadas de decisões, e são um ponto de partida para a revisão de modo a influenciá-la. Por esse motivo, ter mais conhecimento sobre as metodologias e suas contribuições da pesquisa é de grande importância para melhorar o desenvolvimento da instituição (Paul, 2015).

Atualmente as pesquisas se dão em ferramentas criadas pelas próprias IES, um exemplo seria a plataforma Alumni da USP (Universidade de São Paulo), ou o Portal de Egressos do IFRO (Instituto Federal de Rondônia) que buscam manter um vínculo com os ex-alunos, guardando um histórico do desenvolvimento pessoal, técnico e profissional do discente.

Esses sistemas possuem a funcionalidade de armazenar informações relevantes dos universitários em múltiplos períodos, por exemplo no final do curso, e posteriormente mais vezes nos próximos anos, independentemente se o aluno é diplomado, desistente, realizou transferência ou é jubilado.

No Brasil, as instituições são avaliadas regularmente em vários pontos pelo Ministério da Educação, sendo um deles o acompanhamento do egresso. Ponto esse que nos dias de hoje o campus Campinas do Instituto Federal de São Paulo não satisfaz completamente os critérios, por não ter um sistema capaz de realizar esse acompanhamento.

Neste trabalho foi desenvolvido um sistema que armazena informações relevantes para o IFSP (Instituto Federal de São Paulo) – Campus Campinas sobre os egressos do curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Sendo capaz de apresentar

formulários e armazenar as respostas dos estudantes em múltiplos momentos em um banco de dados disponível para acesso somente pela instituição, além de apresentar os dados em gráficos e tabelas pelo próprio sistema, de modo a suprir essa necessidade de guardar as opiniões dos ex-alunos em relação a qualidade da formação no curso, sua situação no mercado de trabalho, dados estatísticos e outras informações gerais.

Como metodologia de validação é necessário realizar a análise dos resultados obtidos. Esses dados colhidos servirão para a tomada de decisão no âmbito no campus Campinas do IFSP, especialmente para a formulação de políticas no que diz respeito ao acompanhamento do aluno mesmo depois de formado.

### 1.1 Justificativa

Em 2004, o Ministério da Educação criou o Sistema Nacional de Avaliação da Educação Superior (Sinaes), que é de responsabilidade e supervisionado pela Comissão Nacional de Avaliação da Educação Superior (Conaes).<sup>1</sup>

O Conaes estabeleceu as diretrizes dos instrumentos de avaliação, que estão presentes no instrumento de avaliação arquitetado e implementado pelo Inep (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira).<sup>2</sup>

No instrumento dessas avaliações das IES (Instituições de Ensino Superior) que ocorrem anualmente, temos alguns indicadores que são usados na hora de calcular a pontuação final da instituição, que se baseia em determinados critérios para determinar o conceito, que seria a “nota” em um indicador específico, que vai de 1 a 5.<sup>3</sup>

Um desses indicador é a “Política de acompanhamento do egresso”, em que para alcançar um conceito de valor 5, os critérios utilizados são:

1)“Mecanismos para a criação de uma base de dados, com informações atualizadas dos egressos.”;

2)“Mecanismos para a promoção de um relacionamento contínuo entre a instituição e seus egressos.”;

3)“Mecanismos para avaliar a adequação da formação do profissional para o mercado de trabalho.”;

---

<sup>1</sup> MINISTÉRIO DA EDUCAÇÃO - MEC. Sinaes.

<sup>2</sup> MINISTÉRIO DA EDUCAÇÃO - MEC. Avaliação externa das instituições de educação superior: diretrizes e instrumento. 2006.

<sup>3</sup> INEP. Indicadores de Qualidade da Educação Superior.

4)“Mecanismos de utilização das opiniões dos egressos para aperfeiçoamento do processo de formação.”.

Atualmente o campus Campinas do Instituto Federal de São Paulo não possui uma ferramenta para armazenar informações e opiniões dos egressos, para guardar os dados e validar a efetividade da formação realizada no curso tecnológico de análise e desenvolvimento de sistemas que o campus oferece, o que se caracteriza como uma possibilidade de melhoria, uma vez que o IFSP não atende completamente todos os critérios da política de acompanhamento do egresso.

Os dados oriundos das respostas dos egressos, além de tirar a defasagem em atender às diretrizes dos Sinaes, irão melhorar os aspectos no que diz respeito a qualidade da educação oferecida pelo curso de TADS do IFSP Campus Campinas, permitindo a identificação de eventuais falhas ou deficiências no processo de formação, bem como avaliar a adequação da formação do profissional para o mercado de trabalho e também pode ser usado como instrumento para a melhoria e evolução do projeto pedagógico do curso, o que contribuirá para aperfeiçoar a qualidade do ensino oferecido e, conseqüentemente, elevar a qualidade da educação como um todo.

## **1.2 Objetivos**

Desenvolver um sistema de egressos que colete informações referentes aos alunos egressos do curso Tecnologia em Análise e Desenvolvimento de Sistemas do IFSP Campus Campinas por meio de formulários e disponibilize os dados por meio de gráficos e tabelas para análise.

### ***1.2.1 Objetivos específicos***

- a) Pesquisar a política de acompanhamento de egressos do IFSP.
- b) Desenvolver um sistema web capaz de exibir dados do formulário filtrando por diferentes critérios de busca, armazenar as perguntas do formulário e as respostas de cada estudante em um banco de dados.
- c) Validar e verificar as funcionalidades de sistema de egressos.

## **2 FUNDAMENTAÇÃO TEÓRICA**

Nesta seção, apresentamos e detalhamos os principais tópicos que fundamentam o desenvolvimento de nosso sistema de egressos.

### **2.1 Egresso**

O dicionário define egresso como “Aquele que deixou de fazer parte do convento” ou “Que já não faz parte de um grupo”<sup>4</sup>, mas no contexto desse projeto se refere ao ex-estudante que deixou de fazer parte dos discentes do Instituto Federal do curso de Análise e Desenvolvimento de Sistemas.

#### ***2.1.1 Pesquisa com egressos***

Espartel (2009) afirma a importância da execução recorrente de pesquisas com os egressos, porque além de não só possibilitar uma perspectiva abrangente do percurso dos alunos após o curso, é proveitoso usá-la como instrumento de controle, já que aspectos avaliados podem servir como indicadores de desempenho e qualidade do curso, e então como referências para possíveis aperfeiçoamentos que sejam considerados vantajosos.

#### ***2.1.2 Programa de acompanhamento de egressos do IFSP***

De acordo com o PDI (Plano de Desenvolvimento Institucional) de 2019/2023<sup>5</sup>, o Instituto Federal de São Paulo atualmente possui uma iniciativa chamada Programa de Acompanhamento de Egressos que busca acompanhar o desempenho dos seus ex-alunos no mercado de trabalho, além de coletar informações sobre sua satisfação com a formação recebida na instituição. O objetivo é identificar possíveis melhorias nos cursos oferecidos, bem como aprimorar a relação do IFSP com seus egressos.

O programa é realizado por meio de pesquisas periódicas, com perguntas sobre sua trajetória profissional, a utilização dos conhecimentos adquiridos durante o curso, suas

---

<sup>4</sup> DICIO. Egresso

<sup>5</sup> IFSP. PDI 2019-2023. Disponível em: <https://www.ifsp.edu.br/component/content/article?id=533>

perspectivas e expectativas em relação ao mercado de trabalho, entre outros aspectos relevantes, que buscam mapear a atuação dos egressos e avaliar sua formação em relação às necessidades do mercado. As informações coletadas são utilizadas para identificar as principais demandas dos empregadores e adaptar o conteúdo dos cursos de acordo com as necessidades identificadas.

Além disso, o Programa de Acompanhamento de Egressos do IFSP busca fortalecer a rede de relacionamento entre os ex-alunos e a instituição, oferecendo serviços e benefícios exclusivos para os egressos, como oportunidades de emprego e atualização profissional.

## **2.2 Bancos de dados**

Um banco de dados é uma coleção de dados, na maioria das vezes usados por uma aplicação. Quando é necessário armazenar muitas informações, existe a necessidade de estruturar esses dados para que eles se mantenham corretos e relacionados. Por esse motivo surge a teoria chamada modelo relacional de dados, que define padrões para a estrutura e a maneira que esses dados serão exibidos para o usuário (Date, 2004).

### **2.2.1 Sistema de banco de dados**

Date (2004) define sistema de bancos de dados como “um sistema computadorizado cuja finalidade geral é armazenar informações e permitir que os usuários busquem e atualizem informações quando as solicitar”, onde os usuários do sistema podem realizar ações como: inserir dados, realizar buscas entre os dados armazenados, excluir os dados existentes e alterar os dados já guardados.

### **2.2.2 SQL**

*Structured Query Language (SQL)*, em português Linguagem de Consulta Estruturada, é uma linguagem para dados de bancos de dados relacionais, ela é o padrão mundial para linguagens de dados, sendo usada para descrever as tabelas, as colunas presentes nelas e o tipo de dado presente nessa coluna de um banco relacional. Além de ser utilizando ela que as operações de inserir, buscar, excluir e alterar são realizadas no banco pelo usuário. (MELTON,1996)

### **2.2.2.1 MySQL**

*MySQL* é um sistema de gerenciamento de banco de dados (SGBD) relacional que utiliza a linguagem *SQL*. Ele é portátil e roda em sistemas operacionais comerciais, como *Windows* e *Mac*, e em hardware até servidores corporativos. É um dos SGBDs mais populares nos dias de hoje, capaz de lidar com bancos de dados de bilhões de linhas (Dubois, 2008).

## **2.3 Javascript**

Segundo Flanagan (2004), “*Javascript* é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais”. Essa linguagem é usada na maioria dos sites hoje em dia, é de uso geral e eficiente, de modo a ser uma escolha recomendada na hora de desenvolver o *front-end* de um sistema.

### **2.3.1 Typescript**

*TypeScript* é uma linguagem de programação de código aberto desenvolvida pela *Microsoft* que estende o *JavaScript* ao adicionar tipagem estática opcional e outros recursos avançados. Segundo a documentação oficial da *Microsoft* (“*The starting point for learning TypeScript*”, [s.d.]), “*TypeScript* é um superconjunto de *JavaScript* que adiciona tipos opcionais, classes e módulos, permitindo o desenvolvimento de aplicações *JavaScript* em escala empresarial”.

A utilização do *TypeScript* proporciona benefícios como detecção de erros em tempo de desenvolvimento, autocompletar mais preciso e melhor refatoração de código.

### **2.3.2 Node**

O site oficial do *Node* (<https://nodejs.org/>) o define como “um ambiente de execução *JavaScript* assíncrono orientado a eventos”. Ele possui muitos benefícios, como: Desenvolvimento de sistemas web em apenas uma linguagem, utiliza o *JSON* (*JavaScript Object Notation*) que é um formato de troca de dados muito popular atualmente e é nativo do *JavaScript*, e é compatível com a maioria dos navegadores (Cantelon et al., 2014).



### 2.3.3 *React*

*React* é uma biblioteca *JavaScript* para construção de interfaces do usuário por meio de componentes reutilizáveis, é de código aberto e mantida pelo *Facebook*.<sup>6</sup>

Ela permite o desenvolvimento de aplicações web que não precisam de atualizações frequentes da página (Aplicação de Página Única), e é usada como a *View* no padrão *Model-View-Controller (MVC)*, além de abstrair o *Document Object Model (DOM)*, fazendo com que a experiência de desenvolvimento seja muito mais eficiente e performante (Aggarwal,2018).

---

<sup>6</sup> REACT. Disponível em: < <https://pt-br.reactjs.org/> >.

### **3 METODOLOGIA**

Para a elaboração desse trabalho, de modo a alcançar o objetivo proposto, estão previstos os seguintes procedimentos metodológicos:

#### **3.1 Projetos semelhantes**

Em 2014 na Universidade Federal de Santa Catarina (UFSC) foi realizado um projeto da criação de um sistema de acompanhamento de egressos. Este projeto apresentou a importância do acompanhamento dos egressos para a melhoria da qualidade do ensino, pesquisa, extensão e administração universitária, e como isso contribui para o aprimoramento das atividades institucionais (Silva, 2014).

Assim como no projeto da UFSC, o objetivo do sistema de egressos do IFSP é buscar subsídios para melhorar a qualidade do ensino, identificar potencialidades e limitações a missão institucional, com a intenção de aprimorar a estrutura política-pedagógica e a gestão do campus, e, conseqüentemente, melhorar a qualidade de seus serviços.

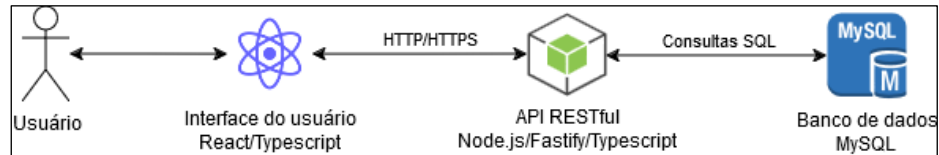
A implantação do Sistema de Acompanhamento dos Egressos na UFSC, que segue as determinações do Sistema Nacional de Avaliação da Educação Superior (SINAES), mostrou-se efetiva para promover uma cultura de avaliação e melhorar a qualidade do ensino, pesquisa, extensão e administração universitária. A participação dos egressos é fundamental para identificar as necessidades do mercado de trabalho e ajustar as propostas políticas pedagógicas da instituição, visando a uma formação acadêmica e profissional de qualidade.

Além disso, o Portal de Acompanhamento dos Egressos permitiu fortalecer o relacionamento entre a UFSC e os egressos, obtendo avaliações e sugestões para melhorias nos serviços da instituição. A participação dos egressos nesse processo criou um vínculo de parceria e comprometimento institucional, fortalecendo a missão da universidade e contribuindo para a qualidade de vida em sociedade (Silva, 2014).

Nesse sentido, o sistema de egressos proposto para o IFSP campus Campinas segue os mesmos objetivos, contribuir para a melhoria da qualidade do ensino, pesquisa, extensão e administração do campus, fortalecendo o relacionamento com os egressos e atendendo às expectativas da sociedade.

### 3.2 Arquitetura do sistema

**Figura 1** - Arquitetura geral do sistema



Fonte: Elaborado pelo autor (2024)

Conforme ilustrado na **Figura 1**, o sistema de egressos desenvolvido é estruturado em duas camadas principais: *front-end* e *back-end*. Além dessas camadas, várias tecnologias e ferramentas foram utilizadas para suportar e otimizar o funcionamento do sistema, como o *React* para a interface do usuário, o *Node.js* para o servidor de aplicação e o *MySQL* para o armazenamento de dados.

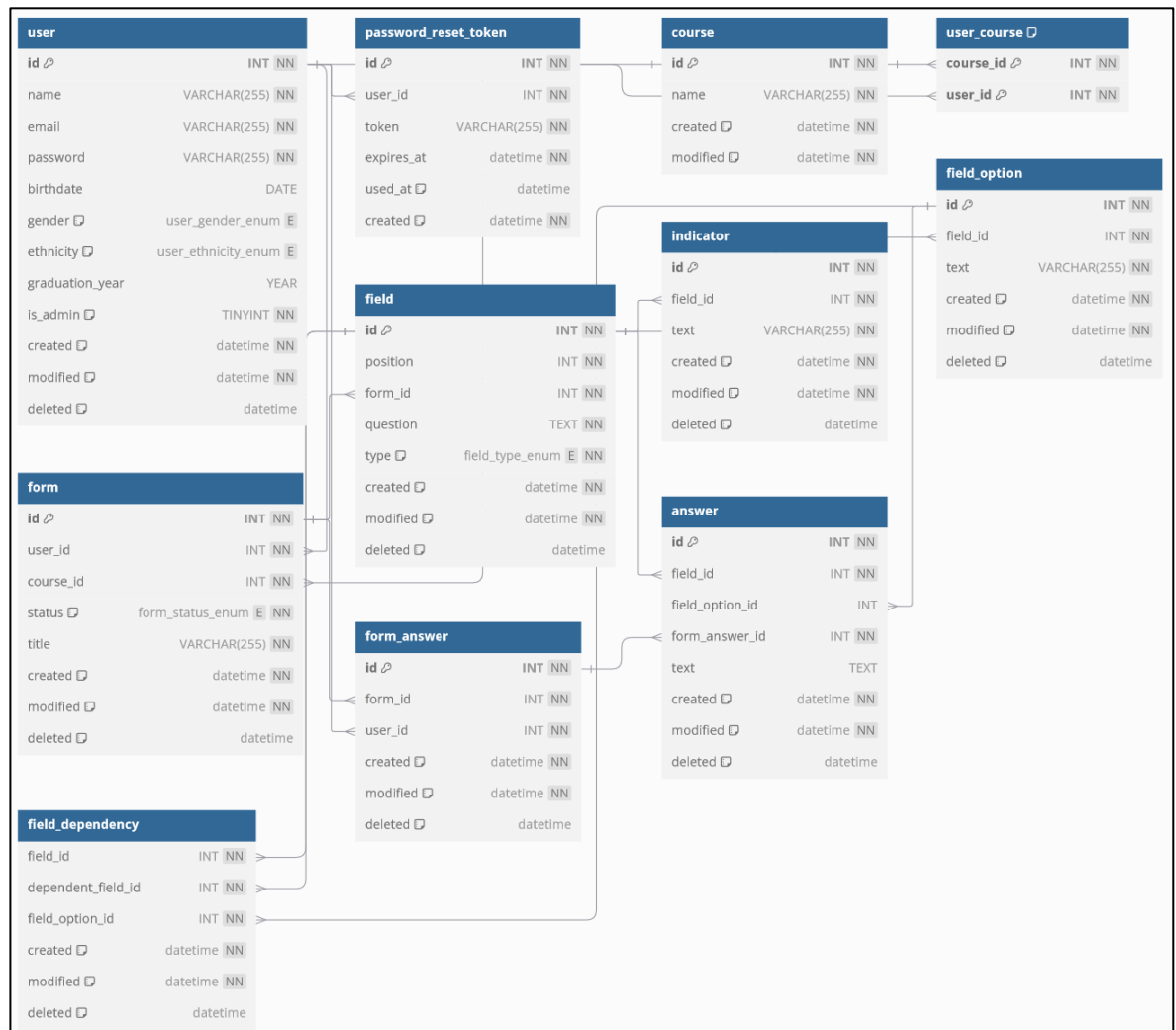
- **Front-end:** Esta é a camada com a qual os usuários interagem diretamente. Desenvolvido utilizando *React* com *TypeScript*, o *front-end* proporciona uma interface dinâmica e responsiva. Os usuários podem acessar o sistema via aplicação web, realizar cadastro, efetuar login, atualizar seus dados pessoais e consultar informações relevantes.
- **Back-end:** Responsável por processar as requisições do *front-end*, aplicar as regras de negócio e interagir com o banco de dados. Implementado com *Node.js*, *Fastify* e *TypeScript*, o *back-end* fornece uma *API RESTful* que expõe *end-points* para as operações necessárias. A estrutura do *back-end* segue o padrão MVC (*Model-View-Controller*), organizada em rotas, controladores e modelos, facilitando a manutenção e escalabilidade do sistema. A autenticação é gerenciada através de *JSON Web Tokens (JWT)*, garantindo a segurança no acesso às funcionalidades protegidas.
- **Banco de Dados MySQL:** Utilizado para o armazenamento persistente dos dados, como informações dos egressos, credenciais de acesso e outros dados relevantes. A modelagem do banco de dados foi realizada para atender aos requisitos do sistema, assegurando a integridade e consistência das informações.
- **Integração Front-end e Back-end:** A comunicação entre o *front-end* e o *back-end* é realizada através de requisições HTTP/HTTPS, utilizando o formato JSON para troca de dados.

### 3.3 Desenvolvimento do banco de dados

Para o armazenamento dos dados do sistema de egressos, optou-se pelo uso do banco de dados relacional MySQL, devido à sua ampla adoção, facilidade de uso e compatibilidade com as tecnologias empregadas no *back-end*. A seguir, apresenta-se a modelagem dos dados desenvolvida para atender aos requisitos do sistema.

#### 3.3.1 Modelagem dos dados

A modelagem dos dados foi realizada considerando as necessidades do sistema no armazenamento de informações, especificamente dos dados dos usuários (tanto administradores quanto alunos egressos), dos formulários, suas questões e respostas, garantindo integridade, eficiência e facilidade de manutenção. O diagrama apresentado na **Figura 2** ilustra a estrutura geral do banco de dados.

**Figura 2 - Diagrama da estrutura do banco de dados**

Fonte: Elaborado pelo autor (2024)

A coluna de identificação **id** está presente em quase todas as tabelas. Além disso, é importante destacar o uso das colunas **created**, **modified** e **deleted** em todas as tabelas do sistema, que melhoram a auditoria e o rastreamento, permitindo o acompanhamento das alterações nos registros e a implementação de exclusões lógicas, mantendo um histórico dos dados. Tendo essas informações em mente, a seguir será apresentado informações relevantes sobre as entidades do banco.

## 1. Cursos

Para o gerenciamento dos cursos, utilizamos a tabela **course**, que representa os cursos oferecidos pelo campus. Nessa tabela, é armazenado apenas o nome do curso no campo **name**. Considerando o escopo do projeto, atualmente o único curso registrado é o de Tecnologia em

Análise e Desenvolvimento de Sistemas (TADS), mas a estrutura permite a adição de novos cursos no futuro.

## 2. Usuários

A tabela principal para os usuários é a ***user***, que contém informações importantes. Alguns dos campos principais são:

- ***name* (Nome)**: Nome do aluno ou do administrador do sistema.
- ***email***: Endereço de e-mail utilizado para login do usuário.
- ***password* (Senha)**: (Senha): Senha criptografada para autenticação.
- ***is\_admin* (é administrador)**: Indica se o usuário é um egresso (0) ou um administrador (1).
- ***birthdate* (Data de nascimento), *gender* (Gênero), *ethnicity* (Etnia)**: Campos referentes a dados pessoais do aluno, utilizados na análise de dados para agrupamento de indicadores. Não são utilizados no caso de um usuário administrador.
  - O campo *gender* é do tipo enum, contendo as seguintes opções:
    - *male* (homem)
    - *female* (mulher)
    - *trans\_male* (homem trans)
    - *trans\_female* (mulher trans)
    - *non\_binary* (não binário)
    - *other* (outro)
  - O campo *ethnicity* também é do tipo enum e as suas possíveis opções são:
    - *white*(branca),
    - *black*(preta),
    - *brown*(parda),
    - *yellow*(amarela),
    - *indigenous*(indígena),
    - *not\_declared* (Não informado)
- ***graduation\_year*(Ano de graduação)**: Usado no filtro com mesmo nome na aba de Dados do administrador.

A associação entre um usuário e um curso é realizada através da tabela ***user\_course***, que armazena o *user\_id* e o *course\_id*, estabelecendo um relacionamento muitos-para-muitos.

Essa tabela é preenchida apenas para usuários egressos; administradores não possuem um registro nela.

Outra tabela relevante é a *password\_reset\_token*, onde são salvos os tokens de recuperação de senha, com validade limitada e uso único.

### 3. Formulários

Para os formulários, a tabela principal é a *form*, que representa o registro de um formulário específico. Esta tabela está relacionada a um único curso por meio da coluna *course\_id* e ao usuário administrador que o criou através da coluna *user\_id*. O campo status indica o estado do formulário, podendo ser *draft* (rascunho) ou *published* (publicado).

### 4. Questões

As questões dos formulários estão salvas na tabela *field*, que contém o *form\_id* referenciando o formulário ao qual pertencem, o texto da questão no campo *question*, sua posição no campo *position* e seu tipo em *type*.

O campo *type* é do tipo *ENUM*, com os seguintes valores:

- *text* (texto)
- *single\_choice* (escolha única)
- *multiple\_choice* (múltipla escolha)
- *date* (data)

Para questões do tipo escolha única ou múltipla escolha, as opções disponíveis são armazenadas na tabela *field\_option* que possui o campo *text* com o texto conteúdo da opção.

A tabela *field\_dependency* é responsável por armazenar as regras de dependência entre questões, permitindo que a exibição de uma pergunta dependa da resposta fornecida em outra.

### 5. Respostas de formulários

As respostas de um formulário são armazenadas em duas tabelas:

- *form\_answer*: Centraliza a resposta de um aluno específico a um formulário, contendo os campos *user\_id* (usuário que respondeu) e *form\_id* (formulário respondido).
- *answer*: Armazena as respostas individuais a cada pergunta do formulário. Inclui os campos *form\_answer\_id* (referência à resposta do formulário), *field\_id* (identificador da questão), *field\_option\_id* (identificador da opção selecionada, se aplicável) e *text* (conteúdo da resposta para questões de texto ou data).

A modelagem dos dados apresentada foi fundamental para atender às necessidades específicas do sistema de egressos, proporcionando uma base sólida para o armazenamento e gerenciamento eficiente das informações. A estrutura do banco de dados, cuidadosamente planejada, permite que as entidades principais—cursos, usuários, formulários e respostas—interajam de forma coerente e integrada. O uso de chaves primárias e estrangeiras assegura a integridade referencial, enquanto a implementação de campos de auditoria como *created*, *modified* e *deleted* facilita o rastreamento de alterações e a manutenção dos dados ao longo do tempo. Além disso, a consideração de aspectos de segurança, como a criptografia de senhas, garante a confidencialidade e privacidade das informações dos usuários.

Dessa forma, a modelagem dos dados não apenas atende aos requisitos funcionais do sistema, mas também estabelece uma estrutura flexível e escalável para futuras evoluções, assegurando a longevidade e eficiência da aplicação.

### 3.4 Desenvolvimento do backend

O *back-end* do sistema foi desenvolvido utilizando *Node.js* com o *framework Fastify*, escrito em *TypeScript*. Essas tecnologias foram escolhidas devido à eficiência que proporcionam no desenvolvimento de *APIs* robustas e escaláveis, possibilitando a interação eficiente entre o *front-end* e o banco de dados.

#### 3.4.1 Bibliotecas utilizadas

Além das tecnologias principais mencionadas, diversas bibliotecas foram empregadas para implementar funcionalidades específicas do sistema:

- ***bcrypt***: Utilizada para a criptografia de senhas, garantindo que as senhas dos usuários sejam armazenadas de forma segura no banco de dados.
- ***crypto***: Usada para a geração de tokens únicos de recuperação de senha, adicionando uma camada extra de segurança ao processo de recuperação de contas.
- ***mysql2***: Biblioteca que facilita a interação com o banco de dados *MySQL*, permitindo executar consultas e manipular dados de forma eficiente.
- ***nodemailer***: Empregada para o envio de e-mails de recuperação de senha, possibilitando a comunicação automática com os usuários para processos de redefinição de senha.



- **@fastify/jwt**: Plugin do *Fastify* utilizado para a geração e verificação de tokens *JWT* (*JSON Web Tokens*), implementando a autenticação baseada em tokens de forma integrada ao *framework*.
- **dotenv**: Responsável por carregar variáveis de ambiente a partir de um arquivo *.env* para o ambiente de execução da aplicação. Isso permite a configuração segura de informações sensíveis, como senhas, chaves de API e credenciais de banco de dados, sem expor esses dados diretamente no código-fonte.

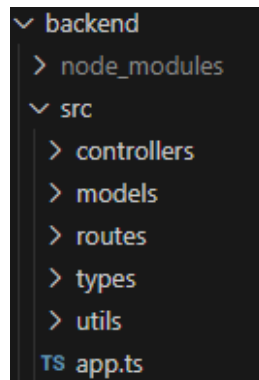
### 3.4.2 Estrutura do backend

O *back-end* foi desenvolvido seguindo o padrão arquitetural *Model-View-Controller* (MVC), o que proporcionou uma separação clara das responsabilidades, facilitando a manutenção e evolução do código. Essa arquitetura é composta por três partes, mas no sistema desenvolvido foram implementadas da seguinte forma:

- **Models** (Modelos): Representam as entidades do sistema e contêm a lógica de acesso e manipulação dos dados no banco de dados.
- **Controllers** (Controladores): Contêm a lógica de negócios, processando as requisições recebidas, interagindo com os Models e definindo as respostas adequadas.
- **Routes** (Rotas): Definem os *end-points* da *API*, mapeando as requisições para os métodos correspondentes nos *Controllers*.

Essa organização permitiu que o desenvolvimento do sistema seja mais eficiente, com componentes bem definidos e reutilizáveis, além de facilitar a identificação e correção de possíveis erros.

**Figura 3** - Estrutura de pastas do *back-end*



Fonte: Elaborado pelo autor (2024)

A estrutura do *back-end* está organizada de forma a refletir a separação de responsabilidades promovida pelo padrão arquitetural MVC. Conforme ilustrado na **Figura 3**, as pastas principais são:

- **controllers/**: Contém os controladores que implementam a lógica de negócio e processam as requisições.
- **models/**: Abriga os modelos que representam as entidades e interagem com o banco de dados.
- **routes/**: Define as rotas da *API*, mapeando os *endpoints* para os controladores correspondentes.
- **types/**: Armazena tipos do *typescript* usados em múltiplos lugares do sistema.
- **utils/**: Contém funções utilitárias e utilizados em diversas partes do *back-end*, como por exemplo a função de conexão com o banco e a função que faz a criptografia da senha.

Além dessas pastas, o arquivo **app.ts** atua como ponto de entrada da aplicação, configurando o servidor e inicializando os componentes necessários.

#### 3.4.4 Integração com o banco de dados

A integração com o banco de dados é um componente fundamental do *back-end* do sistema, permitindo o armazenamento e a recuperação eficiente dos dados necessários para o funcionamento do sistema como um todo. A comunicação com o banco de dados *MySQL* foi estabelecida utilizando a biblioteca **mysql2**, que oferece uma maneira simples e eficaz de executar consultas SQL no *Node.js*.

### Configuração da Conexão:

- Variáveis de Ambiente:
  - As credenciais de acesso ao banco de dados (host, usuário, senha, nome do banco) são armazenadas em variáveis de ambiente, utilizando a biblioteca *dotenv*.
- Estabelecimento da Conexão:
  - Uma conexão com o banco de dados é estabelecida quando o servidor é inicializado, utilizando um *pool* de conexões para melhorar o desempenho e a escalabilidade da aplicação.

**Figura 4** - Trecho de código que realiza a conexão com o banco de dados

```
import mysql from 'mysql2/promise';
import dotenv from 'dotenv';

dotenv.config();

export const db = mysql.createPool({
  host: process.env.DB_HOST || 'localhost',
  user: process.env.DB_USER || 'root',
  password: process.env.DB_PASSWORD || '',
  database: process.env.DB_NAME || 'ifsp_campus_campinas',
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
});
```

Fonte: Elaborado pelo autor (2024)

Como demonstrado na **Figura 4**, a conexão é realizada usando as variáveis de ambiente, isso aumenta a segurança, evitando expor informações sensíveis no código-fonte.

- Interação com o Banco de Dados:
  - Consultas Parametrizadas:
    - As consultas SQL são construídas de forma parametrizada, prevenindo ataques de injeção de SQL. Por exemplo:

**Figura 5** - Exemplo de função com consulta *SQL* parametrizada

```
async findByEmail(email: string): Promise<User | null> {
  const [rows] = await db.execute('SELECT * FROM `user` WHERE email = ?', [email]);
  const users = rows as User[];
  return users.length > 0 ? users[0] : null;
},
```

Fonte: Elaborado pelo autor (2024)

No exemplo da **Figura 5**, vemos uma consulta que busca o usuário com um e-mail específico, essa informação é passada como parâmetro na função `execute`, responsável pela execução de um código SQL no banco.

O uso de consultas parametrizadas não só previne ataques de injeção de SQL, mas também promove uma codificação mais segura e robusta. Que é essencial para proteger os dados sensíveis dos usuários e manter a segurança do sistema.

### 3.4.3 Implementação das funcionalidades principais

#### 3.4.3.1 Autenticação e autorização

Uma das funcionalidades centrais do sistema é a autenticação de usuários. Inicialmente, foi considerada a possibilidade de integrar a autenticação do sistema com o **SUAP** (Sistema Unificado de Administração Pública) utilizando **LDAP** (*Lightweight Directory Access Protocol*), com o objetivo de aproveitar a infraestrutura existente e facilitar o acesso dos usuários já cadastrados na instituição.

No entanto, após entrar em contato com o setor de T.I. do campus, foi informado que não seria possível realizar essa integração devido a restrições institucionais e de segurança. Diante disso, optou-se por desenvolver um sistema de autenticação próprio, implementado no backend utilizando **JSON Web Tokens (JWT)**, a partir da biblioteca **@fastify/jwt**, o fluxo de autenticação inclui:

- Registro de Usuários:

Os usuários se registram passando nome, e-mail, senha e no caso de usuários egressos, a data de nascimento, o ano de graduação, o gênero e a etnia. A rota, como explicado na seção 3.4.2, fica na pasta `/routes`, no arquivo `userRoutes.ts`, que possui a seguinte estrutura, conforme a **Figura 6**:

**Figura 6** - Arquivo `userRoutes.ts`

```
export default async function userRoutes(app: FastifyInstance) {
  app.post('/register', registerUser);
  app.post('/admin-register', { preValidation: [app.authenticate] }, registerUser);
  app.post('/login', loginUser);
  app.get('/me', { preValidation: [app.authenticate] }, getUserProfile);
  app.put('/update-password', { preValidation: [app.authenticate] }, updatePassword);
  app.post('/request-password-reset', requestPasswordReset);
  app.post('/reset-password', resetPassword);
}
```

Fonte: Elaborado pelo autor (2024)

No *userController.ts*, antes de passar as informações que vieram na requisição para o *model* do *user*, validações são feitas para garantir a integridade dos dados, demonstradas na **Figura 7**:

**Figura 7** - Validações de dados no *userController.ts*

```
if (!name || !email || !password) {
  return res.status(400).send({ error: 'Campos obrigatórios faltando' });
}

try {
  const existingUser = await userModel.findByEmail(email);
  if (existingUser) {
    return res.status(400).send({ message: 'Usuário já registrado com este email.' });
  }
}
```

Fonte: Elaborado pelo autor (2024)

Antes da senha ser armazenada no banco, é executada a criptografia da senha com a biblioteca *bcrypt*, mantendo a segurança dos dados do usuário, para fazer isso, foi criada uma função *hashPassword*, no arquivo *hashUtils.ts* na pasta */utils* como apresentado na **Figura 8**:

**Figura 8** - Função de criptografia utilizando a biblioteca *bcrypt*

```
export const hashPassword = async (password: string): Promise<string> => {
  const saltRounds = 10;
  return await bcrypt.hash(password, saltRounds);
};
```

Fonte: Elaborado pelo autor (2024)

Então finalmente fazemos a inserção dos dados do usuário registrado, com uma consulta SQL com estrutura semelhante à da **Figura 5**.

- *Login* de Usuários:

Inicialmente, é verificado a existência do e-mail no banco de dados, para posteriormente validar se a senha (que é criptografada novamente para realizar a validação) é a correta.

Garantindo que essa tentativa de *login* foi bem-sucedida, geramos então um *token JWT* contendo o *id* do usuário e se ele é administrador ou não com base no campo *is\_admin*, como exibido na **Figura 9**.

**Figura 9** - Login do usuário no *UserController.ts*

```

try {
  const user = await userModel.findByEmail(email);
  if (!user) {
    return res.status(401).send({ error: 'Email não encontrado ou senha inválida' });
  }

  const isPasswordCorrect = await comparePassword(password, user.password);
  if (!isPasswordCorrect) {
    return res.status(401).send({ error: 'Email não encontrado ou senha inválida' });
  }

  const token = await res.jwtSign({ id: user.id, is_admin: user.is_admin });

  const { password: _, ...userWithoutPassword } = user;
  return res.status(200).send({ token, user: userWithoutPassword });
} catch (error) {
  req.log.error(error);
  return res.status(500).send({ error: 'Erro interno do servidor' });
}

```

Fonte: Elaborado pelo autor (2024)

Vale destacar também que o retorno em caso de senha errada ou e-mail não encontrado é o mesmo, para garantir a segurança da conta dos usuários.

- Proteção de rotas

Em rotas que precisam de controle de acesso baseado no nível de permissão (egresso ou administrador), existe o *middleware* por parte do *Fastify* que permite a checagem dessa informação a partir do *token* enviado no parâmetro *Authorization* da requisição HTTP/HTTPS. Essa informação é facilmente checada nas funções dos controladores do sistema, como exemplificado na **Figura 10**:

**Figura 10** - Exemplo de validação de permissão

```

const user = req.user as ReqUserType;

if (!user?.is_admin) {
  return res.status(403).send({ message: 'Acesso negado.' });
}

```

Fonte: Elaborado pelo autor (2024)

- Recuperação de senha

A recuperação de senha é feita inicialmente por um pedido no *front-end*, apenas passando o e-mail do usuário. Então um e-mail é enviado usando a biblioteca *nodemailer*, contendo um token gerado pela biblioteca *crypto*, que expira em uma hora após o pedido de recuperação de senha.

Após o usuário preencher no *front-end* o formulário com a nova senha, fazemos a atualização dela no banco após criptografá-la do mesmo jeito que é feito no registro de novo usuário.

#### **3.4.3.2 Gerenciamento de formulários com dependências**

O gerenciamento de formulários é uma funcionalidade essencial do sistema, permitindo que os administradores criem, editem e publiquem formulários personalizados para os egressos. Os formulários podem conter questões de diferentes tipos e incluir dependências entre as perguntas.

##### **Principais Funcionalidades:**

- **Criação e Edição de Formulários:**
  - Os administradores podem adicionar ou remover questões.
  - É possível definir dependências entre as questões, de forma que uma pergunta seja exibida apenas se uma resposta específica for selecionada em uma questão anterior.
- **Publicação de Formulários:**
  - Após a conclusão, os formulários podem ser publicados, ficando disponíveis para que os egressos os respondam.
- **Visualização de Respostas:**
  - Os administradores podem visualizar as respostas dos egressos, acessando dados de forma individual ou agregada.
- **Análise de Dados:**
  - O sistema permite gerar estatísticas e relatórios baseados nas respostas coletadas, possibilitando o agrupamento por etnia, gênero ou ano de graduação.

##### **Implementação Técnica:**

- As funcionalidades relacionadas aos formulários foram implementadas seguindo o padrão MVC descrito anteriormente.
- As rotas correspondentes estão protegidas por autenticação, garantindo que apenas administradores autorizados possam acessar essas funcionalidades.

- A lógica de dependência entre questões é gerenciada no *back-end*, armazenando as informações na tabela *field\_dependency* e fornecendo ao frontend os dados necessários para renderizar o formulário corretamente.

#### **3.4.3.3 Outras funcionalidades**

- **Criação de Novos Administradores:**
  - Os administradores existentes têm a capacidade de criar administradores.
  - Validações são realizadas para garantir que apenas usuários com permissão possam realizar essa ação.
- **Coleta e Armazenamento de Respostas:**
  - Os egressos podem responder aos formulários disponíveis.
  - As respostas são armazenadas no banco de dados, associadas ao usuário e ao formulário correspondente.
  - Validações são realizadas para assegurar a consistência e integridade dos dados coletados.
- **Gerenciamento de Perfil do Usuário:**
  - Os usuários podem visualizar e editar suas informações pessoais.
  - É possível atualizar a senha, sendo necessário fornecer a senha atual para confirmação.
  - Validações garantem que a senha antiga informada seja correta antes de permitir a alteração.

As funcionalidades desenvolvidas no backend proporcionam uma base importante para o frontend. A implementação detalhada da autenticação e autorização garante a segurança e integridade dos dados, enquanto as demais funcionalidades, como o gerenciamento de formulários e análise de respostas, vão permitir que a instituição obtenha informações valiosas sobre os egressos. A combinação dessas funcionalidades atende plenamente aos requisitos definidos para o sistema, demonstrando a eficácia das tecnologias e abordagens utilizadas no desenvolvimento.



### 3.5 Desenvolvimento do *frontend*

O *front-end* do sistema de egressos foi desenvolvido utilizando *React* com o *Vite*, uma ferramenta de *build* moderna que proporciona um ambiente de desenvolvimento rápido e eficiente. Foi escolhido utilizar *TypeScript* para garantir maior segurança no código e facilitar a manutenção, além de melhorar a produtividade durante o desenvolvimento.

#### 3.5.1 Escolha das tecnologias

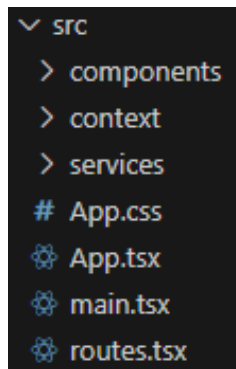
Para o desenvolvimento do *front-end*, foram escolhidas as seguintes tecnologias e bibliotecas:

- ***React***: Biblioteca popular e robusta para construção de interfaces de usuário, que permitiu o desenvolvimento de componentes reutilizáveis.
- ***Vite***: Foi escolhido como ferramenta de build em vez da biblioteca *create-react-app* devido à sua performance superior, especialmente em projetos com múltiplos componentes.
- ***TypeScript***: O uso do TypeScript, além de proporcionar tipagem estática, ajudou a evitar erros comuns durante o desenvolvimento, especialmente em um projeto com múltiplas funcionalidades interdependentes como o sistema de egressos.
- ***react-router-dom***: Biblioteca utilizada para gerenciar o roteamento da aplicação, permitindo a navegação entre diferentes páginas e protegendo rotas que necessitam de autenticação.
- ***axios***: Utilizado para realizar requisições *HTTP* ao *back-end*, facilitando a integração entre *front-end* e *back-end*.
- ***recharts***: Biblioteca usada para criação de gráficos personalizados, com bastante liberdade para alterações.
- ***react-beautiful-dnd***: Essa biblioteca foi necessária para o reordenamento de questões implementado no componente de edição de formulário.

#### 3.5.2 Estrutura do *frontend*

O *front-end* foi estruturado de forma modular, visando a separação de responsabilidades e a reutilização de componentes.

**Figura 11** - Estrutura de pastas do *front-end*



Fonte: Elaborado pelo autor (2024)

A estrutura principal do projeto segue o padrão demonstrado acima na **Figura 11**, em que temos as seguintes pastas:

- **components/**: Contém todos os componentes do sistema, como botões, formulários, *header*, *footer* etc.
- **contexts/**: Armazena os contextos globais usados na aplicação, como autenticação e seleção de cursos.
- **services/**: Contém funções para realizar chamadas à API e lidar com a comunicação entre *front-end* e *back-end*.

### 3.5.3 Implementação das funcionalidades principais

#### 3.5.3.1 Roteamento

O roteamento é feito usando a biblioteca *react-router-dom*, em que usamos um arquivo único para determinar as rotas do sistema, com o componente *React Routes* determinando a lista de rotas do sistema, e o componente *Route* determinando uma rota específica, conforme apresentado na **Figura 12**.

#### 3.5.3.2 Fluxo de login e registro:

- No registro, o usuário fornece as informações necessárias, como nome, e-mail, senha, e no caso de egressos, dados adicionais (data de nascimento, ano de graduação, gênero, etnia).

Após o login, o token JWT é utilizado para autenticar as requisições subsequentes.

**Figura 12** - Roteamento do sistema no arquivo *routes.tsx*

```
export const routes = () => {
  return (
    <Routes>
      <Route path="/" element={<Homepage />} />
      <Route path="/register" element={<Register />} />
      <Route path="/login" element={<Login />} />
      <Route path="/password-recovery" element={<PasswordRecovery />} />
      <Route path="/reset-password" element={<PasswordReset />} />
      <Route path="/admin/*" element={<AdminLayout/>>
        <Route path="dashboard" element={<Dashboard/>} />
        <Route path="data" element={<AdminData />} />
        <Route path="forms" element={<AdminFormPage />} />
        <Route path="forms/:formId" element={<AdminFormEditPage/>} />
        <Route path="forms/:formId/responses" element={<FormResponsesPage />} />
        <Route path="forms/:formId/responses/:userId" element={<UserAnswersPage />} />
        <Route path="register-admin" element={<AdminRegister />} />
      </Route>
      <Route path="/profile" element={<Profile />} />
      <Route path="/alumni" element={<Alumni />} />
      <Route path="/form/:id" element={<AlumniForm />} />
      <Route path="*" element={<Error name="Página não encontrada" />} />
    </Routes>
  );
}
```

Fonte: Elaborado pelo autor (2024)

### 3.5.3.3 Autenticação de usuários

A autenticação foi implementada utilizando o contexto de usuário (*userContext* na pasta *context*). Quando um usuário faz login com sucesso, o *token JWT* retornado pelo *backend* é armazenado no *localStorage* (o armazenamento interno do navegador, que não é perdido ao fechá-lo), permitindo que o usuário permaneça autenticado entre as sessões.

**Figura 13** - *UserProvider* do *UserContext.tsx*

```
export const UserProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const [user, setUser] = useState<UserData | null>(null);

  useEffect(() => {
    const token = localStorage.getItem('token');

    if (token) {
      getUserData()
        .then(userData => {
          setUser(userData);
        })
        .catch(() => {
          localStorage.removeItem('token');
        });
    }
  }, []);

  return (
    <UserContext.Provider value={{ user, setUser }}>
      {children}
    </UserContext.Provider>
  );
};
```

Fonte: Elaborado pelo autor (2024)

Na **Figura 13** acima é possível observar como o *UserProvider* é estruturado, contendo um *hook* do *React* que checa se o *token* está no armazenamento interno do navegador, caso esteja, ele pega os dados do usuário com a função *getUserData()*, que está no arquivo *userService*, onde, como mencionado na seção 3.5.2, ficam as requisições para o backend.

**Figura 14** - *getUserData* do *userService.ts*

```
export const getUserData = async () => {
  const token = localStorage.getItem('token');

  try {
    const response = await axios.get(`${API_URL}/users/me`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    });
    const userData = response.data

    userData.birthdate = userData.birthdate ? new Date(userData.birthdate).toLocaleDateString() : ''
    userData.gender = genderTranslations[userData.gender]
    userData.ethnicity = ethnicityTranslations[userData.ethnicity]

    return userData;
  } catch (error) {
    console.error('Erro ao buscar dados do usuário:', error);
    throw error;
  }
}
```

Fonte: Elaborado pelo autor (2024)

Como observado na **Figura 14**, as requisições para o *back-end* são feitas usando a biblioteca *axios*, conforme seção 3.5.1, usando o retorno da requisição para popular o contexto

de usuário com os dados do usuário, ou seja, qualquer componente do sistema, que importe o `AuthProvider`, é capaz de acessar os dados do usuário que está usando o sistema no momento.

#### 3.5.3.4 Dashboard do egresso e administrador

- **Dashboard do Egresso:** Exibe formulários disponíveis para preenchimento e permite que o egresso visualize suas respostas anteriores.
  - Composto pelos componentes ***Alumni***, que lista os formulários pendentes e preenchidos e ***AlumniForm*** que exibe o formulário para preenchimento do aluno.
- **Dashboard do Administrador:** Permite gerenciar formulários, visualizar estatísticas e acessar respostas de alunos específicos.
  - Envolve diversos componentes, entre eles:
    - ***Dashboard***: Este componente é o principal da aba de Resumo do administrador.
    - ***AdminData***: É a aba de dados, que tem os subcomponentes ***Chart*** (Gráfico gerado com os dados retornados pelo *backend*) e ***YearRangeSelector*** (seletor de anos na parte de filtros), é possível gerar gráficos de barra, de setores e tabela, e baixar em vários formatos de arquivo, como .PDF, .PNG e .CSV.
    - ***AdminFormPage***: Lista de formulários de um curso (com base no curso atual no *courseContext*), onde é possível acessar os dados de um formulário preenchido com os componentes ***FormResponsesPage*** e ***UserAnswersPage***.
    - ***AdminFormEditPage***: Página de edição de formulário, contém os subcomponentes ***AddFieldModal***, ***EditFieldModal***, ***FieldItem***, ***RenameFormModal***

Existem também vários componentes reutilizados em múltiplas páginas, por exemplo: ***Button*** (botão), ***Form***(formulário), ***Tooltip***(uma pequena caixa de ajuda, explicando um contexto/botão) e ***Modal***(janela que bloqueia e escurece o resto da tela, limitando o usuário para apenas o conteúdo dentro dela) e outros importantes que são carregados antes da própria página, como ***Header*** (cabeçalho da página) e ***Footer*** (rodapé da página).

**Figura 15** – Checagem de dependências do componente *Form*

```
const checkDependencies = (fieldName: string) => {
  const field = fields.find(f => f.name === fieldName);
  if (!field || (!field.dependencies || field.dependencies.length === 0)) return true;
  return field.dependencies.some(dep => {
    const dependentValue = formData[dep.fieldId];
    if (Array.isArray(dependentValue)) {
      return dep.optionIds.some(optionId => dependentValue.includes(String(optionId)));
    }
    return dep.optionIds.includes(Number(dependentValue));
  });
};
```

Fonte: Elaborado pelo autor (2024)

Na **Figura 15** é possível observar como é realizada a checagem de dependências no componente de formulário, validando se a dependência é cumprida antes de renderizar ele para o preenchimento do egresso. Apesar de ser usado no formulário que o ex-aluno preenche, também é usado em fluxos como registro e *login*.

**Figura 16** – Validação de campos na criação de questão

```
const validate = () => {
  let validationErrors: { [key: string]: string } = {};

  if (!question.trim()) {
    validationErrors.question = 'O texto da questão é obrigatório.';
  }

  if ([FieldType.SINGLE_CHOICE, FieldType.MULTIPLE_CHOICE].includes(type)) {
    if (!indicator.trim()) {
      validationErrors.indicator = 'O indicador é obrigatório para questões de escolha única ou múltipla.';
    }

    if (options.filter(opt => opt.text.trim()).length < 2) {
      validationErrors.options = 'É necessário adicionar pelo menos duas opções.';
    }

    options.forEach((option, idx) => {
      if (!option.text.trim()) {
        validationErrors[`option_${idx}`] = `A opção ${idx + 1} não pode estar vazia.`;
      }
    });
  }

  setErrors(validationErrors);
  return Object.keys(validationErrors).length === 0;
};
```

Fonte: Elaborado pelo autor (2024)

Como demonstrado na **Figura 16**, existe a validação da entrada do administrador em componentes como o da criação de questão (*AddFieldModal*), fazendo com que não sejam enviados dados errados ou faltantes para o *back-end*, reduzindo as chances de bugs e problemas relacionados aos dados da requisição.

As funcionalidades desenvolvidas no *front-end* proporcionam uma interface intuitiva e eficiente para os usuários do sistema de egressos. A utilização de tecnologias modernas como **React** e **TypeScript** permitiram a criação de uma aplicação *web* dinâmica e responsiva, facilitando a interação tanto para egressos quanto para administradores. A implementação de componentes reutilizáveis, a gestão de estado através de contextos e a integração eficaz com o *back-end* asseguraram uma experiência de usuário fluida e segura.

A autenticação de usuários, o roteamento protegido e a persistência de sessão garantem que apenas usuários autorizados acessem as funcionalidades apropriadas, mantendo a segurança e a integridade dos dados. Funcionalidades como o gerenciamento de formulários, incluindo a criação de questões com dependências e a validação de entradas, atendem plenamente aos requisitos definidos para o sistema. Além disso, a capacidade de visualizar e analisar dados através de gráficos interativos enriquece a experiência do administrador, permitindo obter insights valiosos sobre os egressos.

Dessa forma, o *front-end* desenvolvido não só complementa as funcionalidades do *back-end*, mas também contribui significativamente para a usabilidade e acessibilidade do sistema. A combinação das tecnologias e abordagens utilizadas no desenvolvimento do *front-end* demonstra eficácia e alinhamento com as necessidades do projeto, assegurando que o sistema de egressos cumpra seus objetivos de maneira eficiente e satisfatória.

## 4 RESULTADOS

Nesta seção, serão apresentados os resultados obtidos com o desenvolvimento do sistema de egressos. Serão demonstrados os fluxos de uso principais tanto para o usuário egresso quanto para o administrador, através de capturas de tela que ilustram as funcionalidades implementadas.

### 4.1 Fluxos de uso do sistema

Os fluxos de uso do sistema foram organizados de acordo com os perfis de usuário:

- Usuário Egresso
- Usuário Administrador

#### 4.1.1 Fluxos do usuário egresso

Abaixo serão apresentadas as duas principais funcionalidades envolvendo o usuário egresso.

##### 4.1.1.1 Tela de login e registro

**Figura 17 - Tela de Login**

Sistema de Egressos

INSTITUTO FEDERAL  
São Paulo  
Câmpus Campinas

Login Registrar

**Login**

Email

Senha

Enviar

[Esqueceu sua senha?](#)

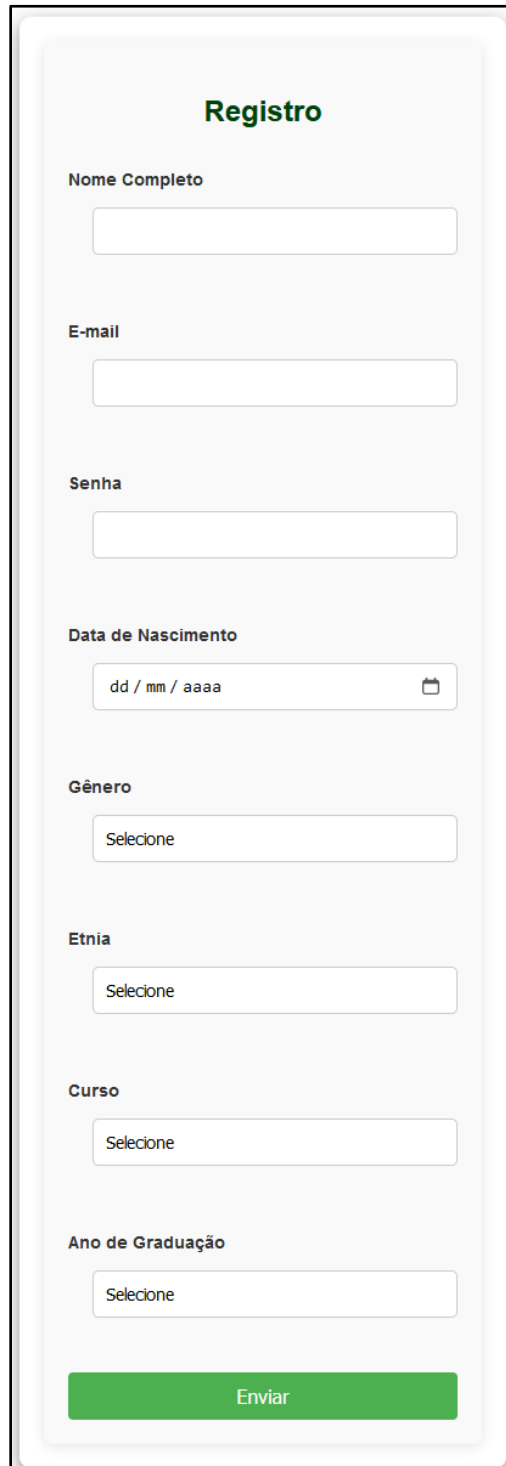
Desenvolvido por Pedro Henrique Pereira Soares - 2024



Fonte: Elaborado pelo autor (2024)

A tela de *login* (**Figura 17**) permite que o egresso acesse o sistema inserindo seu e-mail e senha cadastrados. Caso não possua uma conta, o usuário pode optar por se registrar.

**Figura 18** - Tela de Registro



O formulário de registro, intitulado "Registro", contém os seguintes campos:

- Nome Completo:** Campo de texto para o nome completo.
- E-mail:** Campo de texto para o endereço de e-mail.
- Senha:** Campo de texto para a senha.
- Data de Nascimento:** Campo de data com máscara "dd / mm / aaaa" e ícone de calendário.
- Gênero:** Campo de seleção com o texto "Selecione".
- Etnia:** Campo de seleção com o texto "Selecione".
- Curso:** Campo de seleção com o texto "Selecione".
- Ano de Graduação:** Campo de seleção com o texto "Selecione".

Um botão verde com o texto "Enviar" está localizado na base do formulário.

Fonte: Elaborado pelo autor (2024)

Na tela de registro (**Figura 18**), o egresso pode criar uma conta fornecendo informações como nome, e-mail, senha, data de nascimento, ano de graduação, gênero e etnia. Validações são realizadas para garantir a integridade dos dados inseridos.

#### **4.1.1.2 Preenchimento de formulários**

**Figura 19** - Dashboard do Egresso



Fonte: Elaborado pelo autor (2024)

Após o login, o egresso é direcionado ao seu dashboard (**Figura 19**), onde pode visualizar os formulários disponíveis para preenchimento e acompanhar suas respostas anteriores.

**Figura 20** - Preenchimento do Formulário

**Sistema de Egressos**

Olá, Pedro Henrique Pereira Soares!

INSTITUTO FEDERAL  
São Paulo  
Câmpus Campinas

Início Perfil Logout

**Formulário de Egressos 2024**

Quando você entrou no curso de TADS do IFSP, você trabalhava?

Você trabalhava em?

Você considera que a sua formação em Análise e Desenvolvimento de Sistemas no IFSP foi relevante para entrar no mercado de trabalho?

**Enviar**

Desenvolvido por Pedro Henrique Pereira Soares - 2024

Fonte: Elaborado pelo autor (2024)

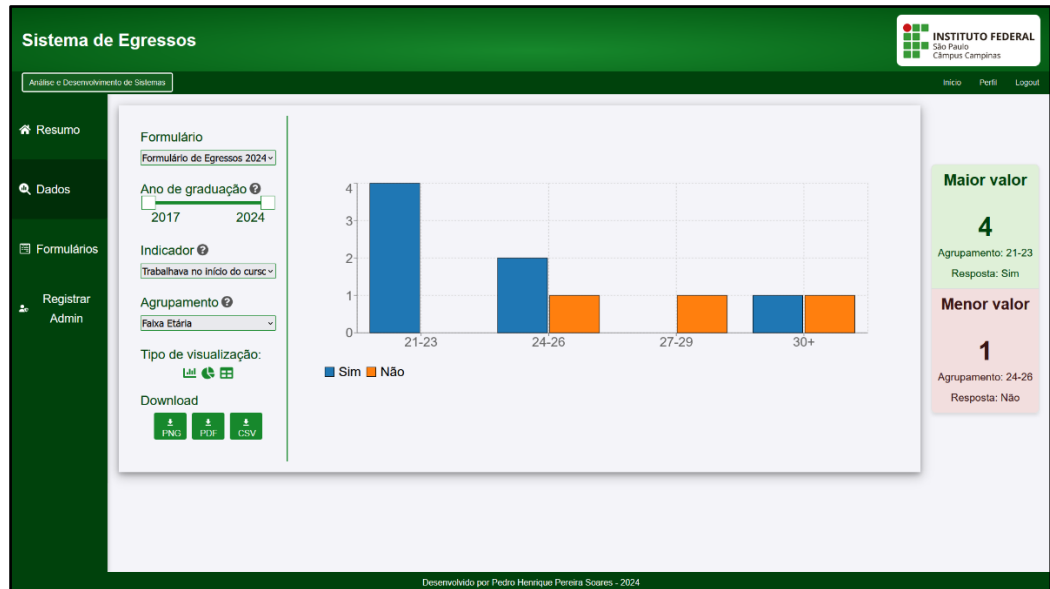
Ao selecionar um formulário, o egresso é apresentado às questões (**Figura 20**). As perguntas são exibidas de acordo com as dependências definidas, ou seja, algumas questões aparecem ou não com base nas respostas anteriores. Isso torna o formulário dinâmico e personalizado.

#### ***4.1.2 Fluxos do usuário administrador***

Abaixo serão exibidas alguns dos fluxos principais que o usuário Administrador pode realizar.

#### 4.1.2.1 Página de análise de dados

**Figura 21** - Página de Análise de Dados

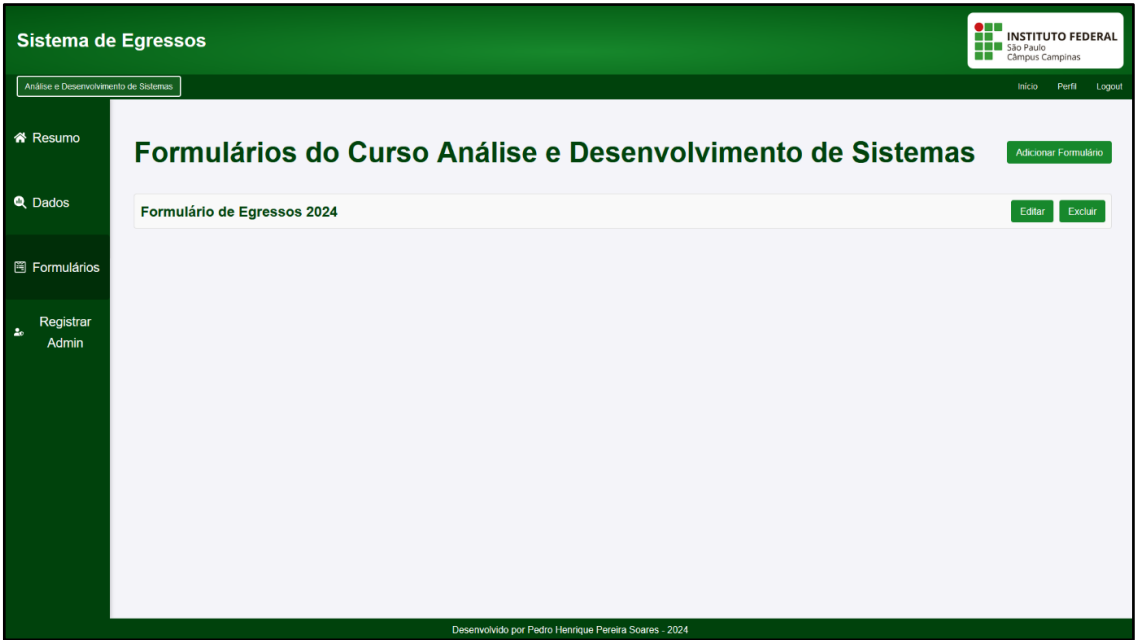


Fonte: Elaborado pelo autor (2024)

A página de análise de dados (Figura 21) permite que o administrador visualize informações coletadas dos egressos através de gráficos de barras, de setores e tabelas. Selecionando o indicador (determinado na criação de uma questão do tipo escolha única ou múltipla escolha), é possível aplicar filtro por ano de graduação, agrupar por gênero, etnia e faixa etária. Os gráficos podem ser exportados em diversos formatos, como PDF, PNG e CSV.

4.1.2.2 Página de criação e edição de formulários

Figura 22 - Página de Formulários



Fonte: Elaborado pelo autor (2024)

Na página de formulários (**Figura 22**), o administrador pode:

- Criar formulários, definindo o título.

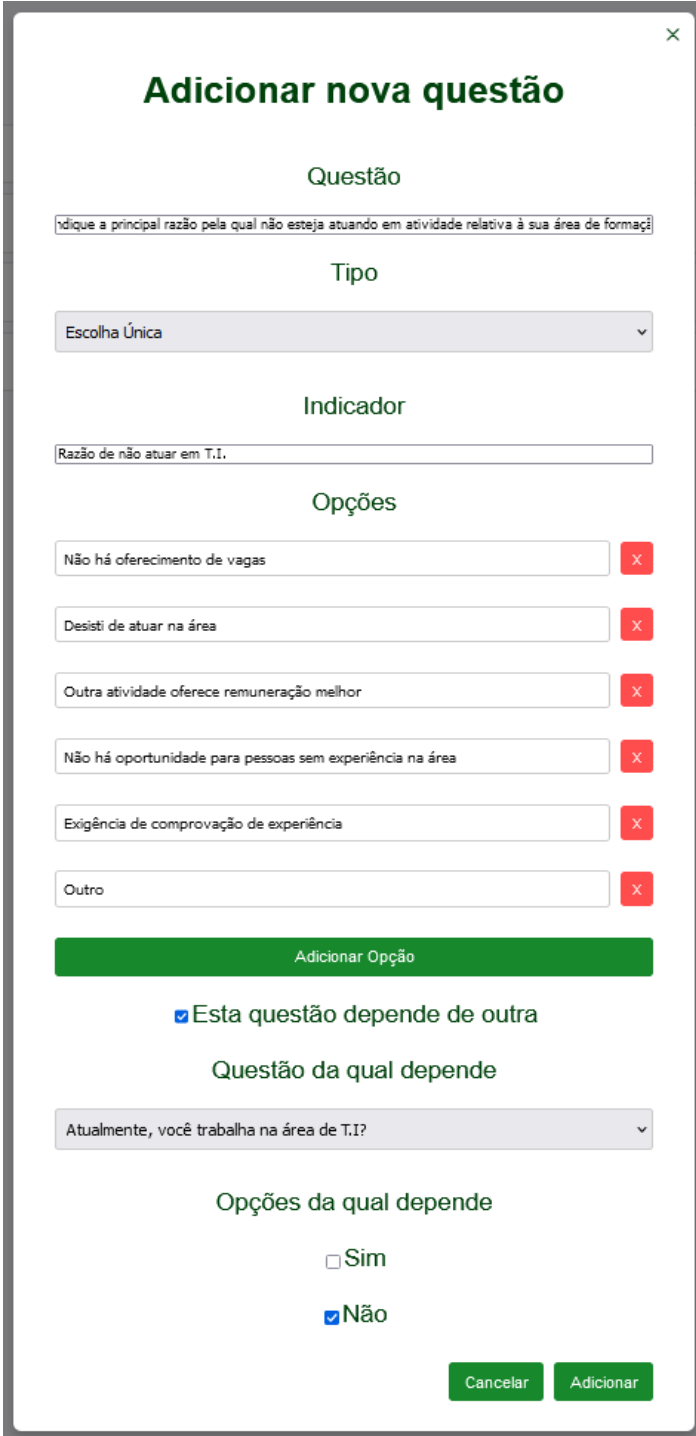
Figura 23 - Página de Edição de Formulários



Fonte: Elaborado pelo autor (2024)

Na página de edição de formulários (**Figura 23**), o administrador pode:

**Figura 24** - Modal de adição de questão



Modal de adição de questão. O formulário contém os seguintes campos e opções:

- Adicionar nova questão** (título)
- Questão** (campo de texto): Indique a principal razão pela qual não esteja atuando em atividade relativa à sua área de formação.
- Tipo** (dropdown menu): Escolha Única.
- Indicador** (campo de texto): Razão de não atuar em T.I.
- Opções** (lista de opções com botões de exclusão):
  - Não há oferecimento de vagas
  - Desisti de atuar na área
  - Outra atividade oferece remuneração melhor
  - Não há oportunidade para pessoas sem experiência na área
  - Exigência de comprovação de experiência
  - Outro
- Adicionar Opção** (botão verde)
- ☒ Esta questão depende de outra
- Questão da qual depende** (dropdown menu): Atualmente, você trabalha na área de T.I?
- Opções da qual depende** (checkboxes):
  - ☐ Sim
  - ☒ Não
- Cancelar** e **Adicionar** (botões verdes)

Fonte: Elaborado pelo autor (2024)

- Adicionar (**Figura 24**), editar ou remover questões
  - A remoção de questões é realizada ao clicar no botão de lixeira no item da questão, caso ela tenha dependência, o sistema não permite a deleção e pede para o usuário remover a dependência antes.
- Estabelecer dependências entre questões, permitindo que a exibição de uma pergunta dependa da resposta a uma questão anterior.
- Reordenar questões utilizando a funcionalidade de arrastar e soltar (*drag and drop*) fornecida pela biblioteca *react-beautiful-dnd*.

Os fluxos de uso apresentados demonstram como o sistema de egressos permite uma interação eficiente e intuitiva tanto para os ex-alunos quanto para os administradores. Através das telas de login, registro e preenchimento de formulários, os egressos podem fornecer informações valiosas de forma simples. Por outro lado, os administradores dispõem de ferramentas para criar e gerenciar formulários personalizados, além de analisar os dados coletados, facilitando a tomada de decisões e o aprimoramento do processo educativo. Assim, os resultados obtidos evidenciam que o sistema atende aos objetivos propostos, contribuindo significativamente para a política de acompanhamento de egressos da instituição.

## 5 CONCLUSÕES

O presente trabalho teve como objetivo desenvolver um sistema de egressos para o curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de São Paulo (IFSP) – Campus Campinas, visando coletar informações relevantes dos ex-alunos por meio de formulários, armazená-las em um banco de dados e disponibilizá-las para análise através de gráficos e tabelas. Essa iniciativa surgiu da necessidade identificada na instituição de atender às diretrizes estabelecidas pelo Sistema Nacional de Avaliação da Educação Superior (Sinaes), em especial no que tange à política de acompanhamento do egresso.

A partir da pesquisa sobre a política de acompanhamento de egressos do IFSP e das diretrizes do Sinaes, verificou-se a importância de criar mecanismos que permitissem não apenas a coleta de dados dos ex-alunos, mas também a utilização dessas informações para o aprimoramento do processo formativo e a adequação da formação profissional às demandas do mercado de trabalho. Com base nessa necessidade, o desenvolvimento do sistema foi planejado para atender aos critérios estabelecidos, contribuindo para que a instituição alcance conceitos elevados nas avaliações do Ministério da Educação.

O sistema foi desenvolvido seguindo uma arquitetura composta por banco de dados, *back-end* e *front-end*, integrados de forma coesa para proporcionar uma experiência eficiente e segura aos usuários.

- Banco de Dados: Foi modelado para armazenar de forma eficiente as informações dos usuários, formulários e respostas. A estrutura relacional, implementada em *MySQL*, garantiu a integridade referencial e a segurança dos dados. Foram adotadas boas práticas de normalização e segurança, assegurando a confidencialidade das informações conforme a Lei Geral de Proteção de Dados (LGPD).
- *Back-end*: Desenvolvido em *Node.js* com o *framework Fastify* e escrito em *TypeScript*, o *back-end* implementou as funcionalidades principais do sistema. Dentre elas, destacam-se:
  - Autenticação e Autorização: Implementadas utilizando *JSON Web Tokens* (JWT), garantiram a segurança no acesso ao sistema. A criptografia de senhas com *bcrypt* e a gestão de *tokens* asseguraram a proteção dos dados dos usuários.
  - Gerenciamento de Usuários: Permitiu o registro de egressos e administradores, com validações e controles de acesso adequados.
  - Gerenciamento de Formulários: Administradores podem criar e editar formulários, incluindo questões com dependências, proporcionando flexibilidade na coleta de informações.



- Armazenamento de Respostas: As respostas dos egressos são armazenadas de forma estruturada, permitindo análises posteriores.
- *Front-end*: Construído com *React* e *TypeScript*, o *frontend* proporcionou uma interface amigável e intuitiva para os usuários. Foram implementadas funcionalidades como:
  - Fluxo de Login e Registro: Com validações e *feedback* ao usuário, assegurando usabilidade e segurança.
  - Dashboard do Egresso e Administrador: Interfaces personalizadas que permitem aos usuários acessarem as funcionalidades relevantes de acordo com seu perfil.
  - Preenchimento de Formulários: Com questões dinâmicas e tratamento de dependências entre perguntas, facilitando a interação do egresso com o sistema.
  - Visualização e Análise de Dados: Administradores podem visualizar gráficos e tabelas gerados a partir das respostas, possibilitando a extração de insights valiosos.

A integração entre o banco de dados, *backend* e *frontend* foi realizada de forma eficiente, garantindo a comunicação entre as camadas e o funcionamento adequado do sistema. As tecnologias escolhidas se mostraram adequadas ao projeto, proporcionando um desenvolvimento ágil e eficaz. A utilização de *TypeScript* tanto no backend quanto no frontend aumentou a segurança e a qualidade do código, reduzindo erros comuns e facilitando a manutenção.

Com o sistema desenvolvido, os objetivos propostos foram alcançados:

- Atendimento à Política de Acompanhamento do Egresso: O sistema permite a criação de uma base de dados atualizada dos egressos, promove um relacionamento contínuo entre a instituição e seus ex-alunos e fornece mecanismos para avaliar a adequação da formação profissional ao mercado de trabalho.
- Utilização das Opiniões dos Egressos: As informações coletadas podem ser utilizadas para aperfeiçoar o processo de formação, permitindo que a instituição identifique pontos fortes e áreas que necessitam de melhorias.
- Contribuição para a Qualidade da Educação: Ao proporcionar ferramentas para a análise dos dados coletados, o sistema auxilia na tomada de decisões estratégicas que visam elevar a qualidade do ensino oferecido pelo curso de TADS do IFSP – Campus Campinas.

Durante o desenvolvimento, foram enfrentados desafios, como a impossibilidade de integrar a autenticação com o SUAP via LDAP devido a restrições institucionais. Esse

obstáculo foi superado com a implementação de uma autenticação própria utilizando JWT, o que demonstrou adaptabilidade e capacidade de solucionar problemas.

Como trabalhos futuros, sugere-se:

- **Expansão do Sistema:** Adaptar o sistema para atender outros cursos do campus, ampliando o alcance e os benefícios proporcionados pela ferramenta.
- **Integração com Sistemas Institucionais:** Caso as restrições atuais sejam superadas, integrar o sistema com o SUAP ou outros sistemas institucionais para facilitar ainda mais o acesso e a gestão dos usuários.
- **Melhorias na Usabilidade e Funcionalidades:** A partir de testes com usuários reais e coleta de feedback, aprimorar a interface e adicionar novas funcionalidades que atendam às necessidades da instituição e dos egressos.
- **Análises Avançadas de Dados:** Implementar técnicas de inteligência artificial ou aprendizado de máquina para extrair insights mais profundos dos dados coletados.

Conclui-se que o desenvolvimento do sistema de egressos alcançou os objetivos propostos, oferecendo uma ferramenta eficaz para o acompanhamento dos ex-alunos, contribuindo para o aprimoramento da qualidade do ensino e atendendo às diretrizes do Ministério da Educação. O sistema desenvolvido não apenas supre uma necessidade institucional, mas também representa um avanço significativo na gestão educacional, reforçando o compromisso do IFSP – Campus Campinas com a excelência acadêmica e o desenvolvimento contínuo. Além de atingir os objetivos propostos, o desenvolvimento deste sistema também proporcionou ao autor uma oportunidade valiosa de aprimorar suas habilidades na área de desenvolvimento de aplicações web, consolidando conhecimentos prévios e adquirindo novas competências técnicas que ampliaram sua experiência e confiança nesse campo.

## REFERÊNCIAS

AGGARWAL, S. et al. Modern web-development using reactjs. **International Journal of Recent Research Aspects**, v. 5, n. 1, p. 133-137, 2018. Disponível em: <https://www.semanticscholar.org/paper/133-Modern-Web-Development-using-ReactJS-Aggarwal/f3f0067ab28222e80f36d3bd656921dd44fd4598>. Acesso em: 18 set. 2022.

BRASIL. MINISTÉRIO DA EDUCAÇÃO. **Avaliação externa de instituições de educação superior: diretrizes e instrumento**. 2006. Disponível em: <https://www.gov.br/inep/pt-br/centrais-de-conteudo/acervo-linha-editorial/publicacoes-institucionais/avaliacoes-e-exames-da-educacao-superior/avaliacao-externa-das-instituicoes-de-educacao-superior-2013-diretrizes-e-instrumento>. Acesso em: 14 set. 2022.

BRASIL. MINISTÉRIO DA EDUCAÇÃO. INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA (INEP). **Roteiro de Auto-Avaliação Institucional: orientações gerais**. Brasília, INEP, 2004. Disponível em: <https://www.gov.br/inep/pt-br/centrais-de-conteudo/acervo-linha-editorial/publicacoes-institucionais/avaliacoes-e-exames-da-educacao-superior/roteiro-de-auto-avaliacao-institucional-2013-orientacoes-gerais>. Acesso em: 15 set. 2022.

BRASIL. MINISTÉRIO DA EDUCAÇÃO. INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA (INEP). **Sistema Nacional de Avaliação da Educação Superior (SINAES): da concepção à regulamentação**. 2009. Disponível em: <https://www.gov.br/inep/pt-br/centrais-de-conteudo/acervo-linha-editorial/publicacoes-institucionais/avaliacoes-e-exames-da-educacao-superior/sinaes-2013-da-concepcao-a-regulamentacao-2013-5a-edicao-revisada-e-ampliada>. Acesso em: 16 set. 2022.

CANTELON, M. **Node.js in action**. Shelter Island, Ny: Manning Publications, 2014.  
DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro. Elsevier Brasil, 2004.

DUBOIS, P. **MySQL**. [s.l.] Pearson Education, 2008.

EGRESSO. In: DICIO. Disponível em: <https://www.dicio.com.br/egresso/>. Acesso em: 22 de out. 2022.

ESPARTEL, L. B. O uso da opinião dos egressos como ferramenta de avaliação de cursos: o caso de uma instituição de ensino superior catarinense. **Revista Alcance**, v. 16, n. 1, p. 102-114, 2009.

FLANAGAN, D. **JavaScript o guia definitivo**. Porto Alegre: Bookman, 2004.

MASSE, M. **REST API Design Rulebook**. [s.l.] “O’Reilly Media, Inc.”, 2011.

MELTON, J. Sql language summary. **Acm Computing Surveys (CSUR)**, v. 28, n. 1, p. 141-143, 1996. Disponível em: <https://dl.acm.org/doi/10.1145/234313.234374>. Acesso em: 20 set. 2022.

MICHELAN, L. S. et al. Gestão de egressos em instituições de ensino superior: possibilidades e potencialidades. **Colóquio internacional sobre gestão universitária na américa do sul**, v. 9, p. 1-16, 2009. Disponível em: <https://repositorio.ufsc.br/handle/123456789/36720>. Acesso em: 13 set. 2022.

MICROSOFT. **Typescript: the starting point for learning TypeScript**. Disponível em: <https://www.typescriptlang.org/docs/>. Acesso em: 20 out. 2024.

PAUL, J.-J. ACOMPANHAMENTO DE EGRESSOS DO ENSINO SUPERIOR: experiência brasileira e internacional. **Caderno CRH**, v. 28, p. 309–326, 2015. Disponível em: <https://www.scielo.br/j/ccrh/a/TjHy6zTq5LzMMjLkHJg7JRc/>. Acesso em: 14 set. 2022.

SCHONS, L. **O que é MVC? — Conceitos Básicos**. Medium. Disponível em: <https://medium.com/@sschonss/o-que-%C3%A9-mvc-conceitos-b%C3%A1sicos-6363f9662f8c>. Acesso em: 22 out. 2024.

SILVA, J. M. da; BEZERRA, R. O. Sistema de acompanhamento dos egressos aplicado na Universidade Federal de Santa Catarina. **Revista Gestão Universitária na América Latina-GUAL**, v. 8, n. 3, p. 1-15, 2015. Disponível em: <https://www.redalyc.org/articulo.oa?id=319342694016>. Acesso em: 23 out. 2022.