

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Vitória Silva Cardoso

**Implementação de um Backend e Integração
com Frontend para Acompanhamento de
Egressos da UFU**

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Vitória Silva Cardoso

**Implementação de um Backend e Integração com
Frontend para Acompanhamento de Egressos da UFU**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, como parte dos requi-
sitos exigidos para a obtenção título de Ba-
charel em Sistemas de Informação.

Orientador: Maria Adriana Vidigal de Lima

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2025

Vitória Silva Cardoso

Implementação de um Backend e Integração com Frontend para Acompanhamento de Egressos da UFU

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 06 de Maio de 2025:

Maria Adriana Vidigal de Lima
Orientadora

Rafael Dias Araújo
Membro da Banca

Renato de Aquino Lopes
Membro da Banca

Uberlândia, Brasil
2025

Dedico

A Deus, fonte de toda força e fortaleza, a quem agradeço profundamente pela caminhada acadêmica, pelas graças recebidas ao longo dela e por guiar-me na realização do meu sonho.

À minha mãe, Gleiciane da Silva Roza, e a minha avó, Ilda da Silva Roza, que sempre foram o abraço nos momentos difíceis e o sorriso que ilumina meus dias. Obrigada por estarem comigo em todas as batalhas, enxugando minhas lágrimas e celebrando minhas conquistas.

O amor de vocês é minha maior inspiração.

Ao meu tio, Eliezer Rodrigues Soares, referência de coragem, dedicação e sabedoria. Obrigada por acreditar em mim quando eu mesma duvidava, e por ser a voz de incentivo que ecoa em meu coração.

Ao meu namorado, Pedro Henrique Ferreira Matos, que esteve presente em cada momento difícil, especialmente nas inúmeras noites em claro dedicadas à finalização deste TCC. Sua paciência, compreensão e apoio incansável foram meu porto seguro nos momentos de cansaço e dúvida. Obrigada por acreditar em mim, por incentivar meus sonhos e por ser a força silenciosa que me impulsionou a seguir adiante.

Aos meus amigos, minha segunda família, que com cada palavra de apoio, risada e momento compartilhado, se tornaram tesouros inesquecíveis que carrego comigo. Este trabalho não é apenas meu, mas também de todos vocês, pois sem sua dedicação e carinho, nada disso teria sido possível.

A vocês, dedico cada passo dessa jornada!

Agradecimentos

Gostaria de expressar minha profunda gratidão à minha orientadora, Maria Adriana, pela paciência e compreensão demonstradas ao longo de todo o desenvolvimento deste trabalho. Sua orientação cuidadosa, incentivo constante e disponibilidade para esclarecer dúvidas foram fundamentais para a realização deste projeto.

Agradeço também pela confiança depositada em mim, pelo apoio nos momentos de dificuldade e pela motivação contínua para aprimorar minhas ideias e habilidades. Seu comprometimento e profissionalismo foram inspiradores e contribuíram significativamente para meu crescimento acadêmico e pessoal.

Muito obrigada por todo o suporte e dedicação!

Resumo

Este trabalho descreve o desenvolvimento de um backend para uma plataforma de acompanhamento de egressos da Universidade Federal de Uberlândia (UFU), usando Java com Spring Boot e banco de dados PostgreSQL. O sistema visa consolidar dados sobre a trajetória acadêmica e profissional dos ex-alunos, fornecendo indicadores sobre empregabilidade, titulação e produção científica, em conformidade com as diretrizes do MEC e da CAPES. Além da construção das APIs, este projeto incluiu a integração direta com o frontend, desenvolvido previamente em Angular. Para isso, foram criados serviços específicos garantindo a comunicação eficiente entre as camadas do sistema. Uma adaptação da metodologia ágil Scrum foi utilizada para organizar o desenvolvimento, com ênfase em entregas incrementais e revisões contínuas. O sistema apresenta funcionalidades como painel estatístico com filtros dinâmicos, módulos para cadastro e consulta de informações acadêmicas, experiências profissionais, publicações e depoimentos. O resultado é uma plataforma modular, segura, escalável e alinhada aos objetivos institucionais, promovendo a aproximação entre universidade e egresso, além de auxiliar na tomada de decisões estratégicas.

Palavras-chave: Egressos, Backend, Integração Frontend, Spring Boot, Java, Scrum.

Lista de ilustrações

Figura 1 – Estrutura do sistema.	28
Figura 2 – Página inicial do Sistema de Egressos.	29
Figura 3 – Dashboard do sistema.	30
Figura 4 – Trecho do serviço <code>DashboardService</code> no frontend que realiza a comunicação com o backend.	31
Figura 5 – Continuação do trecho do serviço <code>DashboardService</code> no frontend que realiza a comunicação com o backend.	31
Figura 6 – Visualização das Informações Acadêmicas.	32
Figura 7 – Cadastro de Informações Acadêmicas.	32
Figura 8 – Serviço Angular <code>InformacaoAcademicaService</code> responsável pela comunicação com a API.	33
Figura 9 – Visualização das Informações Profissionais.	34
Figura 10 – Edição de Informação Profissional.	34
Figura 11 – Cadastro de Informação Profissional.	34
Figura 12 – Serviço Angular <code>ProfessionalInformationService</code> responsável pela comunicação com a API.	35
Figura 13 – Visualização dos Depoimentos.	36
Figura 14 – Edição e Exclusão de Depoimentos.	36
Figura 15 – Cadastro de Depoimentos.	37
Figura 16 – Serviço Angular <code>TestimonialsService</code> utilizado para comunicação com o backend.	37
Figura 17 – Visualização de todas as Publicações.	38
Figura 18 – Edição de Publicação.	39
Figura 19 – Cadastro de Publicação.	39
Figura 20 – Serviço Angular <code>PublicationsService</code> utilizado para comunicação com o backend.	40
Figura 21 – Visualização do Perfil – Parte I	41
Figura 22 – Visualização do Perfil – Parte II	41
Figura 23 – Serviço Angular <code>EgressoService</code> responsável pela comunicação com o backend na seção de perfil.	42

Lista de abreviaturas e siglas

API	Application Programming Interface (Interface de Programação de Aplicações)
BPMN	Business Process Model and Notation
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CRUD	Create, Read, Update, Delete
CTIC	Centro de Tecnologia da Informação e Comunicação
DSGov	Design System do Governo Federal
HTTP	HyperText Transfer Protocol
MEC	Ministério da Educação
REST	Representational State Transfer
SCRUM	(Sem tradução direta; é um framework ágil para desenvolvimento de software)
SEI	Sistema Eletrônico de Informações
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
TCC	Trabalho de Conclusão de Curso
UFU	Universidade Federal de Uberlândia

Sumário

1	INTRODUÇÃO	10
1.1	Motivação	12
1.2	Objetivos	12
1.2.1	Objetivo Geral	12
1.3	Metodologia	13
1.4	Organização do Trabalho	14
2	TRABALHOS RELACIONADOS	15
3	FUNDAMENTAÇÃO TEÓRICA	17
3.1	Sistemas de Informação para Web	17
3.2	Metodologia de Desenvolvimento Ágil	18
3.2.1	A Utilização da Metodologia Scrum	18
3.2.2	Vantagens da Utilização do Framework Scrum	19
3.3	Tecnologias Utilizadas no Desenvolvimento Backend	20
3.3.1	Linguagem Java	20
3.3.1.1	Spring Framework	20
3.3.1.2	Apache Maven	20
3.3.2	Arquitetura Cliente-Servidor	21
3.3.2.1	Conceitos e Componentes de um Sistema Cliente-Servidor	21
3.3.2.2	Comunicação entre Frontend e Backend	21
3.3.2.3	Requisições HTTP e APIs RESTful	22
3.3.3	Base de Dados Relacional com PostgreSQL	23
3.3.3.1	Fundamentos de Banco de Dados Relacional	23
3.3.3.2	PostgreSQL	24
3.3.4	Padrões de Interface e Usabilidade	24
3.3.4.1	Uso do DSGov para o Sistema de Acompanhamento de Egressos	24
3.3.5	Integração com outras Plataformas e Sistemas UFU	26
3.3.5.1	Integração com o Sistema Acadêmico da Universidade	26
3.3.5.2	Integração com Plataformas Externas de Registro Profissional (Lattes e LinkedIn)	26
4	IMPLEMENTAÇÃO BACKEND E INTEGRAÇÃO FRONTEND	27
4.1	Estrutura do Sistema	27
4.1.1	Página Inicial	28
4.1.2	Dashboard	29
4.1.3	Informações Acadêmicas	31

4.1.4	Informações Profissionais	33
4.1.5	Depoimentos	35
4.1.6	Publicações	38
4.1.7	Perfil	40
5	CONCLUSÃO	43
	REFERÊNCIAS	45

1 Introdução

O acompanhamento dos egressos de uma universidade é o processo de monitorar a trajetória profissional e acadêmica dos ex-alunos após a conclusão do curso. Essa iniciativa permite avaliar a qualidade da formação oferecida, identificar melhorias nos currículos, fortalecer a relação com os egressos e produzir dados relevantes para contribuir com o planejamento estratégico da universidade.

A importância do monitoramento de egressos ultrapassa o contexto acadêmico, impactando diretamente questões sociais. De acordo com [Guimarães \(2013\)](#), instituições que não realizam esse tipo de acompanhamento demonstram falhas de responsabilidade institucional. Por meio da coleta e análise de dados, é possível identificar pontos fortes e frágeis na formação oferecida, promovendo melhorias contínuas. Nesse sentido, o acompanhamento de egressos configura-se como uma prática de responsabilidade social, contribuindo para o desenvolvimento nacional e para a construção de uma sociedade mais justa e equitativa.

O acompanhamento de egressos também atende às exigências do Ministério da Educação (MEC) ([MEC, 2023](#)) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) ([CAPES, 2018](#)), especialmente no que diz respeito ao desempenho institucional das universidades. A Portaria nº 92, de 31 de janeiro de 2014, tornou obrigatória a implementação de políticas e ações voltadas ao monitoramento dos egressos, incluindo a avaliação de sua atuação no contexto socioeconômico.

A utilização de um sistema de acompanhamento de egressos da Universidade Federal de Uberlândia (UFU) é uma demanda importante e uma atividade fundamental para garantir um monitoramento contínuo dos graduados e pós-graduados ao longo do tempo. Esse tipo de sistema permite a coleta de dados valiosos sobre empregabilidade, áreas de atuação, níveis salariais, expectativas e competências, possibilitando à universidade avaliar o impacto de sua formação acadêmica. Com o aumento do número de egressos, a gestão manual dessas informações tornou-se inviável, reforçando a necessidade de uma solução tecnológica que adote boas práticas de segurança, como autenticação, criptografia e controle de acesso, assegurando a proteção dos dados sensíveis dos egressos.

A relevância do acompanhamento de egressos está ligada à importância de conhecer a trajetória dos ex-alunos e sua inserção no mercado de trabalho. A escolha do tema se justifica pela necessidade de analisar como a universidade atua como mediadora entre a formação acadêmica e o mundo do trabalho. Compreender as condições sob as quais essa mediação ocorre é fundamental para avaliar o papel da instituição na preparação dos jovens para os desafios profissionais ([GUIMARÃES, 2013](#)). Além disso, essa proposta

contribui para que a UFU obtenha subsídios que possibilitem o aprimoramento dos cursos de graduação, impactando positivamente a formação dos estudantes e beneficiando toda a comunidade acadêmica.

Um primeiro trabalho realizado no âmbito do projeto foi responsável pelo levantamento dos requisitos do sistema, identificando as funcionalidades necessárias, os perfis de usuários, os fluxos de uso e as integrações desejadas com plataformas institucionais e profissionais. Esse mapeamento inicial, realizado por [Mendes \(2023\)](#) serviu como base para o planejamento e o desenvolvimento das etapas seguintes. Um segundo trabalho, realizado por [Carvalho \(2025\)](#), propôs uma arquitetura frontend e o design da interface do sistema, seguindo o Design System do Governo Brasileiro (DsGov) ([BRASILEIRO, 2022](#)).

Na fase atual do projeto, o foco voltou-se para o desenvolvimento do backend, estruturado em duas etapas, cada uma conduzida por uma aluna. A primeira etapa ficou sob responsabilidade de Paula Fernanda Rosa do Nascimento, que atuou na modelagem do banco de dados relacional e na implementação inicial de métodos no backend, utilizando a linguagem Java. A segunda etapa foi conduzida pela presente aluna, que desenvolveu e integrou as funcionalidades do backend com o frontend, utilizando Java e o framework Spring Boot, com ênfase na modelagem das entidades, na segurança da informação e na realização de testes unitários.

Durante o desenvolvimento do projeto, foram empregadas ferramentas de software selecionadas com base em critérios técnicos. O backend foi implementado com a linguagem Java ([BESSA, 2023](#)), reconhecida por sua estabilidade e segurança, em conjunto com o framework Spring Boot ([SPRING, 2025](#)), escolhido por sua modularidade e robustez. A arquitetura do sistema baseou-se em APIs REST ([REDHAT, 2024](#)), permitindo a integração entre plataformas distintas. O Apache Maven ([SOUZA, 2023](#)) foi utilizado no gerenciamento de dependências e automação de testes. Para o armazenamento dos dados, optou-se pelo banco de dados relacional PostgreSQL ([POSTGRESQL, 2025](#)), conhecido por sua confiabilidade e conformidade com os padrões SQL.

Todas as decisões técnicas seguiram diretrizes estabelecidas pelos Padrões de Interoperabilidade de Governo Eletrônico (ePING) ([BRASILEIRO, 2018](#)), pelo Design System (DsGov) ([BRASILEIRO, 2022](#)) e pelo Modelo de Acessibilidade em Governo Eletrônico (eMAG) ([BRASILEIRO, 2014](#)). Tais diretrizes foram discutidas e validadas em conjunto com o Centro de Tecnologia da Informação e Comunicação (CTIC), garantindo conformidade com os padrões institucionais.

1.1 Motivação

O acompanhamento de egressos tem-se tornado uma prática essencial nas instituições de ensino superior, pois permite avaliar a efetividade dos cursos oferecidos e o impacto da formação acadêmica na trajetória profissional dos egressos. No entanto, a **UFU** ainda carece de uma plataforma específica que centralize e organize esses dados de forma automatizada, segura e acessível. A ausência desse acompanhamento dificulta a análise de indicadores como empregabilidade, titulação e inserção no mercado, além de limitar o retorno de informações relevantes para melhorias curriculares. A proposta de desenvolvimento de um sistema backend específico busca preencher essa lacuna institucional.

Além da importância estratégica para a universidade, a motivação deste trabalho está relacionada à observação direta das dificuldades enfrentadas por cursos que tentam mapear informalmente a situação de seus egressos. A construção de uma solução tecnológica voltada a esse fim representa não apenas uma contribuição prática para a **UFU**, mas também uma oportunidade de aplicar conhecimentos adquiridos ao longo da graduação em um projeto com potencial de impacto real.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver parcialmente uma plataforma backend voltada ao acompanhamento da trajetória acadêmica e profissional dos egressos da Universidade Federal de Uberlândia (**UFU**), por meio da implementação de funcionalidades principais, integração com o frontend e testes unitários, com o objetivo de apoiar futuras ações de gestão institucional e análise pedagógica.

- **Compreender e analisar os requisitos funcionais e não funcionais**, identificados em conjunto com os stakeholders (como docentes e gestores), por meio de análise documental, com foco em usabilidade, segurança da informação e conformidade com os padrões institucionais, a fim de embasar a implementação do backend.
- **Desenvolver funcionalidades no backend** com Java 18 e Spring Boot, implementando métodos principais da lógica de negócio e garantindo boas práticas de estruturação em camadas e separação de responsabilidades;
- **Integrar a aplicação backend com o frontend**, permitindo a comunicação entre as interfaces e os serviços da API, com foco em operações essenciais para o funcionamento da aplicação;

- **Validar as funcionalidades desenvolvidas por meio de testes unitários**, assegurando que os métodos do backend estejam funcionando corretamente de forma isolada, de acordo com os requisitos especificados.

1.3 Metodologia

O desenvolvimento deste Trabalho de Conclusão de Curso (TCC) foi conduzido com o apoio de reuniões semanais, que contaram com a participação das duas alunas responsáveis pelo projeto e da orientadora. Essas reuniões tinham como objetivo acompanhar o progresso do trabalho, promover a troca de conhecimentos entre as participantes e revisar as entregas técnicas de forma colaborativa. Cada encontro era estruturado para:

1. Analisar as atividades executadas no período anterior;
2. Identificar dificuldades e desafios enfrentados no desenvolvimento do projeto;
3. Propor e implementar ajustes necessários;
4. Planejar as próximas etapas do trabalho.

As reuniões realizadas ao longo do desenvolvimento do projeto tinham caráter informal e focavam na troca de informações, esclarecimento de dúvidas e apresentação do andamento das atividades. Não foram elaborados relatórios formais, sendo a comunicação feita de forma direta entre os membros da equipe. O processo de trabalho seguiu as seguintes etapas principais:

1. **Análise de Projetos Anteriores:** revisão de trabalhos anteriores com o objetivo de compreender o sistema desenhado, as metodologias aplicadas, extrair boas práticas e identificar pontos de melhoria.
2. **Apresentação de caso prático:** apresentação de cenários usando o frontend previamente construído, contando com a participação do aluno responsável, compartilhando os desafios enfrentados e as soluções adotadas. A exposição serviu de referência para a equipe atual, promovendo reflexão e discussão.
3. **Distribuição de Tarefas:** definição de uma estratégia de trabalho colaborativo entre as discentes, na qual, enquanto uma se dedicava ao desenvolvimento de determinada funcionalidade, a outra assumia a responsabilidade pelos testes e validações correspondentes, promovendo integração contínua e complementaridade.
4. **Ajustes no Banco de Dados:** atualização na estrutura do banco de dados com o objetivo de atender às novas demandas identificadas durante a implementação das

funcionalidades, garantindo maior alinhamento entre a modelagem e os requisitos do sistema.

Um dos principais desafios técnicos enfrentados foi a **inconsistência de nomenclaturas** entre as camadas do sistema. O backend utilizava nomes de tabelas e variáveis em português, enquanto o frontend seguia o padrão em inglês. Isso gerou conflitos na integração, exigindo ajustes tanto no banco de dados quanto no código Java.

A necessidade de padronização foi discutida nas reuniões e, com base nisso, as adaptações foram feitas para garantir a coerência entre backend e frontend. Essa medida foi essencial para o correto funcionamento da comunicação entre as camadas e para evitar inconsistências nos dados apresentados ao usuário.

1.4 Organização do Trabalho

Este trabalho está estruturado da seguinte forma:

- Capítulo 1 – Introdução: Apresenta o contexto, a justificativa, os objetivos, a metodologia utilizada e a estrutura do trabalho.
- Capítulo 2 – Trabalhos Relacionados: Analisa sistemas e pesquisas já desenvolvidas sobre acompanhamento de egressos.
- Capítulo 3 – Fundamentação Teórica: Conceitua os elementos técnicos e acadêmicos relacionados ao projeto, como sistemas web, métodos ágeis, arquitetura REST, entre outros.
- Capítulo 4 – Implementação: Descreve a arquitetura, as tecnologias utilizadas, a modelagem e as funcionalidades implementadas.
- Capítulo 5 – Conclusão: Apresenta os resultados alcançados, as limitações do projeto e as propostas de trabalhos futuros.

2 Trabalhos Relacionados

A construção de um sistema para o acompanhamento de egressos da Universidade Federal de Uberlândia (UFU) foi iniciada com o trabalho de [Mendes \(2023\)](#), que teve como foco o levantamento e a análise de requisitos para o desenvolvimento de uma aplicação destinada a monitorar a trajetória acadêmica e profissional dos ex-alunos da instituição. O estudo visou compreender os processos envolvidos e identificar as necessidades institucionais, com o objetivo de fornecer uma base sólida para o desenvolvimento de um software aderente às demandas da universidade.

Para alcançar esse propósito, foram realizadas reuniões, análise documental e investigação de sistemas semelhantes utilizados por outras instituições de ensino superior. A partir dessas atividades, foi possível especificar os requisitos funcionais e não funcionais do sistema, além de modelar os processos de negócio utilizando a notação BPMN (Business Process Model and Notation). O resultado desse trabalho foi uma visão clara e estruturada das funcionalidades essenciais do sistema, orientando o desenvolvimento das etapas subsequentes do projeto.

Dando continuidade à iniciativa, o trabalho de [Carvalho \(2025\)](#) concentrou-se na construção da interface do sistema e na definição da arquitetura frontend. Utilizando o framework Angular e seguindo as diretrizes do Design System do Governo Brasileiro (DsGov), o autor desenvolveu uma aplicação responsiva e alinhada aos padrões de acessibilidade e interoperabilidade estabelecidos pelo governo. Essa etapa foi essencial para garantir uma experiência de usuário satisfatória e coerente com o propósito institucional do sistema.

Ambos os trabalhos foram conduzidos de forma colaborativa. Na sequência, contou-se com a participação de Paula Fernanda Rosa do Nascimento, responsável pelo projeto e implementação do banco de dados relacional em PostgreSQL, além do desenvolvimento dos métodos iniciais no backend, utilizando Java e o framework Spring Boot. Essa etapa foi essencial para estruturar adequadamente a base de dados e preparar o ambiente necessário para as futuras integrações entre o frontend e o backend.

O presente trabalho dá continuidade a essa linha de desenvolvimento, com foco na implementação da **plataforma backend**. Sua principal contribuição está na ampliação das funcionalidades do sistema por meio do desenvolvimento de APIs RESTful, aplicação de boas práticas de segurança (como autenticação, controle de acesso e tratamento de dados sensíveis), modelagem e refinamento de entidades, e implementação de testes unitários para validação das funcionalidades.

A conexão entre os trabalhos de levantamento de requisitos, desenvolvimento do

frontend e estruturação/implementação do backend representa uma evolução coordenada e complementar. Enquanto os estudos anteriores estabeleceram as bases conceituais, visuais e estruturais, este trabalho foca na consolidação da lógica de negócio, integração de componentes e garantia da robustez técnica da aplicação. Essa continuidade garante que o sistema atenda aos objetivos institucionais da UFU, promovendo uma prova de conceito para o acompanhamento eficiente dos egressos e possibilitando futuras expansões, como integração com redes profissionais e geração de relatórios para apoio à tomada de decisões acadêmicas e administrativas.

3 Fundamentação Teórica

3.1 Sistemas de Informação para Web

Os Sistemas de Informação (SI) para Web constituem uma categoria de software voltada à coleta, processamento, armazenamento e disponibilização de informações por meio da internet. Segundo (LAUDON; LAUDON, 2021), um sistema de informação é um conjunto inter-relacionado de componentes que coleta (ou recupera), processa, armazena e distribui informações para apoiar a tomada de decisão, a coordenação e o controle em uma organização.

Com a popularização da internet e das arquiteturas distribuídas, os SI evoluíram para modelos baseados em navegadores, acessíveis a partir de qualquer dispositivo conectado em rede, caracterizando os chamados Web Information Systems (WIS). Conforme (PRESSMAN; MAXIM, 2023), os WIS integram três camadas principais: a interface com o usuário (frontend), a lógica de aplicação (backend) e o sistema gerenciador de banco de dados (SGBD), usualmente distribuídas em arquiteturas cliente-servidor ou em múltiplos microsserviços.

A principal vantagem dos sistemas de informação baseados na Web está na acessibilidade universal, eliminando restrições físicas ou geográficas, além de possibilitar integrações com outras plataformas por meio de APIs (*Application Programming Interfaces*). Ainda, tais sistemas tendem a ser **altamente escaláveis** e adaptáveis a diferentes cargas de usuários. Além disso, os sistemas web modernos adotam boas práticas como:

- Separação de responsabilidades entre frontend e backend;
- Padrões de interoperabilidade como REST, GraphQL e JSON;
- Camadas de segurança com autenticação baseada em tokens (ex.: JWT), criptografia TLS e controle de acesso baseado em papéis;
- Observabilidade e monitoramento por meio de logs estruturados, métricas e rastreamento de requisições.

No contexto deste trabalho, o sistema de acompanhamento de egressos foi concebido com base na arquitetura RESTful, utilizando Java 18 e o framework Spring Boot no backend. A comunicação entre cliente e servidor é realizada via requisições HTTP com troca de dados no formato JSON, seguindo os princípios propostos por (FIELDING, 2000) em sua tese sobre arquiteturas web. Já a persistência dos dados é gerenciada por um

banco de dados relacional PostgreSQL, devido à sua robustez, flexibilidade e aderência ao padrão SQL.

Tais decisões visam garantir manutenibilidade, segurança, reuso de componentes e escalabilidade horizontal, características essenciais em sistemas institucionais com potencial de crescimento e múltiplos usuários simultâneos.

3.2 Metodologia de Desenvolvimento Ágil

Na Metodologia Ágil, as pessoas são essenciais para o sucesso dos projetos, sendo fundamental uma comunicação clara e motivação nas equipes, com grande foco na qualidade. Ela valoriza a entrega de um produto funcional que atenda realmente às necessidades do cliente. O objetivo é desenvolver o software, gerando a maior parte da documentação a partir das ferramentas de produção. Dessa forma, o cliente participa ativamente de cada iteração, influenciando os novos requisitos, ao contrário das metodologias tradicionais, onde quase tudo é planejado no início.

3.2.1 A Utilização da Metodologia Scrum

O framework Scrum tem sido amplamente aplicado no desenvolvimento de projetos com requisitos voláteis e alta complexidade. Inicialmente proposto por Jeff Sutherland e Ken Schwaber nos anos 1990, ele se consolidou como uma das principais abordagens ágeis voltadas à entrega contínua de valor e adaptação rápida a mudanças de escopo (SUTHERLAND; SCHWABER, 2020).

A aplicação prática do Scrum ocorre por meio de ciclos curtos de trabalho, denominados Sprints, nos quais as equipes desenvolvem partes incrementais do produto. Ao final de cada ciclo, realiza-se uma revisão com os envolvidos, permitindo ajustes antes do próximo Sprint. Essa estrutura iterativa e incremental facilita a incorporação de feedback constante e promove maior alinhamento entre a equipe e os objetivos do projeto (FERREIRA; LIMA, 2021).

Durante sua utilização, o framework também incentiva práticas como reuniões diárias (Daily Scrum), revisões de Sprint e retrospectivas, que contribuem para a transparência, a comunicação eficaz e a melhoria contínua dos processos. Tais práticas tornam o Scrum não apenas uma estrutura de gestão, mas uma ferramenta ativa de adaptação organizacional (SILVA; ROCHA, 2022).

Portanto, ao adotar o Scrum, equipes conseguem lidar melhor com a complexidade e incertezas do desenvolvimento de software, priorizando a colaboração, o feedback rápido e a entrega incremental de valor.

3.2.2 Vantagens da Utilização do Framework Scrum

A adoção do Scrum em projetos de desenvolvimento tem se mostrado eficaz especialmente em ambientes marcados por complexidade e incerteza. Segundo [Awari \(2023\)](#), a aplicação prática desse framework, traz diversas vantagens que contribuem para a organização do trabalho, a produtividade da equipe e a qualidade das entregas. Entre os principais benefícios observados na sua utilização, destacam-se:

1. **Maior flexibilidade e adaptação:** Com ciclos de desenvolvimento curtos (Sprints) e revisões frequentes, o Scrum permite ajustes contínuos no projeto com base no feedback recebido, ajudando a equipe a adaptar-se rapidamente a mudanças nos requisitos ou nas necessidades do cliente.
2. **Foco na entrega de valor:** O Scrum enfatiza a entrega de incrementos funcionais e de valor para o cliente ao final de cada Sprint, garantindo que o produto evolua com entregas que trazem benefícios concretos e que atendem às expectativas.
3. **Engajamento do cliente:** A participação ativa do cliente em cada ciclo de desenvolvimento e nas revisões de Sprint permite um alinhamento constante entre o que é desenvolvido e o que o cliente realmente precisa. Esse engajamento ajuda a evitar mal-entendidos e mantém o projeto no caminho certo.
4. **Melhoria da comunicação e colaboração:** O Scrum promove uma comunicação frequente e estruturada entre os membros da equipe e os stakeholders, por meio de cerimônias regulares (Daily Stand-ups, Revisão de Sprint, Planejamento de Sprint), incentivando a colaboração e a troca de informações.
5. **Incremento na produtividade:** Com metas claras e bem definidas para cada Sprint, o Scrum mantém o foco e a motivação da equipe, resultando em maior produtividade e menos dispersão de esforços. A equipe trabalha em tarefas priorizadas, o que evita a sobrecarga e desperdício de recursos.
6. **Deteção e resolução rápida de problemas:** A abordagem iterativa do Scrum permite a identificação precoce de problemas e falhas, reduzindo o tempo de resposta e possibilitando que a equipe resolva os problemas antes que se tornem grandes obstáculos.
7. **Transparência e melhoria contínua:** Através de revisões constantes e da retrospectiva ao final de cada Sprint, o Scrum promove uma cultura de transparência, onde o progresso e os desafios são visíveis para todos. A equipe é incentivada a refletir sobre o que funcionou e o que pode ser melhorado, criando um ciclo de melhoria contínua.

Essas vantagens reforçam a eficácia do Scrum na condução de projetos dinâmicos, contribuindo para entregas mais assertivas, maior satisfação dos envolvidos e processos organizacionais mais eficientes.

3.3 Tecnologias Utilizadas no Desenvolvimento Backend

3.3.1 Linguagem Java

Java é uma das plataformas mais amplamente utilizadas no desenvolvimento de aplicações em todo o mundo. Conforme os dados da Pesquisa “Tecnologias mais populares de 2022”, realizada pelo StackOverflow, a linguagem Java figura como a 6^a mais utilizada por pessoas desenvolvedoras (BESSA, 2023), correspondendo a 33% do total pesquisado em relação a outras linguagens existentes.

3.3.1.1 Spring Framework

O Spring Framework é um conhecido framework de software livre, voltado para criação de aplicações empresariais na JVM que facilita a codificação, oferecendo suporte completo para o desenvolvimento de software. Ele reduz a complexidade do desenvolvimento e oferece diversas ferramentas e bibliotecas para acelerar o processo de desenvolvimento (QNAX, 2024).

3.3.1.2 Apache Maven

O Maven é uma ferramenta de gerenciamento e compreensão de projetos feitas para auxiliar os códigos feitos. Mas possui conceitos e funcionalidades diferentes, em especial podemos falar da base do Maven que é o Modelo de Objeto de Projeto, este modelo organiza todas as informações do projeto em um único arquivo: o `pom.xml` (SOUZA, 2023).

O arquivo `pom.xml` é utilizado para descrever as características do projeto de software em desenvolvimento, incluindo suas dependências externas, a ordem de compilação, os diretórios envolvidos e os plug-ins necessários para a construção do projeto (APACHE, 2025). Nesse arquivo estão especificadas todas as dependências que serão utilizadas, permitindo que o Maven realize automaticamente o download e o armazenamento desses componentes em um repositório local.

3.3.2 Arquitetura Cliente-Servidor

3.3.2.1 Conceitos e Componentes de um Sistema Cliente-Servidor

O termo Cliente-Servidor refere-se ao modelo de distribuição de aplicações computacionais em múltiplas plataformas. Nesse modelo, as aplicações geralmente são divididas entre um provedor de acesso e uma central de dados, além de diversos clientes com interfaces gráficas que possibilitam o acesso e a manipulação de dados (CURTI, 2004). Esse modelo é composto pelos componentes Cliente e Servidor, que representam, respectivamente, as partes solicitante e fornecedora de serviços ou recursos. O Cliente é responsável por realizar solicitações, enquanto o Servidor é o programa ou dispositivo encarregado de atender a essas requisições.

O Cliente ou Frontend é um dispositivo de computação, como desktops, laptops, smartphones, tablets ou aplicativos que, através de protocolos de rede se conectam a servidores para fazer uma solicitação de serviço ou recurso e aguarda a resposta. Por sua vez, o Servidor ou Backend é o componente do sistema que fornece os serviços ou recursos solicitados pelos clientes. Estes servidores podem ser um computador projetado especificamente para gerenciar e fornecer serviços aos clientes, ou ainda, um software que é executado em um servidor de rede (BASS; CLEMENTS; KAZMAN, 2012).

A arquitetura cliente-servidor está incorporada aos SGBDs. Nos clientes estão os programas de aplicação e a interface do usuário, enquanto as funcionalidades de processamento de consultas e transações permanecem no lado do servidor. Quando é necessário acesso ao banco de dados, a aplicação solicita uma conexão com o SGBD, que está no lado do servidor, e, uma vez que a conexão é estabelecida, os programas no cliente podem se comunicar com o SGBD (CORTES, 2004).

3.3.2.2 Comunicação entre Frontend e Backend

O Backend e o Frontend, quando desenvolvidos utilizando **Spring Boot (Java)** e **Angular**, permitem uma combinação eficiente para o desenvolvimento de aplicações web. Juntos, esses frameworks permitem a criação de sistemas dinâmicos e interativos, otimizando o uso de recursos e garantindo um desempenho eficaz.

A comunicação entre o Frontend e o Backend é realizada por meio de requisições e respostas, seguindo o protocolo **HTTP** (Hyper Text Transfer Protocol). O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso. Embora esses métodos possam ser descritos como substantivos, eles também são comumente referenciados como *HTTP Verbs* (VerbosHTTP).

3.3.2.3 Requisições HTTP e APIs RESTful

As **APIs RESTful** (Representational State Transfer) utilizam o protocolo **HTTP** para facilitar a comunicação entre sistemas na internet. Elas permitem que diferentes aplicações interajam de forma padronizada, executando operações como criar, ler, atualizar e deletar registros em um recurso.

Cada requisição HTTP é composta por um **verbo** (método), um **endpoint** (URL), **cabeçalhos** e, opcionalmente, um **corpo**. A seguir, são apresentados os métodos mais comuns e suas respectivas funcionalidades:

- **GET**: Usado para recuperar informações de um recurso específico. As requisições com esse método devem apenas obter dados, sem realizar alterações no servidor.
- **HEAD**: Similar ao GET, porém retorna apenas os cabeçalhos da resposta, sem incluir o corpo da mensagem.
- **POST**: Serve para enviar dados a um recurso, geralmente resultando na criação ou modificação de informações no servidor.
- **PUT**: Substitui completamente um recurso existente com os dados fornecidos na requisição.
- **DELETE**: Remove um recurso específico do servidor.
- **CONNECT**: Estabelece um túnel entre o cliente e o servidor, permitindo comunicações seguras ou intermediadas.
- **OPTIONS**: Exibe as opções disponíveis de comunicação para um determinado recurso ou servidor.
- **TRACE**: Realiza um teste de conexão, retornando exatamente o que foi enviado na requisição para fins de depuração.
- **PATCH**: Aplica alterações parciais a um recurso existente, modificando apenas os campos especificados na requisição.

Esses métodos permitem que as APIs RESTful realizem operações padrão de banco de dados, conhecidas como **CRUD** (Create, Read, Update, Delete), de maneira consistente e previsível ([RAMAKRISHNAN; GEHRKE, 2008](#)).

A arquitetura REST estabelece um conjunto de restrições ([FIELDING et al., 2017](#)) que, quando corretamente aplicadas, contribuem para a construção de sistemas escaláveis, de fácil manutenção e com desempenho otimizado. Entre essas restrições, destacam-se o uso de uma interface uniforme, a representação de recursos por meio de URLs e a

comunicação sem estado entre cliente e servidor. As requisições HTTP desempenham um papel essencial nas APIs RESTful, viabilizando a interação eficiente entre sistemas distintos e a execução de operações variadas de maneira padronizada e segura.

3.3.3 Base de Dados Relacional com PostgreSQL

3.3.3.1 Fundamentos de Banco de Dados Relacional

Um banco de dados, ou base de dados, pode ser entendido como um espaço – físico ou digital – destinado ao armazenamento estruturado e indexado de informações (RAMAKRISHNAN; GEHRKE, 2008). Esse conceito não se limita apenas à área da informática, mas também se aplica a contextos físicos e analógicos. Com isso em mente, exemplos de bancos de dados incluem um bloco de notas, um documento de texto, uma planilha, um caderno e até mesmo um baú. Além disso, a própria internet pode ser considerada um vasto banco de dados.

Os **bancos de dados relacionais** são sistemas projetados para armazenar, organizar e gerenciar dados estruturados em tabelas inter-relacionadas. Cada tabela, também conhecida como relação, contém colunas, chamadas de atributos, e é preenchida por registros, ou tuplas, que armazenam as informações do banco de dados. O conjunto dessas tabelas e seus respectivos atributos compõe o que é conhecido como um **esquema relacional**:

- **Tabelas:** Estruturas bidimensionais que armazenam dados em linhas e colunas. Cada tabela representa uma entidade do mundo real, como “Clientes” ou “Pedidos”.
- **Linhas (ou Registros/Tuplas):** Cada linha em uma tabela representa uma instância específica dessa entidade, contendo dados relacionados a um item individual.
- **Colunas (ou Atributos):** Cada coluna representa uma característica da entidade, como “Nome”, “Data de Nascimento” ou “Endereço”.

Para garantir a organização e a integridade dos dados, os bancos de dados relacionais utilizam chaves para estabelecer relações entre tabelas e identificar registros de forma única:

- **Chave Primária:** Um identificador único para cada registro em uma tabela, garantindo que cada entrada possa ser distinguida das demais (ORACLE, 2020a).
- **Chave Estrangeira:** Um campo que estabelece uma ligação entre duas tabelas, permitindo a criação de relações e garantindo a integridade referencial dos dados (ORACLE, 2020b).

Em resumo, os fundamentos dos bancos de dados relacionais proporcionam uma base sólida para o armazenamento e gerenciamento eficiente de dados, sendo essenciais para o funcionamento de inúmeras aplicações modernas.

3.3.3.2 PostgreSQL

O **PostgreSQL** é um sistema de gerenciamento de banco de dados objeto-relacional de código aberto, reconhecido por sua robustez, extensibilidade e conformidade com os padrões SQL. Por muito tempo, foi discriminado no mundo dos bancos de dados, e o seu recente aumento de popularidade veio de usuários de outros bancos de dados em busca de um sistema com melhores garantias de confiabilidade, melhores recursos de consulta, mais operação previsível, ou simplesmente querendo algo mais fácil de aprender, entender e usar ([POSTGRESQL, 2025](#)). Ele se destaca por algumas vantagens e ampla aplicabilidade em diferentes cenários, como:

- **Extensibilidade:** O PostgreSQL permite a adição de novos tipos de dados, funções e operadores, oferecendo flexibilidade para atender a necessidades específicas de diferentes aplicações.
- **Suporte a Transações ACID:** Garante a confiabilidade das operações de banco de dados, assegurando que as transações sejam atômicas, consistentes, isoladas e duráveis.
- **Comunidade Ativa:** Conta com uma comunidade global ativa que contribui para o desenvolvimento contínuo, documentação e suporte, garantindo a evolução constante da plataforma.

Devido a essas características, o PostgreSQL é amplamente adotado em diversos setores, desde startups até grandes corporações, para aplicações que exigem confiabilidade, escalabilidade e flexibilidade no gerenciamento de dados.

3.3.4 Padrões de Interface e Usabilidade

3.3.4.1 Uso do DSGov para o Sistema de Acompanhamento de Egressos

O **DSGov**, ou Design System do Governo Federal, é um conjunto de diretrizes, componentes e padrões visuais desenvolvido para padronizar e aprimorar a experiência dos usuários nos serviços digitais do governo brasileiro. Seu objetivo é garantir consistência, acessibilidade e eficiência na interface e na usabilidade dos sites e aplicativos governamentais. As principais características de utilizar o **DSGov** de acordo com o (??) :

- **Consistência Visual e Funcional:** Ao adotar o DSGov, os órgãos públicos asseguram que seus serviços digitais mantenham uma identidade visual uniforme, facilitando o reconhecimento e a navegação por parte dos cidadãos.
- **Acessibilidade:** O design system incorpora práticas que atendem às diretrizes de acessibilidade, garantindo que pessoas com deficiência possam utilizar os serviços digitais sem barreiras.
- **Eficiência no Desenvolvimento:** Disponibilizando uma biblioteca de componentes reutilizáveis, o DSGov acelera o processo de desenvolvimento e manutenção de interfaces, permitindo que as equipes técnicas foquem em funcionalidades específicas sem se preocupar com a criação de elementos básicos.

A adoção do **DSGov** no desenvolvimento do **Sistema de Acompanhamento de Egressos**, conforme apresentado em [Carvalho \(2025\)](#), resultou em benefícios significativos para a interface do sistema, promovendo padronização, acessibilidade e eficiência. Por se tratar de um projeto voltado ao acompanhamento de ex-alunos, a utilização de um design consistente e alinhado às diretrizes institucionais é fundamental para garantir uma navegação intuitiva e uma experiência de uso aprimorada. O uso do DSGov, portanto, contribui diretamente para a construção de uma interface mais inclusiva, coerente e funcional. Entre os principais benefícios, destacam-se:

- **Consistência Visual e Identidade Governamental:** A utilização dos componentes padronizados do DSGov assegura que o sistema mantenha a identidade visual dos serviços públicos digitais, facilitando o reconhecimento pelos usuários e promovendo maior confiabilidade.
- **Acessibilidade Aprimorada:** O design system segue diretrizes que garantem a inclusão de pessoas com deficiência, tornando o sistema acessível a todos os egressos, independentemente de limitações físicas ou tecnológicas.
- **Agilidade no Desenvolvimento:** A biblioteca de componentes reutilizáveis do DSGov acelera o processo de construção da interface, permitindo que a equipe de desenvolvimento se concentre na implementação das funcionalidades específicas do sistema.
- **Melhoria na Experiência do Usuário:** Com padrões bem definidos e interface responsiva, os egressos poderão interagir com o sistema de forma intuitiva, reduzindo a curva de aprendizado e aumentando a adesão à plataforma.

A aplicação no **Sistema de Acompanhamento de Egressos** pode ser realizada por meio da utilização dos Web Components do **DSGov**, garantindo que formulários, botões e

layouts sigam o padrão governamental, proporcionando uma interface visual padronizada e intuitiva (CARVALHO, 2025). Além disso, a implementação de diretrizes de acessibilidade assegura que o sistema seja navegável por meio de tecnologias assistivas, como leitores de tela, tornando a plataforma inclusiva para todos os usuários. Por fim, a adaptação do design no Figma permite a criação de protótipos alinhados às diretrizes visuais do governo, facilitando a uniformidade estética e funcional da aplicação.

Dessa forma, a adoção do **DSGov** no **Sistema de Acompanhamento de Egressos** não apenas melhora a experiência dos usuários, mas também assegura que a aplicação siga as melhores práticas do governo digital, promovendo eficiência e inclusão digital.

3.3.5 Integração com outras Plataformas e Sistemas UFU

3.3.5.1 Integração com o Sistema Acadêmico da Universidade

A integração do Sistema Acadêmico da Universidade Federal de Uberlândia (UFU) com outras plataformas e sistemas é essencial para aprimorar a eficiência administrativa e a experiência acadêmica. A UFU utiliza o Sistema Eletrônico de Informações (SEI-UFU), composto por módulos que facilitam a gestão de processos internos e a comunicação institucional (UFU, 2024).

3.3.5.2 Integração com Plataformas Externas de Registro Profissional (Lattes e LinkedIn)

A Plataforma Lattes, gerenciada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), é um padrão nacional para o registro da trajetória acadêmica e profissional de pesquisadores e estudantes no Brasil (CNPQ, 2025). Sua integração com os sistemas universitários permite a atualização automática de dados acadêmicos e de pesquisa, facilitando processos de avaliação e fomentando a transparência.

Por outro lado, o LinkedIn é uma plataforma global voltada para networking profissional, permitindo que usuários compartilhem experiências, habilidades e conquistem oportunidades no mercado de trabalho. A integração dos sistemas universitários com o LinkedIn pode aumentar a visibilidade dos alunos e egressos, conectando-os a uma rede profissional mais ampla e potencializando suas carreiras (LINKEDIN, 2025).

4 Implementação Backend e Integração Frontend

Este capítulo detalha a implementação do backend do sistema, incluindo a construção das APIs REST, a modelagem da base de dados e a lógica de negócios. Além disso, foi desenvolvida a integração com o frontend, criando os serviços necessários para que a interface do usuário se comunique corretamente com o backend, garantindo a troca de dados e o funcionamento harmonioso de todas as funcionalidades apresentadas neste sistema.

Todo o código-fonte relacionado ao backend e à integração com o frontend encontra-se disponível publicamente em um repositório no GitHub¹, permitindo o acompanhamento do histórico de versões, o registro e monitoramento de problemas técnicos e a colaboração aberta com outros desenvolvedores interessados na evolução da plataforma.

4.1 Estrutura do Sistema

O sistema foi projetado com uma interface limpa e intuitiva, utilizando um menu lateral fixo à esquerda da tela. Esse menu é composto por ícones e descrições textuais que facilitam a navegação dos usuários pelas principais funcionalidades da plataforma. As opções disponíveis incluem: **Página Inicial**, **Dashboard**, **Informações Acadêmicas**, **Informações Profissionais**, **Depoimentos**, **Publicações** e **Perfil** do usuário. Cada item do menu redireciona para uma seção específica do sistema, promovendo uma navegação rápida e organizada.

A Figura 1 apresenta a estrutura visual do sistema, ilustrando o layout da interface com o menu lateral e os elementos de navegação. Ressalta-se que essa imagem representa apenas a interface do frontend, sem integração com a lógica de funcionamento do backend.

¹ <https://github.com/VitoriaCardoso/egress-system>

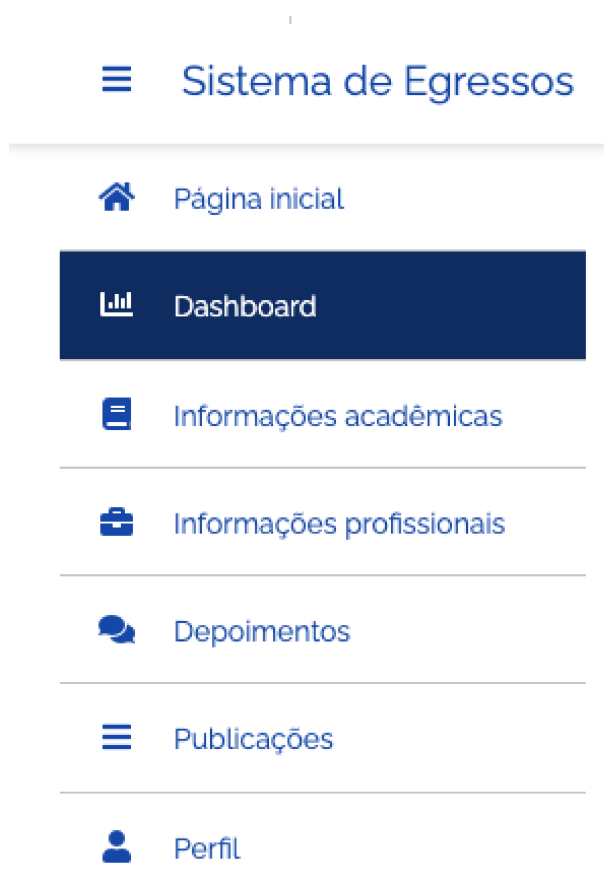


Figura 1 – Estrutura do sistema.

4.1.1 Página Inicial

A página inicial exibe uma mensagem institucional de boas-vindas ao usuário, ressaltando a importância de manter a conexão com a UFU. São incentivadas ações como o acompanhamento de novidades, oportunidades de desenvolvimento profissional e a manutenção de laços com a comunidade acadêmica que fez parte da trajetória do egresso.

A Figura 2 ilustra a interface visual da página inicial do sistema, apresentando o conteúdo exibido ao usuário logo após o login. É importante destacar que esta imagem representa apenas a estrutura do frontend, ainda sem qualquer integração com o backend ou funcionalidades dinâmicas implementadas.

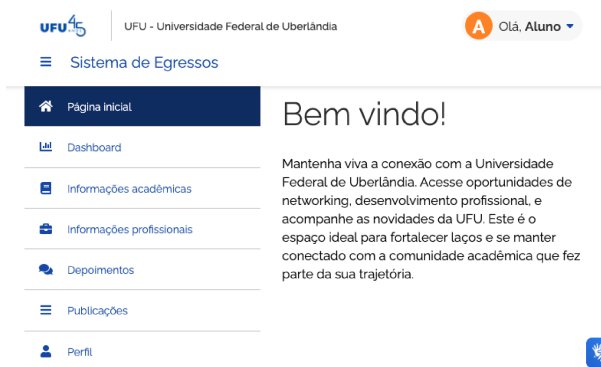


Figura 2 – Página inicial do Sistema de Egressos.

A comunicação entre o frontend e o backend do sistema é realizada por meio de uma API REST, que serve como intermediária para a troca de dados entre as duas camadas. O backend, desenvolvido com Java e Spring Boot, disponibiliza endpoints HTTP que expõem os serviços necessários para o funcionamento do sistema, incluindo acesso a informações armazenadas no banco de dados relacional.

No frontend, desenvolvido com Angular, o módulo `HttpClient` é utilizado para enviar requisições HTTP para esses endpoints, buscando ou enviando dados conforme as necessidades da interface. Essa arquitetura permite que o frontend permaneça desacoplado da lógica de negócio e da persistência de dados, concentrando-se na apresentação e na interação com o usuário.

4.1.2 Dashboard

O *dashboard* disponibiliza dados públicos dos egressos por meio de filtros dinâmicos que permitem segmentar as informações por semestre letivo, curso, titulação e campus. Os dados são exibidos em formato de gráficos e tabelas, proporcionando uma visualização intuitiva e de fácil interpretação. Entre os elementos apresentados estão:

- Quantidade de estudantes ativos e inativos;
- Distribuição por campus;
- Distribuição por titulação acadêmica;
- Listagem de depoimentos relacionados aos cursos.

As Figuras 3a e 3b ilustram a interface visual do *dashboard*. Embora representem apenas o frontend, neste ponto do desenvolvimento já existe integração com o backend, possibilitando a exibição de dados reais vindos da API. Assim, o frontend realiza chamadas para a API para obter dados atualizados sobre os egressos, tais como status acadêmico,

campus, titulação e depoimentos. Os dados recebidos no formato JSON são processados e exibidos dinamicamente em gráficos e tabelas, garantindo que as informações apresentadas sejam sempre atuais e reflitam o estado real do banco de dados.

Além da recuperação de dados, a integração também suporta a aplicação de filtros combinados, enviados como parâmetros nas requisições HTTP. Isso possibilita consultas específicas, otimizando a quantidade de dados trafegados e melhorando a experiência do usuário.

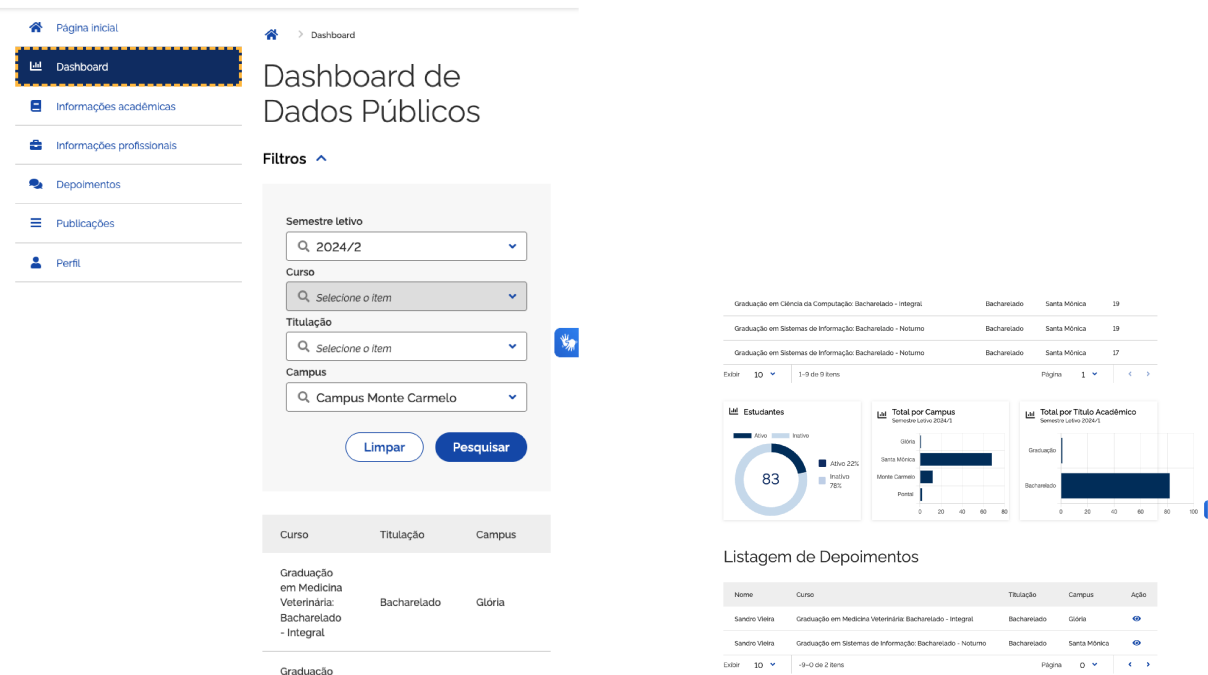


Figura 3 – Dashboard do sistema.

No frontend, implementado com Angular, o módulo `HttpClient` é utilizado para consumir esses serviços, enviando requisições HTTP e recebendo dados no formato JSON. O serviço ilustrado nas Figuras 4 e 5 exemplifica essa comunicação, apresentando métodos que permitem a recuperação de informações variadas, como status dos estudantes, campus, cursos, titulações, semestres e depoimentos. Além disso, esses métodos suportam a filtragem combinada dos dados com base nos parâmetros selecionados pelo usuário, otimizando as consultas e a exibição dos resultados.

```

You, semana passada | 1 author (You)
@Injectable({
  providedIn: 'root',
})
export class DashboardService {
  private apiUrl = 'http://localhost:8080/api/dashboard';

  constructor(private http: HttpClient) {}

  getStatusEstudantes(): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/estudantes/status`);
  }

  getNivelEstudantes(): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/estudantes/nivel`);
  }

  getCampusEstudantes(): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/estudantes/campus`);
  }

  getCampi(): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/campus`);
  }

  getCursos(): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/cursos`);
  }

  getTitulacoes(): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/titulacoes`);
  }
}

```

Figura 4 – Trecho do serviço DashboardService no frontend que realiza a comunicação com o backend.

```

getSemestres(): Observable<SelectOption[]> {
  return this.http.get<SelectOption[]>(`${this.apiUrl}/semestres`);
}

getDepoimentos(): Observable<StudentTestimony[]> {
  return this.http.get<StudentTestimony[]>(`${this.apiUrl}/depoimentos`);
}

getCursoCampusTitulacao(
  campus?: string,
  semestre?: string,
  titulacao?: string,
  curso?: string
): Observable<Course[]> {
  let params = new HttpParams();
  if (campus) params = params.set('campus', campus);
  if (semestre) params = params.set('semestre', semestre);
  if (titulacao) params = params.set('titulacao', titulacao);
  if (curso) params = params.set('curso', curso);

  return this.http.get<Course[]>(`${this.apiUrl}/estudantes/curso-campus-titulacao`, { params });
}
}

```

Figura 5 – Continuação do trecho do serviço DashboardService no frontend que realiza a comunicação com o backend.

4.1.3 Informações Acadêmicas

Esta seção exibe a formação acadêmica do egresso. O sistema permite apenas a adição de novos registros, não sendo possível editar ou excluir as informações após o

cadastro. Essa abordagem foi adotada para garantir a integridade e autenticidade dos dados inseridos.

As Figuras 6 e 7 apresentam, respectivamente, a visualização e o formulário de cadastro das informações acadêmicas na interface do sistema. Embora essas imagens representem apenas a interface do frontend, elas já estão integradas ao backend. Os dados exibidos na tela são obtidos diretamente do banco de dados por meio de chamadas a uma API REST.

A interface exibe o menu lateral "Sistema de Egressos" com opções: Página inicial, Dashboard, Informações acadêmicas (selecionada), Informações profissionais, Depoimentos, Publicações e Perfil. O cabeçalho da seção "Informações Acadêmicas" contém o botão "Novo Curso". O conteúdo principal mostra os dados de uma graduação em Medicina Veterinária:

Graduação em Medicina Veterinária: Bacharelado - Integral - UFU			
Nome da Instituição	Tipo de Instituição	Nome do Curso	Nível do Curso
Universidade Federal de Uberlândia	Instituição Pública	Graduação em Medicina Veterinária: Bacharelado - Integral	Bacharelado
Pais	Data de Ingresso	Data de Conclusão	Estado
Brasil	22/03/2012	25/04/2017	Minas Gerais
Cidade	Matrícula		
Uberlândia	11221MEV112		

Abaixo, há uma seção para "Graduação em Sistemas de Informação: Bacharelado - Noturno - UFU".

Figura 6 – Visualização das Informações Acadêmicas.

A interface exibe o menu lateral "Sistema de Egressos" com opções: Página inicial, Dashboard, Informações acadêmicas (selecionada), Informações profissionais, Depoimentos, Publicações e Perfil. O cabeçalho da seção "Cadastrar Informações" contém o subtítulo "Somente cursos integralizados externamente à UFU". O formulário principal contém os seguintes campos:

Nome da Instituição	Tipo de Instituição (dropdown: "Selecione o item")	Nome do Curso	Matrícula
Campus	Número de Registro	Titulação (dropdown: "Selecione o item")	
Data de Ingresso (exemplo: 02/02/2024)	Data de Conclusão (exemplo: 02/02/2024)		
Pais	Estado	Cidade	CPF (exemplo: 123.456.789-14)

Um botão "Salvar" está localizado na parte inferior do formulário.

Figura 7 – Cadastro de Informações Acadêmicas.

Essa comunicação entre frontend e backend é realizada por meio de um serviço Angular, responsável por buscar, cadastrar e gerenciar as informações acadêmicas. A Figura 8 mostra a implementação desse serviço, chamado `InformacaoAcademicaService`.

```
@Injectable({
  providedIn: 'root',
})
export class InformacaoAcademicaService {
  private apiUrl = 'http://localhost:8080/api/informacoes/academicas';

  constructor(private http: HttpClient) {}

  buscarTodos(): Observable<AcademicInformation[]> {
    return this.http.get<AcademicInformation[]>(`${this.apiUrl}`);
  }

  buscarPorEgresso(cpf: string): Observable<AcademicInformation[]> {
    return this.http.get<AcademicInformation[]>(`${this.apiUrl}/egresso/${cpf}`);
  }

  buscarPorInformacaoAcademica(id: string): Observable<AcademicInformation[]> {
    return this.http.get<AcademicInformation[]>(`${this.apiUrl}/editar/${id}`);
  }

  buscarCursoPorCpf(cpf: string): Observable<SelectOption[]> {
    return this.http.get<SelectOption[]>(`${this.apiUrl}/cursos/${cpf}`);
  }

  criarInformacaoAcademica(data: AcademicInformation): Observable<AcademicInformation> {
    return this.http.post<AcademicInformation>(this.apiUrl, data);
  }

  atualizarInformacaoAcademica(id: string, data: AcademicInformation): Observable<AcademicInformation> {
    return this.http.put<AcademicInformation>(`${this.apiUrl}/${id}`, data);
  }

  excluirInformacaoAcademica(id: number): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

Figura 8 – Serviço Angular `InformacaoAcademicaService` responsável pela comunicação com a API.

4.1.4 Informações Profissionais

Na seção de informações profissionais, o egresso pode registrar experiências de trabalho vinculadas à sua formação acadêmica. Diferentemente da seção anterior, aqui é possível editar individualmente cada experiência cadastrada. Essa funcionalidade oferece flexibilidade para o egresso manter suas informações atualizadas, além de facilitar análises da trajetória profissional.

As Figuras 9, 10 e 11 apresentam, respectivamente, a visualização geral das experiências cadastradas, a interface de edição e o formulário de cadastro de novas informações profissionais. Embora as imagens mostrem apenas a interface do frontend, esta seção já está completamente integrada ao backend, permitindo a recuperação e o envio de dados reais provenientes do banco de dados.

UFU45 UFU - Universidade Federal de Uberlândia

S Olá, Sandro

Sistema de Egressos

Página inicial

Dashboard

Informações acadêmicas

Informações profissionais

Depoimentos

Publicações

Perfil

Informações Profissionais

Nova Experiência

Editar Experiência

Clínica Vida Animaluu - Médico Veterinário

Nome da Empresa	Categoria	Tipo de Emprego	Localidade
Clínica Vida Animaluu	Privada	Tempo Integral	Remoto
Cargo	Nível de Cargo	Data de Início	Data de Término
Médico Veterinário	Júnior	15/03/2019	15/05/2022
Média Salarial	Função	Informações Acadêmicas Relacionadas	Identificador
R\$ 7.200,00	Atendimento clínico de animais de pequeno porte	Graduação em Medicina Veterinária: Bacharelado - Integral	37bd7667-15c0-4dc1-bdff-e6e80169014b

dfgb - hhh

Figura 9 – Visualização das Informações Profissionais.

UFU45 UFU - Universidade Federal de Uberlândia

S Olá, Sandro

Sistema de Egressos

Página inicial

Dashboard

Informações acadêmicas

Informações profissionais

Depoimentos

Publicações

Perfil

Editar Informações

Nome da Empresa

Categoria

Tipo de Emprego

Localidade

Cargo

Nível do Cargo

Função

Média Salarial

Data de Ingresso

Data de Conclusão

Curso Relacionado

Excluir

Salvar

Figura 10 – Edição de Informação Profissional.

UFU45 UFU - Universidade Federal de Uberlândia

S Olá, Sandro

Sistema de Egressos

Página inicial

Dashboard

Informações acadêmicas

Informações profissionais

Depoimentos

Publicações

Perfil

Cadastrar Informações

Nome da Empresa

Categoria

Tipo de Emprego

Localidade

Cargo

Nível do Cargo

Função

Média Salarial

Data de Ingresso

Data de Conclusão

Curso Relacionado

Salvar

Figura 11 – Cadastro de Informação Profissional.

A comunicação com o backend é feita por meio do serviço Angular `ProfessionalInformationService`, responsável por executar as operações de busca, criação, edição e exclusão dos registros. A Figura 12 apresenta a implementação desse serviço.

```
You, semana passada | 1 author (You)
@Injectable({
  providedIn: 'root',
})
export class ProfessionalInformationService {
  private apiUrl = 'http://localhost:8080/api/informacoes/profissionais';

  constructor(private http: HttpClient) {}

  listarTodos(): Observable<ProfessionalInfo[]> {
    return this.http.get<ProfessionalInfo[]>(this.apiUrl);
  }

  listarPorId(id: string): Observable<ProfessionalInfo> {
    return this.http.get<ProfessionalInfo>(`${this.apiUrl}/editar/${id}`);
  }

  buscarPorEgressoCpf(cpf: string): Observable<ProfessionalInfo[]> {
    return this.http.get<ProfessionalInfo[]>(`${this.apiUrl}/egresso/${cpf}`);
  }

  criar(dto: ProfessionalInfo): Observable<ProfessionalInfo> {
    return this.http.post<ProfessionalInfo>(this.apiUrl, dto);
  }

  atualizar(id: string, dto: ProfessionalInfo): Observable<ProfessionalInfo> {
    return this.http.put<ProfessionalInfo>(`${this.apiUrl}/${id}`, dto);
  }

  excluir(id: string): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

Figura 12 – Serviço Angular `ProfessionalInformationService` responsável pela comunicação com a API.

4.1.5 Depoimentos

A seção de depoimentos permite ao egresso compartilhar relatos sobre sua vivência na UFU. É possível visualizar todos os depoimentos cadastrados, bem como criar, editar ou excluir depoimentos. Essa funcionalidade contribui para o fortalecimento da identidade institucional da universidade, valorizando as experiências dos ex-alunos.

As Figuras 13, 14 e 15 ilustram, respectivamente, a visualização geral dos depoimentos cadastrados, as opções de edição e exclusão, e a tela de cadastro de um novo depoimento. Embora as imagens reflitam apenas a interface do frontend, os dados exibidos já estão integrados ao backend. Ou seja, os depoimentos são persistidos em banco de dados e recuperados por meio de requisições à API.



Figura 13 – Visualização dos Depoimentos.

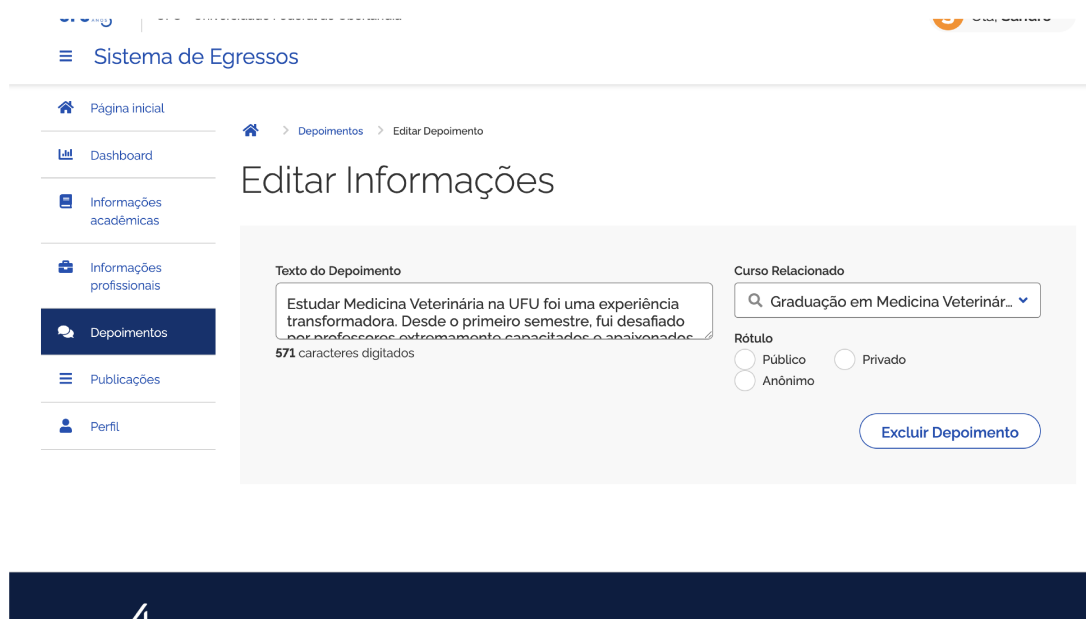


Figura 14 – Edição e Exclusão de Depoimentos.

The screenshot shows a web application titled 'Sistema de Egressos'. On the left is a sidebar menu with options: 'Página inicial', 'Dashboard', 'Informações acadêmicas', 'Informações profissionais', 'Depoimentos' (highlighted), 'Publicações', and 'Perfil'. The main content area is titled 'Cadastrar Informações' and contains a form for a new testimonial. The form has a text input labeled 'Texto do Depoimento' with a character count '0 caracteres digitados'. To the right is a dropdown menu labeled 'Curso Relacionado' with the placeholder 'Selecione o item'. Below these are three radio buttons for 'Rótulo': 'Público', 'Privado', and 'Anônimo'.

Figura 15 – Cadastro de Depoimentos.

Essa integração é viabilizada pelo serviço Angular `TestimonialsService`, responsável por realizar a comunicação entre o frontend e o backend. A Figura 16 apresenta a implementação deste serviço.

```
You, semana passada | 1 author (You)
@Injectable({
  providedIn: 'root',
})
export class TestimonialsService {
  private apiUrl = 'http://localhost:8080/api/depoimentos';

  constructor(private http: HttpClient) {}

  buscarPorId(id: string): Observable<Testimonial> {
    return this.http.get<Testimonial>(`${this.apiUrl}/${id}`);
  }

  criar(depoimentoDTO: Testimonial): Observable<Testimonial> {
    return this.http.post<Testimonial>(this.apiUrl, depoimentoDTO);
  }

  excluir(id: string): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }

  listarPorCpf(cpf: string): Observable<Testimonial[]> {
    return this.http.get<Testimonial[]>(`${this.apiUrl}/egresso/${cpf}`);
  }

  atualizar(id: string, dto: any): Observable<Testimonial> {
    return this.http.put<Testimonial>(`${this.apiUrl}/${id}`, dto);
  }
}
```

Figura 16 – Serviço Angular `TestimonialsService` utilizado para comunicação com o backend.

4.1.6 Publicações

Nesta seção, os egressos podem cadastrar, editar, visualizar e excluir suas publicações acadêmicas e profissionais. O objetivo é destacar a produção intelectual dos ex-alunos, incentivando o compartilhamento de conhecimento e o reconhecimento das contribuições acadêmicas e científicas.

As Figuras 17, 18 e 19 ilustram a interface da seção de publicações, respectivamente com a visualização geral, o formulário de edição e a tela de cadastro. Embora as imagens apresentem somente o frontend, esta funcionalidade já está completamente integrada ao backend. As informações exibidas são carregadas diretamente do banco de dados, e as ações realizadas pelo usuário (como criar, editar ou excluir publicações) são refletidas na base de dados em tempo real.



Figura 17 – Visualização de todas as Publicações.

The screenshot shows the 'Editar Publicação' (Edit Publication) form within the 'Sistema de Egressos' application. The left sidebar contains navigation links: 'Página inicial', 'Dashboard', 'Informações acadêmicas', 'Informações profissionais', 'Depoimentos', 'Publicações' (highlighted), and 'Perfil'. The main content area has a breadcrumb trail 'Publicações > Editar Publicação' and the title 'Editar Publicação'. The form fields are as follows:

- Título da Publicação:** Input field with the value 'Impacto da Nutrição'.
- Autores:** Input field with the value 'Fernanda Costa, Sandro Vieira'. Below it, a note reads: 'Insira os nomes dos autores no formato 'Nome Sobrenome', separados por ponto e vírgula. Mantenha a ordem de autoria original.'
- Ano da Publicação:** Input field with the value '2022'.
- Veículo:** Input field with the value 'Zootecnia Atual'.
- URL da Publicação:** Input field with the value 'https://zootecniaatu...'.
- Curso Relacionado:** Dropdown menu with the selected option 'Graduação em...'.

At the bottom right of the form are two buttons: 'Excluir' and 'Salvar'.

Figura 18 – Edição de Publicação.

The screenshot shows the 'Cadastrar Publicação' (Register Publication) form within the 'Sistema de Egressos' application. The left sidebar is identical to the previous figure, with 'Publicações' highlighted. The main content area has a breadcrumb trail 'Publicações > Nova Publicação' and the title 'Cadastrar Publicação'. The form fields are as follows:

- Título da Publicação:** Empty input field.
- Autores:** Input field with the placeholder text 'Ex: João Pereira; Maria Santos'. Below it, the same note as in Figure 18 is present: 'Insira os nomes dos autores no formato 'Nome Sobrenome', separados por ponto e vírgula. Mantenha a ordem de autoria original.'
- Ano da Publicação:** Empty input field.
- Veículo:** Empty input field.
- URL da Publicação:** Empty input field.
- Curso Relacionado:** Dropdown menu with the placeholder text 'Selecione o item'.

At the bottom right of the form is a 'Salvar' button. A red error message 'Campo obrigatório' (Required field) is displayed below the 'Curso Relacionado' dropdown.

Figura 19 – Cadastro de Publicação.

Essa integração é viabilizada por meio do serviço `Angular PublicationsService`, responsável por realizar todas as operações de comunicação entre o frontend e a API. A implementação desse serviço é apresentada na Figura 20.

```
You, há 3 semanas | 1 author (You)
@Injectable({
  providedIn: 'root',
})
export class PublicationsService {
  private apiUrl = 'http://localhost:8080/api/publicacoes';

  constructor(private http: HttpClient) {}

  buscarPorEgresso(cpf: string): Observable<Publication[]> {
    return this.http.get<Publication[]>(`${this.apiUrl}/egresso/${cpf}`);
  }

  criar(publicacaoDTO: Publication): Observable<Publication> {
    return this.http.post<Publication>(this.apiUrl, publicacaoDTO);
  }

  atualizar(id: string, publicacaoDTO: Publication): Observable<Publication> {
    return this.http.put<Publication>(`${this.apiUrl}/${id}`, publicacaoDTO);
  }

  buscarPorId(id: string): Observable<Publication> {
    return this.http.get<Publication>(`${this.apiUrl}/${id}`);
  }

  excluir(id: string): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

Figura 20 – Serviço Angular PublicationsService utilizado para comunicação com o backend.

4.1.7 Perfil

A seção de perfil apresenta as informações pessoais do usuário, permitindo a visualização e edição de dados como nome completo, telefone e data de nascimento. No entanto, por razões de segurança e para garantir a identificação única de cada egresso no sistema, os campos de CPF e e-mail são bloqueados para edição.

As Figuras 21 e 22 ilustram a visualização da página de perfil, distribuída em duas partes. Essas informações são provenientes diretamente do banco de dados, acessadas por meio da integração com o backend da aplicação.

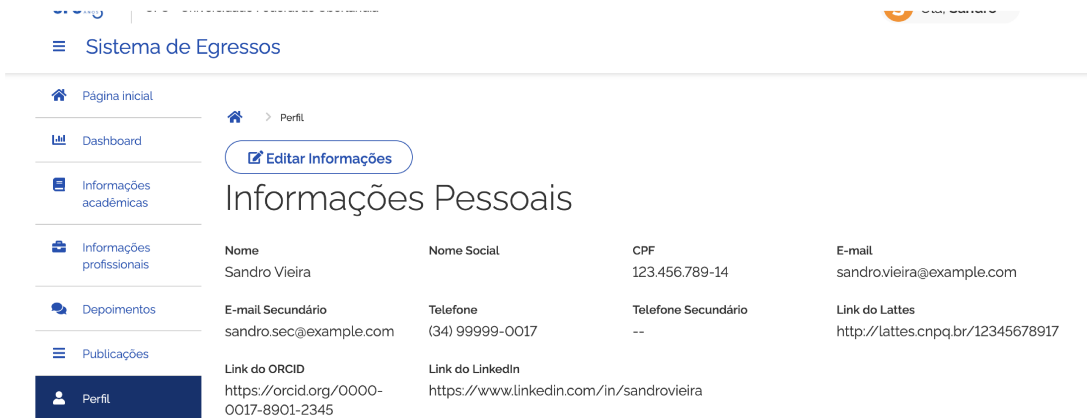


Figura 21 – Visualização do Perfil – Parte I

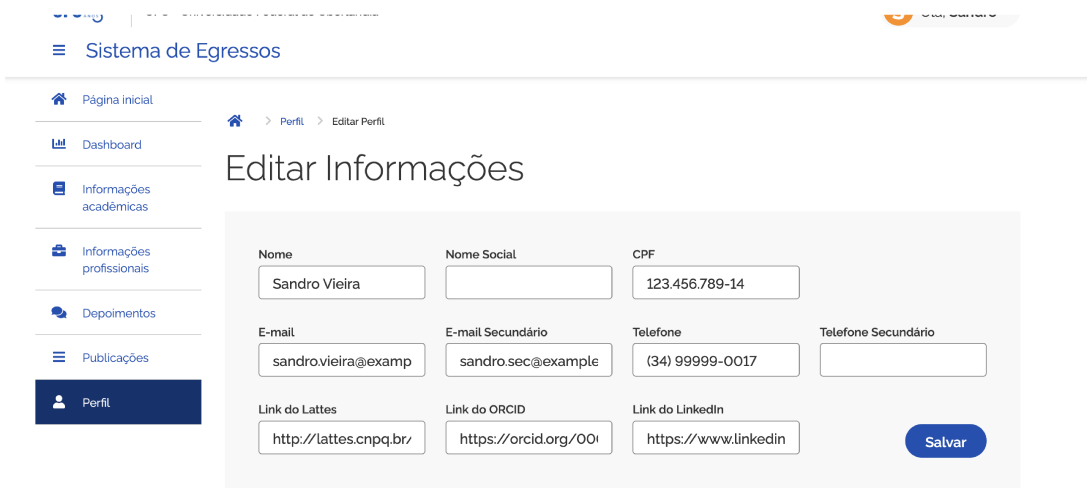


Figura 22 – Visualização do Perfil – Parte II

Essa comunicação é realizada através do serviço Angular `EgressoService`, responsável por buscar e atualizar os dados do egresso. A Figura 23 apresenta a implementação do serviço utilizado para essa finalidade.

```
You, há 4 semanas | 1 author (You)
@Injectable({
  providedIn: 'root',
})
export class EgressoService {
  private apiUrl = 'http://localhost:8080/api/egressos';

  constructor(private http: HttpClient) {}

  buscarTodosEgressos(): Observable<EgressoModel[]> {
    return this.http.get<EgressoModel[]>(this.apiUrl);
  }

  buscarEgressoPorCPF(cpf: string): Observable<EgressoModel> {
    return this.http.get<EgressoModel>(`${this.apiUrl}/${cpf}`);
  }

  atualizarEgresso(cpf: string, egressoDTO: Partial<EgressoModel>): Observable<EgressoModel> {
    return this.http.put<EgressoModel>(`${this.apiUrl}/${cpf}`, egressoDTO);
  }
}
```

Figura 23 – Serviço Angular `EgressoService` responsável pela comunicação com o backend na seção de perfil.

Através desse serviço, o sistema pode buscar os dados do egresso utilizando o CPF como identificador único, bem como atualizar as informações permitidas. Dessa forma, as modificações realizadas pelo usuário são refletidas diretamente no banco de dados, garantindo consistência e integridade dos dados armazenados.

5 Conclusão

O desenvolvimento da plataforma backend para o acompanhamento de egressos da Universidade Federal de Uberlândia (UFU) representa um avanço significativo na consolidação de uma solução tecnológica robusta, segura e alinhada às diretrizes institucionais. Este trabalho deu continuidade à linha de pesquisa iniciada por [Mendes \(2023\)](#), com o levantamento e a modelagem dos requisitos do sistema, e posteriormente ampliada por [Carvalho \(2025\)](#), responsável pela arquitetura e implementação do frontend da aplicação. Juntamente com a estruturação inicial do banco de dados relacional realizada por Paula Mendes, essa base permitiu a construção de um backend funcional e preparado para integrar os diferentes componentes do sistema de forma eficaz.

Ao assumir o foco no backend, este trabalho teve como principal objetivo transformar os requisitos funcionais levantados em implementações concretas, garantindo a comunicação eficiente entre o frontend e a base de dados. A utilização de tecnologias consolidadas como Java, Spring Boot e PostgreSQL proporcionou uma base técnica confiável, enquanto a aplicação de testes unitários assegurou a estabilidade e o correto funcionamento das funcionalidades desenvolvidas.

A gestão do projeto foi conduzida por meio de uma adaptação da Metodologia Ágil Scrum, que possibilitou a organização do trabalho em ciclos iterativos e incrementais, facilitando a adaptação às mudanças de requisitos e promovendo entregas contínuas de valor. Essa abordagem mostrou-se adequada para lidar com as incertezas inerentes ao desenvolvimento de sistemas complexos e para manter a comunicação eficiente entre os membros da equipe.

A arquitetura REST adotada favorece a escalabilidade do sistema e possibilita futuras integrações com plataformas externas, como o Currículo Lattes e o LinkedIn, o que reforça o potencial da plataforma como uma ferramenta estratégica para o mapeamento da trajetória profissional dos egressos. Ao mesmo tempo, a adoção de boas práticas de segurança, como autenticação, controle de acesso e proteção de dados sensíveis, garantiu a conformidade com legislações atuais e futuras, como a Lei Geral de Proteção de Dados (LGPD).

O projeto também dialoga com políticas educacionais vigentes, atendendo às exigências do Ministério da Educação (MEC) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), que destacam o acompanhamento de egressos como um critério relevante na avaliação institucional. O sistema desenvolvido permite à UFU não apenas cumprir essas diretrizes, mas também fortalecer seu vínculo com os ex-alunos, contribuindo para o aprimoramento contínuo dos cursos e para a elaboração de políticas

acadêmicas mais eficazes.

A estrutura modular da aplicação possibilita três frentes principais de evolução:

- **Técnica:** Manutenção e atualização contínua de frameworks, bibliotecas e práticas de desenvolvimento, com foco em segurança e desempenho.
- **Funcional:** Inclusão de novas funcionalidades conforme as demandas institucionais, como dashboards, relatórios avançados e integrações externas.
- **Regulatória:** Adequações às mudanças na legislação e nas políticas de avaliação institucional, garantindo conformidade e relevância do sistema.

Além da contribuição técnica, este trabalho também apresenta uma dimensão acadêmica e social importante. Ao facilitar a coleta, organização e análise de dados sobre os egressos, a plataforma permite à UFU fortalecer sua responsabilidade social, avaliar o impacto de sua formação no mercado de trabalho e promover uma cultura institucional orientada por dados.

Em síntese, a implementação do backend não apenas completou uma etapa essencial do sistema de acompanhamento de egressos da UFU, como também contribuiu para a consolidação de uma base tecnológica sólida que poderá ser expandida e aprimorada continuamente. Os resultados obtidos reforçam a importância do trabalho colaborativo, da definição clara de requisitos e da adoção de boas práticas de engenharia de software para a construção de soluções que gerem impacto positivo na sociedade.

Como trabalhos futuros, destaca-se a possibilidade de aplicar técnicas de inteligência artificial para prever trajetórias profissionais com base nos dados históricos dos egressos, além da análise de impacto dos dados coletados na reformulação curricular. A plataforma aqui desenvolvida configura-se, portanto, como um legado acadêmico e institucional, capaz de evoluir junto com as demandas da universidade e da sociedade.

Referências

APACHE. **Apache Maven Project**. 2025. <<https://maven.apache.org/>>. [Online; accessed 29-outubro-2023]. Citado na página 20.

AWARI. **As Vantagens E Desvantagens Do Scrum: Tudo O Que Você Precisa Saber Sobre Essa Metodologia Ágil**. 2023. <<https://awari.com.br/as-vantagens-e-desvantagens-do-scrum-tudo-o-que-voce-precisa-saber-sobre-essa-metodologia-agil/>>. [Online; accessed 29-outubro-2023]. Citado na página 19.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 3. ed. [S.l.]: Addison-Wesley, 2012. ISBN 978-0321815736. Citado na página 21.

BESSA, A. **Java: o que é, linguagem e um Guia para iniciar na tecnologia**. 2023. <<https://www.alura.com.br/artigos/java?srsId=AfmBOopPFmi3n0UU9UrZnRRLK-ns3GCfKWl5VK18fupjjIIqqr53zYMI>>. [Online; accessed 29-outubro-2023]. Citado 2 vezes nas páginas 11 e 20.

BRASILEIRO, G. **eMAG - Modelo de Acessibilidade em Governo Eletrônico**. 2014. <https://emag.governoeletronico.gov.br/#s2.1>. [Online; accessed 01-outubro-2024]. Citado na página 11.

_____. **Padrões de Interoperabilidade de Governo Eletrônico - ePING**. 2018. <https://eping.governoeletronico.gov.br>. [Online; accessed 01-outubro-2024]. Citado na página 11.

_____. **Padrão Digital de Governo**. 2022. <https://www.gov.br/ds/home>. [Online; accessed 01-outubro-2024]. Citado na página 11.

CAPES. **Aprovada proposta de aprimoramento de avaliação da pós-graduação**. 2018. <https://www.gov.br/capes/pt-br/assuntos/noticias/aprovada-proposta-de-aprimoramento-de-avaliacao-da-pos-graduacao>. [Online; accessed 12-outubro-2024]. Citado na página 10.

CARVALHO, L. A. da S. **Arquitetura Frontend e Interface de um Sistema de Acompanhamento de Egressos para a UFU**. 56 p. Monografia (Graduação em Sistemas de Informação) — Universidade Federal de Uberlândia, Uberlândia, 2025. Disponível em: <<https://repositorio.ufu.br/handle/123456789/45437>>. Citado 5 vezes nas páginas 11, 15, 25, 26 e 43.

CNPQ, C. N. de Desenvolvimento Científico e T. **Plataforma Lattes**. 2025. <<http://lattes.cnpq.br/>>. [Online; accessed 01-março-2025]. Citado na página 26.

CORTES, S. da C. **Um modelo de transações para integração de SGBD a um ambiente de computação móvel**. Tese (Doutorado) — Pontifícia Universidade Católica, Rio de Janeiro, 2004. Orientador: Sérgio Lifschitz. Citado na página 21.

CURTI, J. C. **Análise de segurança em aplicações que utilizam plataformas UNIX e MS-Windows no modelo Cliente/Servidor**. Dissertação (Mestrado) — Universidade Estadual de Campinas (UNICAMP), 2004. Acesso em: 1 mar. 2025.

Disponível em: <<https://www.lasca.ic.unicamp.br/paulo/teses/20040901-MP-Joao.Carlos.Curti-Analise.de.seguranca.em.aplicacoes.que.utilizam.plataformas.UNIX.e.MS.Windows.no.modelo.Cliente.Servidor.pdf>>. Citado na página 21.

FERREIRA, L. A.; LIMA, M. R. **Scrum na Prática: Como aplicar a metodologia ágil em projetos reais**. São Paulo: Novatec, 2021. Citado na página 18.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Ph.D. Thesis) — University of California, Irvine, 2000. Disponível em: <<https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>. Citado na página 17.

FIELDING, R. T.; TAYLOR, R. N.; ERENKRANTZ, J. R.; GORLICK, M. M.; WHITEHEAD, J.; KHARE, R.; OREIZY, P. Reflections on the rest architectural style and "principled design of the modern web architecture. In: **Proceedings of the 2017 11th joint meeting on foundations of software engineering**. New York, NY, USA: ACM, 2017. p. 4–14. Citado na página 22.

GUIMARÃES, M. A. M. **O acompanhamento de egressos como ferramenta de inserção no mercado de trabalho do ponto de vista do setor de estágio e emprego do CEFET/RJ**. Dissertação (Mestrado) — Universidade Federal Fluminense, Curso de Sistemas de Gestão, Departamento de Engenharia de Produção, Niterói, RJ, 2013. Orientador: Mara Telles Salles. Disponível em: <<https://app.uff.br/riuff/handle/1/20507>>. Citado na página 10.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de Informação Gerenciais**. 16. ed. São Paulo: Pearson, 2021. Citado na página 17.

LINKEDIN. **LinkedIn — A Professional Networking Platform**. 2025. <<https://www.linkedin.com/>>. [Online; accessed 01-março-2025]. Citado na página 26.

MEC. **MEC e Inep anunciam melhorias na avaliação da educação superior**. 2023. . [Online; accessed 12-outubro-2024]. Citado na página 10.

MENDES, F. C. **Sistema de acompanhamento de egressos UFU: levantamento e análise de requisitos**. 56 p. Monografia (Graduação em Ciência da Computação) — Universidade Federal de Uberlândia, Uberlândia, 2023. Disponível em: <<https://repositorio.ufu.br/handle/123456789/39896>>. Citado 3 vezes nas páginas 11, 15 e 43.

ORACLE. **Definindo uma Chave Primária**. 2020. <https://docs.oracle.com/cloud/latest/related-docs/OMCEZ/pt_br/FileNewFormsTablePrimaryKeys.htm>. [Online; accessed 01-março-2025]. Citado na página 23.

_____. **Restrições de CHAVE ESTRANGEIRA**. 2020. <https://docs.oracle.com/cd/E17952_01/mysql-5.7-en/create-table-foreign-keys.html>. [Online; accessed 01-março-2025]. Citado na página 23.

POSTGRESQL. **PostgreSQL: The World's Most Advanced Open Source Relational Database**. 2025. <<https://www.postgresql.org/>>. [Online; accessed 01-março-2025]. Citado 2 vezes nas páginas 11 e 24.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**. 9. ed. Porto Alegre: AMGH, 2023. Citado na página 17.

QNAX. **Spring Framework: Tudo o que Você Precisa Saber**. 2024. <<https://qnax.sh/blog/dicas/spring-framework-tudo-o-que-voce-precisa-saber/>>. [Online; accessed 29-outubro-2023]. Citado na página 20.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados-3**. [S.l.]: AMGH Editora, 2008. Citado 2 vezes nas páginas 22 e 23.

REDHAT. **O que é uma API REST?** 2024. . [Online; accessed 01-outubro-2024]. Citado na página 11.

SILVA, M. F. da; ROCHA, T. P. **Gestão Ágil com Scrum: Experiências e aplicações práticas**. [S.l.]: Ciência Moderna, 2022. Citado na página 18.

SOUZA, B. **Conhecendo melhor o Maven**. 2023. <<https://www.alura.com.br/artigos/conhecendomelhormavensrs?tid=AfmBOorPrqndfL39jtZrG6Im5muJLI9SPhFnoVXWfTsiw4fjv088hDAp>>. [Online; accessed 29-outubro-2023]. Citado 2 vezes nas páginas 11 e 20.

SPRING. **Spring Boot: Spring Framework for Rapid Application Development**. 2025. <<https://spring.io/projects/spring-boot>>. [Online; accessed 29-outubro-2023]. Citado na página 11.

SUTHERLAND, J.; SCHWABER, K. **Guia do Scrum: A definição do Scrum**. [S.l.], 2020. Disponível em: <<https://scrumguides.org/scrum-guide.html>>. Citado na página 18.

UFU. **Acesso ao SEI-UFU!** 2024. <<https://ufu.br/sei>>. [Online; accessed 01-março-2025]. Citado na página 26.