UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Erick Cristian de Oliveira Pereira

Sistema de alerta de limites e prazos acadêmicos para discentes de graduação da FACOM

Uberlândia, Brasil

2023

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Erick Cristian de Oliveira Pereira

Sistema de alerta de limites e prazos acadêmicos para discentes de graduação da FACOM

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.

Orientador: Renato Aparecido Pimentel da Silva

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2023

Resumo

Este trabalho aborda a construção de um sistema de alerta para discentes com base em sua situação acadêmica de acordo com as normas de graduação. O objetivo desse sistema é reduzir demandas de reconsideração enviadas para a coordenação da faculdade de computação, enviando e-mails de forma automática para discentes informando-os de sua situação para se atentarem a prazos e limites que precisam ser cumpridos. Serão apresentadas as técnicas de engenharia de software adotadas para projetar o sistema e, as ferramentas adotadas e sua construção, com o resultado final demonstrado por meio da execução do sistema.

Palavras-chave: automação, e-mail, engenharia de software, desenvolvimento de software, sistema, coordenação.

Abstract

This work will address the construction of an alert system for students based on their academic situation in accordance with graduation norms. The purpose of this system is to reduce reconsideration demands sent to the faculty coordination computing, automatically sending e-mails to students informing them of their situation to pay attention to deadlines and limits that need to be met. will be presented the software engineering techniques adopted to design the system and, the tools adopted and their construction, together with the final result demonstrated through running the system.

Key Words: automation, e-mail, software engineering, software development, system, coordination.

Lista de ilustrações

List of abbreviations and acronyms

API	*Application Programming Interface*
CPU	*Central Processing Unit*
CSS	*Cascading Style Sheets*
FACOM	Faculdade de Computação
GB	*Gigabyte*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
MB	*Megabyte*
MB/s	*Megabyte per second*
MHz	*Mega Hertz*
MVC	*Model View Controller*
ORM	*Object-relational mapping*
REST	*Representational State Transfer*
SGBD	Sistema de Gerenciamento de Banco de Dados
SMTP	*Simple Mail Transfer Protocol*
SQL	*Structured Query Language*
SSD	*solid-state drive*
UFU	Universidade Federal de Uberlândia
UML	*Unified Modeling Language*

Sumário

1 Introdução

A FACOM (Faculdade de Computação) atualmente precisa direcionar recursos para medidas corretivas em casos onde discentes falham no cumprimento de prazos previstos em normas, cenário comum onde discentes que não se atentaram aos prazos de suas atividades de graduação perdem aproveitamento dos componentes do curso e acabam recorrendo à coordenação para solucionar essa questão. Essa situação gera volume recorrente de solicitações de reconsideração.

O problema com o cumprimento dos prazos impacta tanto os discentes quanto às coordenações, e se motivando nessa situação o serviço de envio de e-mail a ser implementado neste trabalho de conclusão de curso propõe abordar esse cenário com medidas automáticas de acompanhar e notificar todos os alunos sobre seus respectivos prazos para o cumprimento dos componentes da graduação.

O projeto pedagógico da graduação na FACOM trabalha submetendo os discentes a múltiplos componentes curriculares que devem ser cumpridos para o discente ser efetivamente graduado. Em alguns componentes, e até mesmo na própria graduação, há prazos e limites que o discente deve se atentar e cumprir para evitar complicações neste processo.

1.1 Objetivos

O objetivo desse trabalho é implementar uma solução automatizada para identificar os prazos e eventos e disparar um e-mail para os discentes alertando-os para se atentarem a seu cumprimento para reduzir as demandas de reconsideração subsequentes direcionadas à coordenação dos cursos da FACOM, e tornar essas informações mais acessíveis para os discentes.

Para definição específica do objetivo deste trabalho de conclusão de curso, são definidos os seguintes objetivos específicos que compõem o objetivo geral:

- Identificar prazos e situações previstos em normas da graduação que devem ser cumpridos pelos discentes e que possam ser notificados de maneira automatizada por e-mail.
- Aplicar o uso de técnicas de engenharia de software para aperfeiçoar o desenvolvimento do projeto, adotando práticas de documentação como: Especificação de requisitos funcionais e não funcionais, diagrama de entidades, especificação de requi-

sitos no formato de caso de uso, diagrama de caso de uso e diagrama de atividades, adotando padrões da linguagem UML. (PRESSMAN, 2016).

- Implementar um robô/automação para verificar a situação de cada aluno ao menos uma vez por dia e disparar e-mails com informações necessárias para mantê-lo atualizado de acordo com os prazos e situações que o sistema identificou, onde o aluno deve se atentar para não ter perda de aproveitamento.
- Implementar uma aplicação web (web site) que dispõe de funcionalidades para facilitar o gerenciamento do serviço de disparo de e-mail, cadastrando parâmetros que serão utilizados pelo robô/automação na hora de verificar a situação dos alunos e disparar o e-mail.
- Implementar uma base de dados que persistirá as informações de configuração e parametrização do serviço.
- Implementar uma web API, aplicação responsável por intermediar a comunicação de acesso a base de dados por meio da automação e da aplicação web. Essa web API atenderá a requisições realizadas pela automação e aplicação web para consultar ou atualizar as informações da base de dados.

1.2 Justificativa

Na FACOM, pedidos de reconsideração por falta de cumprimento com prazos previstos nas normas são demandas constantes. Uma situação custosa tanto para os discentes quanto para a coordenação dos cursos da FACOM.

Investir em uma estratégia para facilitar o acesso a essas informações permitiria manter o corpo discente atualizado quanto a suas pendências e prazos, proporcionando recursos para um melhor gerenciamento das atividades da graduação e evitar complicações como perda de aproveitamento pelo descumprimento com as normas dos cursos da FACOM.

Com a implementação do serviço automático de disparo de e-mail que esse trabalho propõe, os discentes terão sua situação analisada de forma individual. Serão fornecidas informações precisas e atualizadas que auxiliarão os discentes a acompanharem suas pendências e prazos com uma estratégia preventiva de informar ao corpo discente. Assim, será possível reduzir as demandas de reconsiderações que chegam às coordenações dos cursos, o que gera uma burocracia que poderia ser evitada e que consome recursos das graduações.

Uma questão que o serviço também se compromete a tratar são os limites presentes nos meios oficiais de divulgação de informação dos cursos. Esses meios normalmente

tratam de divulgar apenas prazos previstos no calendário acadêmico, cabendo ao aluno atentar-se a todos os prazos referentes à sua situação individual, consultando a documentação de normas, a coordenação, os docentes, e mesmo outros discentes. Trabalho manual este que se submete à falha humana e pode acabar deixando algum prazo crítico ignorado. Uma situação que pode ser mitigada pelo serviço que consegue concentrar informações importantes e levá-las até a caixa de entrada do discente.

1.3 Organização

Este trabalho está organizado da seguinte forma: No capítulo Referencial Teórico serão abordados os conceitos e tecnologias fundamentais para o desenvolvimento deste trabalho e, trabalhos correlatos. O capítulo Desenvolvimento mostra de forma detalhada a organização, características e funcionamento de cada componente do sistema. O capítulo de Apresentação ilustra o funcionamento do sistema aos olhos do usuário e, apresenta informações sobre o desempenho computacional do sistema de alerta. No capítulo de Conclusão, são revelados os ganhos e resultados obtidos ao desenvolver o software para cumprir os objetivos deste trabalho.

2 Referencial Teórico

Neste capítulo serão apresentados os fundamentos que constituem a estrutura do projeto deste trabalho em todas as suas fases, como: análise, projeto, desenvolvimento e implementação.

Serão detalhadas tecnologias adotadas na implementação, metodologias e padrões de projeto que guiaram o desenvolvimento do projeto e especificações de prazos previstos em normas da FACOM que irão determinar a dinâmica de execução do sistema.

2.1 Normas

Aqui serão apresentadas normas que determinam prazos e condições previstos na documentação de normas de estágio e normas oficiais para conclusão de atividades de graduação na FACOM.

2.1.1 Estágio Obrigatório

(COLEGIADO, 2022). Art. 13 – Poderão realizar o estágio obrigatório os alunos do bacharelado em Sistemas de Informação que tenham concluído com aproveitamento 1.200 (mil e duzentas) horas-aula.

2.1.2 Estágio Não Obrigatório

(COLEGIADO, 2022). Art. 14 – O estágio curricular não-obrigatório será autorizado somente para os alunos que tenham sido aprovados em todas as disciplinas do primeiro e segundo períodos e que possuam um Coeficiente de Rendimento Acadêmico (CRA) mínimo de 60.

2.1.3 Dilação de Prazo

(GRADUAÇÃO, 2022). Art. 174. § 1- O estudante poderá solicitar dilação de prazo antes de transcorridos 3/4 do período letivo.

2.1.4 Trancamento Geral

(GRADUAÇÃO, 2022): Art. 102. O trancamento geral de matrícula será efetuado por, no máximo, 2 (dois) semestres letivos, consecutivos ou não, para os cursos semestrais, ou por 1 (um) ano, para os cursos anuais.

2.1.5 Trancamento Parcial

(GRADUAÇÃO, 2022). Art. 92. § 5- A matrícula nos componentes curriculares de que trata este artigo será automaticamente cancelada caso o estudante solicite trancamento parcial de matrícula e fique com carga horária inferior a 120 (cento e vinte) horas semestrais em componentes curriculares no seu curso de origem.

2.2 Engenharia de software

A engenharia de software é uma área que define métodos e ferramentas para aumentar as chances de sucesso no desenvolvimento de um software, que em alguns casos apresenta nível considerável de complexidade.

As técnicas de engenharia de software são construídas com base em experiências de acertos e erros mais recorrentes e significativos na criação de softwares com informações coletadas da experiência de grandes fábricas de softwares e engenheiros de software do mercado. Essas informações são analisadas a fim de prever padrões e elaborar técnicas e ferramentas visando antecipar problemas e solucioná-los antes de ocorrerem e guiar a construção do software a fim de o sucesso de seu objetivo de maneira sustentável em todas as fases de desenvolvimento (projeto, desenvolvimento, implantação e sustentação).

A Unified Modeling Language (UML) (PRESSMAN, 2016) é um padrão de linguagem para definir métodos de projeção visuais das estruturas que constituem os elementos do software, por meio de diagramas. Possui grande peso na área de engenharia de software apoiando na elaboração dos projetos de software. A diagramação permite melhor organização e abre visão para adotar melhores práticas e abordagens antes de investir em implementação.

NA figura 1 tem-se o exemplo de diagrama de caso de uso, uma das técnicas de diagramação da UML.

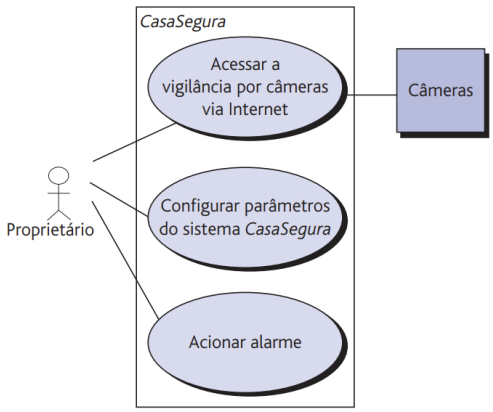

Figura 1 – Diagrama de caso de uso (PRESSMAN, 2016).

2.3 .Net Core

o .Net (NET, 2022) é um ambiente da Microsoft que dispõe de recursos para programar e executar softwares que adotem o uso da plataforma. Possui compatibilidade com sistemas operacionais Windows, sistemas operacionais baseados em Linux e macOS. Inclusive já faz parte das distribuições dos sistemas operacionais Windows mais modernos como o Windows 7, 10, 11 e Windows Server. Alguns recursos que a plataforma dispõe são: gerenciamento de memória, bibliotecas, estruturas de desenvolvimento e compatibilidade entre plataformas.

2.4 C#

O C# (MICROSOFT, 2022) é uma linguagem de programação multiplataforma e multiparadigma que segue como principal linguagem de programação para o *back-end* de aplicações desenvolvidas no ambiente .Net. Se trata de uma ferramenta de programação simples, de fácil aprendizado e ao mesmo tempo poderosa. A ferramenta possui peso no mercado e já é reconhecida como uma das linguagens de programação mundialmente mais utilizadas.

2.5 Banco de Dados - SQL Server

O SQL Server (SQLSERVER, 2022) é um Sistema de Gerenciamento de Banco de Dados (SGBD) mantido pela Microsoft. Sua premissa é dar suporte a outros sistemas dispondo de uma estrutura de persistência poderosa para dados em alta escala, dispondo como meio de consulta a linguagem de pesquisa declarativa SQL. Um banco de dados implementado em SQL Server armazena dados de maneira estruturada em tabelas que organizam as informações em linhas e colunas. Assim uma tabela representa uma entidade, uma linha representa um conjunto de dados referentes a uma unidade de informação e as colunas organizam os tipos de dados de uma unidade de informação. As tabelas podem ter informações estruturadas de forma que se relacionem com informações de outras tabelas.

O SQL Server já é um SGBD de peso no mercado, apresentando a proposta de alta qualidade no gerenciamento de dados por um preço competitivo. A tecnologia já está disponível como serviço em *clouds* de destaque, como Google Cloud e Azure.

2.6 API REST

Uma interface de programação de aplicação (API - *Application Programming Interface*) (MICROSOFT, 2023) se trata de uma ferramenta cuja finalidade é fornecer um meio de comunicação e acesso para diferentes tipos de aplicações implementados de diferentes formas e que se comunicam em linguagens diferentes.

A abreviatura REST se refere a *Representational State Transfer* (Transferência de Estado Representacional), uma arquitetura que define um padrão de implementação para aplicações web. Comumente adotando o protocolo HTTP como padrão, mas não sendo específico para o mesmo. A arquitetura define um padrão de uso para o HTTP para estabelecer comunicação entre aplicações.

Uma api RESTfull é uma interface que faz sua comunicação adotando a arquitetura REST. Utilizando o protocolo HTTP para se comunicarem seguindo padrões que definem a comunicação por meio de métodos de requisições como GET, POST, PUT, PATCH e DELETE.

2.7 HTML

A HyperText Markup Language (HTML) (W3C, 2022a) é uma linguagem que define a estrutura básica de representação de informações na web, criada por Tim Berners-Lee, desenvolvida e mantida pela World Wide Web Consortium (W3C). Essa linguagem permite a renderização de diferentes tipos de mídias como textos, vídeos e imagens. O HTML já é reconhecido como base da estrutura web atual.

A linguagem organiza as informações como elementos que representam alguma informação como textos, imagens, vídeos, caixas de texto, botões, etc. Os elementos são declarados por meio de *tags* que definem o início e o fim de uma informação, com sua localização em uma página web. As *tags* podem ser customizadas por meios de atributos que podem ser explorados por outras linguagens como CSS e Javascript para adicionar estilo visual e eventos dinâmicos na página web.

2.8 CSS

A Cascading Style Sheets (CSS) (W3C, 2022b) é uma linguagem utilizada para estilizar páginas web, projetada para complementar o HTML, customizando o visual de páginas web para deixá-las mais atraentes ao aplicar estilos visuais como por exemplo: Atribuir cores, formas e posicionamento customizáveis a fim apresentar as páginas com um visual mais moderno. A linguagem é implementada por meio de declarações de classes, seletores e atributos que definem o comportamento visual dos elementos HTML.

O CSS permite definir em qual momento uma Tag HTML deve assumir um estilo, assim como o nível de especificidade e de profundidade de efeito no aninhamento de elementos HTML. A alteração do estilo do HTML por meio do CSS pode ser implementada de múltiplas maneiras, como por exemplo:

Em linha diretamente dentro do elemento HTML por meio do atributo style.

Folha de estilos interna. A definição do estilo utilizado no HTML é inserida dentro do próprio arquivo HTML por meio da tag <style> com o objetivo de aplicar o estilo apenas no próprio arquivo HTML.

Folha de estilos externa. Um arquivo é dedicado exclusivamente para a declaração da configuração de estilo, podendo essas definições serem acessadas por múltiplos arquivos HTML por meio de uma referência utilizando o elemento <link>.

2.9 Javascript

Javascript (MDN, 2023) é uma linguagem de programação versátil e leve, de abordagem multi paradigma. É utilizada tanto para desenvolvimento do lado do cliente tornando páginas web dinâmicas, quanto do lado do servidor para implementar o *back-end* em plataformas como node e apache.

2.10 Bootstrap

O Bootstrap (TEAM, 2022) é uma framework de código aberto especializada em CSS que dispõe de um vasto acervo de implementações em html, css e javascript para estilos e animações de uma página web de uma maneira completa e moderna.

A framework dá suporte para a programação do *front-end* de aplicações web de maneira prática e ágil, resolvendo também o desafio de manter uma aplicação responsiva onde uma implementação se adapta bem a múltiplas plataformas, minimizando a necessidade de ter que programar pensando em cada plataforma individualmente.

A framework proporciona tais benefícios por dispor de muitas estruturas de *front-end* semiprontas que podem ser customizadas ao implantar em um *front-end* de maneira bem simples, flexível e econômica.

2.11 Model View Controller (MVC)

O MVC é um padrão de projeto de software construído com base na experiência do mundo de desenvolvimento de software que identificou um problema recorrente envolvendo complexidade na evolução e manutenção de aplicações web onde usuários estão em constante interação com dados por meio de interfaces gráficas.

Para esse problema evidente, o MVC trás uma solução em forma de organização do sistema a fim de minimizar o uso de dependências ao adotar um agrupamento de implementação em camadas com responsabilidades específicas.

O padrão foi formulado a princípio para apoiar o desenvolvimento de softwares que possuem uma interface gráfica de usuário, mas sua organização também pode proporcionar benefícios para outros modelos de aplicações como web APIs ao organizar as responsabilidades em camadas para controlar o acesso aos dados de uma maneira que promova evolução flexível. O padrão assume uma organização em 3 camadas: Visão, Modelo e Controlador.

"O componente Modelo gerencia o sistema de dados e as operações associadas a esses dados. O componente Visão define e gerencia como os dados são apresentados ao usuário. O componente Controlador gerencia a interação do usuário (por exemplo, teclas, cliques do mouse etc.) e passa essas interações para a Visão e o Modelo."(SOMMERVILLE, 2011).

2.12 Trabalhos Relacionados

O trabalho de conclusão de curso de alerta de prazos para alunos de pós-graduação proposto por Scandiffio (2017) apresenta um objetivo técnico similar, com a distinção de focar apenas em notificar os discentes de pós-graduação. No Sistema de controle de prazos para os alunos de pós-graduação, foram levantadas ótimas documentações adotando técnicas de engenharia de software que apresentaram um ótimo resultado ao modelar a implementação, objetivo que também é compartilhado neste trabalho. A definição de ferramentas e metodologias foi explorada com afinco, apresentando um resultado positivo no desenvolvimento do projeto e deixando um guia claro de ferramentas e métodos que poderão ser utilizadas para atingir o objetivo da alerta de prazos.

A fim de contribuir com um objetivo próximo ao do trabalho do TCC de sistema de controle de prazos para os alunos de pós-graduação, este trabalho pretende utilizar de conceitos uma arquitetura cliente-servidor e, estruturar a automação de disparo de e-mails como um componente/aplicação distinto do componente web com a finalidade de explorar essa funcionalidade, permitindo mais autonomia e eficiência com sua modularização como um componente individual, permitindo mais flexibilidade para escolher qual

metodologia/tecnologia usar no componente e também nas opções de agendamento de execução do disparo automático de e-mails.

O trabalho de Sistema Online para Distribuição de Disciplinas proposto por Locatelli (2015) apresenta solução em requisitos de software para automatizar a forma como a FACOM organiza entre os docentes as questões como disciplinas, preferência para lecionar disciplinas, horários e turmas. Uma estratégia em comum com o sistema de alerta de limites e prazos que busca usar de técnicas de desenvolvimento de software para criar um sistema visando melhorar um processo da UFU.

O trabalho de aplicação web para gestão de estoque de um laboratório de pesquisa proposto por Silva (2021) apresenta um protótipo de aplicação web para gestão de estoque de um laboratório de pesquisa da Universidade Federal de Uberlândia. Uma publicação de um Trabalho de Conclusão de Curso com objetivo de reduzir tempo e esforço com gerenciamento de estoque de um laboratório de pesquisa apresentou um tema que em algum momento compartilha de uma mesma motivação que o sistema de alerta de prazos deste trabalho de conclusão de curso. A motivação em comum é investir em aplicações web para reduzir trabalho operacional e redirecionar o tempo para novas áreas de interesse, algo que adota uma parte considerável da tecnologia implementada no sistema de disparo de e-mail. Algo que coopera para a aceitação da abordagem proposta para o disparo de e-mail ao implementar tecnologias similares.

Um ponto interessante seria essa proposta do sistema de gestão do laboratório analisar possíveis oportunidades para transformar atividades manuais ou digitais em rotinas automáticas, como o disparo automático de e-mails de alerta que este se propõe, reduzindo ainda mais trabalho manual ou repetitivo como cadastros, envio de e-mails e processamento de dados. Vale também aproveitar a premissa de automatizar atividades e dar oportunidade para novas tecnologias de reconhecimento de padrões por meio de imagens para substituir um observador humano.

3 Desenvolvimento

Este capítulo contempla os detalhes de toda a arquitetura da aplicação. Serão apresentadas metodologias adotadas para a elaboração do projeto, detalhes técnicos de cada componente do sistema e também como todos os componentes trabalham a fim de realizar o propósito pelo qual o sistema foi projetado e desenvolvido.

3.1 Engenharia de Software

Com o objetivo de lidar com a complexidade de construir um sistema e maximizar suas chances de sucesso ao atingir os objetivos deste trabalho de conclusão de curso, foram adotadas técnicas de engenharia de software presentes em Pressman (2016) como base para a construção da informação a ser utilizada na elaboração do projeto. A seguir, serão apresentadas as implementações de técnicas adotadas para reunir e processar a informação que constitui o escopo do projeto, e que será utilizada como entrada para documentação do projeto, adotando as ferramentas da linguagem UML para formalizar a documentação que guiará o desenvolvimento.

- Levantamento de requisitos

  O levantamento de requisitos se trata da primeira atividade de engenharia de software aplicada no projeto deste trabalho. Nesta atividade foram reunidos os stakeholders do projeto, sendo eles: Coordenação dos cursos da FACOM, professor orientador do trabalho e o aluno responsável pela elaboração deste trabalho de conclusão de curso. Por meio de reuniões capturamos as necessidades preliminares que usamos no levantamento de requisitos que compõem o documento de levantamento de requisitos.

Na tabela 1 está representado o documento de requisitos, formalizando os requisitos funcionais e não funcionais com base no que foi definido com todas as partes interessadas.

Nome	**Descrição**	**Natureza**
Cadastro de Alertas	Sistema deve dispor de ferramentas que permitam ao usuário gerenciar regras de alerta de prazos.	Funcional
Autenticar Usuário.	Sistema deve dispor de uma autenticação para acessar a plataforma.	Funcional
Cadastro de Usuário.	Sistema deve dispor de um método de cadastrar novos usuários.	Funcional
Ativar Usuário	Quando um novo usuário é criado, recebe um e-mail de ativação. Ao acessar o e-mail e clicar no *link* de ativação sua conta deve ser habilitada para uso.	Funcional
Disparar E-mails	Identificar necessidade de alerta e disparar um alerta por e-mail para os discentes. O alerta deve conscientizar o discente da situação.	Funcional
Visualizar Regras	Visualizar regras que são validadas e disparam alertas.	Funcional
Alto volume de dados	O sistema deve iniciar e concluir as atividades no mesmo dia, isso é, varrer a base de dados, validar regras para cada discente para identificar caso de alerta e dispara um e-mail para todos os discentes da base.	Não Funcional
Rotina	O sistema deve realizar a atividade de disparo de alerta de e-mails diariamente.	Não Funcional
Redundância	Só pode haver 1 disparo de e-mail para uma situação de alerta identificada pelo sistema, para o sistema não agir como spam para o usuário final.	Não Funcional
Usuário	A interface web de cadastro deve ser ágil para evitar passar de 6 segundos por interação do usuário.	Não Funcional
Responsividade	A interface web deve dispor de responsividade para se adaptar a dispositivos com telas de diferentes dimensões.	Não Funcional

Tabela 1 – Tabela com requisitos funcionais e não funcionais.

- Documentação de caso de uso

  Após a construção preliminar de requisitos ao reunir os principais no documento de requisitos. Essas informações foram utilizadas como entrada para o processo de construção de cenários / caso de usos do sistema. Assim foi adotada a linguagem UML de documentação de caso de uso para especificar de maneira técnica, materializando os requisitos nas seguintes funcionalidades técnicas:

  – Caso de Uso: [001] Disparar alerta de e-mail

    Ator primário: Automação de alerta.

Meta no contexto:

Executar uma rotina que consulta a base de dados com informações dos discentes matriculados nos cursos da FACOM e também identifica padrões de alerta especificados nos parâmetros da automação para assim disparar um e-mail alertando o discente sobre a situação identificada.

Precondições:

A automação depende dos parâmetros pré-configurados para identificar os eventos. A automação deve ter acesso a uma base de dados atualizada, dispondo dados da situação acadêmica do discente.

Disparador:

Rotina agendada para executar a aplicação de automação.

Cenário:

1. Agendador de tarefas: Verifica constantemente se data e hora atual estão configuradas para executar a automação de alerta.
2. Agendador de tarefas: Executa a automação de alerta na data e hora agendada para o início dessa atividade.
3. Automação: Busca dados de alunos no banco de dados que dispõe os dados dos discentes.
4. Percorre toda a lista de alunos matriculados, verificando casos de necessidade de disparo de e-mail de alerta com base nas regras/parâmetros pré-configurados para a automação de alerta.
5. Dispara um e-mail para discentes, apresentando no corpo no e-mail uma mensagem clara sobre a situação em que o discente se encontra para poder tomar as medidas necessárias antes de qualquer complicação.

Exceções:

1. Nos dias que não forem explicitados pela configuração da rotina do agendador de tarefas, não haverá execução da automação.
2. Caso nenhuma situação seja identificada ao varrer a base de discentes, a automação não irá disparar nenhum e-mail.

A figura 2 apresenta o diagrama para este caso de uso.

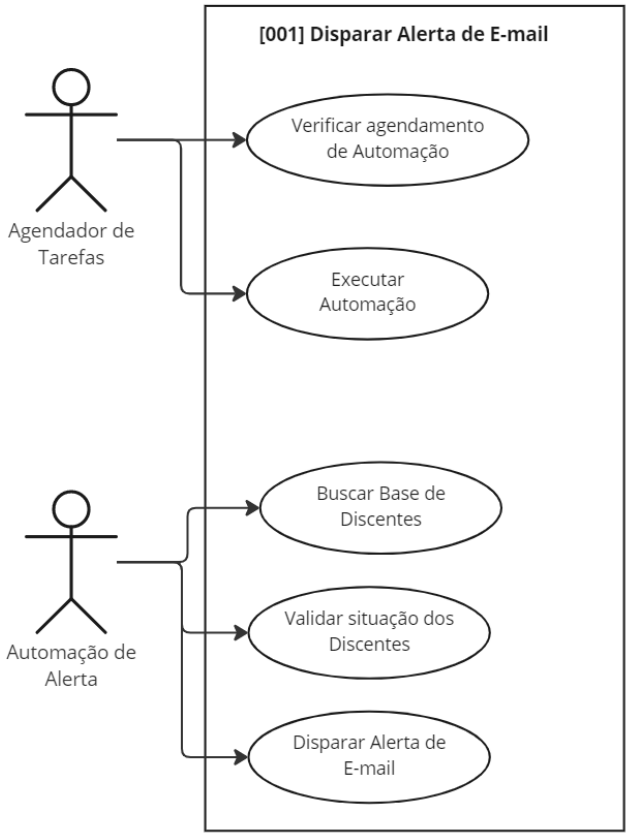

Figura 2 – Diagrama do caso de uso: Disparar alerta de e-mail

– Caso de Uso: [002] Autenticar usuário
Ator primário: Usuário.

Meta no contexto:
Usuário abre a aplicação web de gestão do sistema de alerta no navegador, onde será apresentada a tela de login. Em seguida o usuário entra com suas credenciais e sistema valida as credenciais. Caso sejam válidas o usuário obtém acesso à aplicação.

Precondições:
Usuário deve ter acesso à rede onde a aplicação web encontra-se disponível.
A aplicação web deve estar hospedada em um servidor e sendo disponibilizada na rede.

Disparador:

Usuário entra com a URL da aplicação em seu *browser*.

Cenário:

1. Usuário: Buscar a aplicação entrando com sua URL no *browser*, obtendo acesso à tela de login do sistema.
2. Usuário: Entrar com suas credenciais de acesso nos campos da tela de login.
3. Sistema: Verifica se as credenciais são válidas. Em caso de sucesso direciona o usuário para a página principal do sistema.

Exceções:

1. Sistema: Caso credenciais de usuário sejam inválidas, informar ao usuário e manter a tela de login.

A figura 3 apresenta o diagrama para este caso de uso.

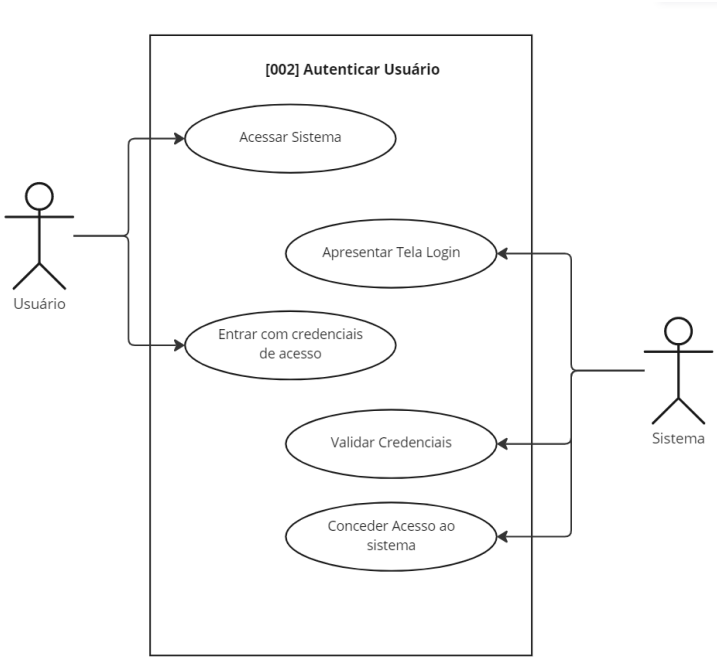

Figura 3 – Diagrama para caso de uso: Autenticar usuário

– Caso de Uso: [003] Visualizar regras

Ator primário: Usuário.

Meta no contexto:

Usuário visualiza no sistema o quadro de regras que estão em vigor validando a situação de alunos e disparando e-mails. Esse quadro apresenta as principais

informações que ajudam o usuário a identificar a natureza da regra.

Precondições:

Usuário deve ter acesso à rede onde a aplicação web encontra-se disponível. A aplicação web deve estar hospedada em um servidor, sendo disponibilizada na rede. Usuário deve estar autenticado no sistema.

Disparador:

Usuário acessa a listagem de regras em vigor.

Cenário:

1. Usuário: Autenticar na aplicação, através do login.
2. Sistema: Conceder acesso ao usuário.
3. Usuário: Selecionar regras em vigor.
4. Sistema: Apresentar quadro de regras ao usuário.

Exceções:

1. Usuário entra com credenciais inválidas e acesso à aplicação não é concedido.

A figura 4 apresenta o diagrama para este caso de uso.

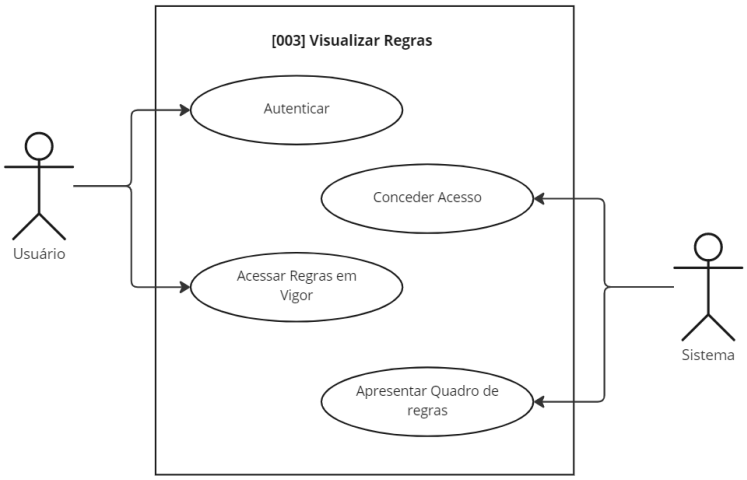

Figura 4 – Diagrama para caso de uso: Visualizar regras

– Caso de Uso: [004] Gerenciar regras

Ator primário: Usuário.

Meta no contexto:

Usuário deve acessar informações sobre a regra que selecionar e ter o poder de alterar os parâmetros dessa regra ou até mesmo desabilitá-la.

Precondições:

Usuário deve ter acesso à rede onde a aplicação web encontra-se disponível. A aplicação web deve estar hospedada em um servidor, sendo disponibilizada na rede. Usuário deve estar autenticado no sistema. Sistema deve possuir alguma regra configurada.

Disparador: Usuário acessa informações da regra.

Cenário:

1. Usuário: Autenticar no sistema.
2. Sistema: Concede acesso ao usuário.
3. Usuário: Seleciona visualizar o quadro de regras.
4. Sistema: Apresenta quadro de regras em vigor.
5. Usuário: Seleciona uma regra para visualizar detalhes.
6. Sistema: Apresenta ao usuário um formulário com dados da regra.
7. Usuário: Visualiza dados da regra, modifica parâmetros e salva o formulário.
8. Sistema: Valida modificações na regra.
9. Sistema: Salva modificações e retorna ao quadro de regras.

Exceções:

1. Se as modificações feitas pelo usuário estiverem com erro ou ferirem alguma regra, sistema não salva o formulário e mantém o formulário aberto.

A figura 5 apresenta o diagrama para este caso de uso.

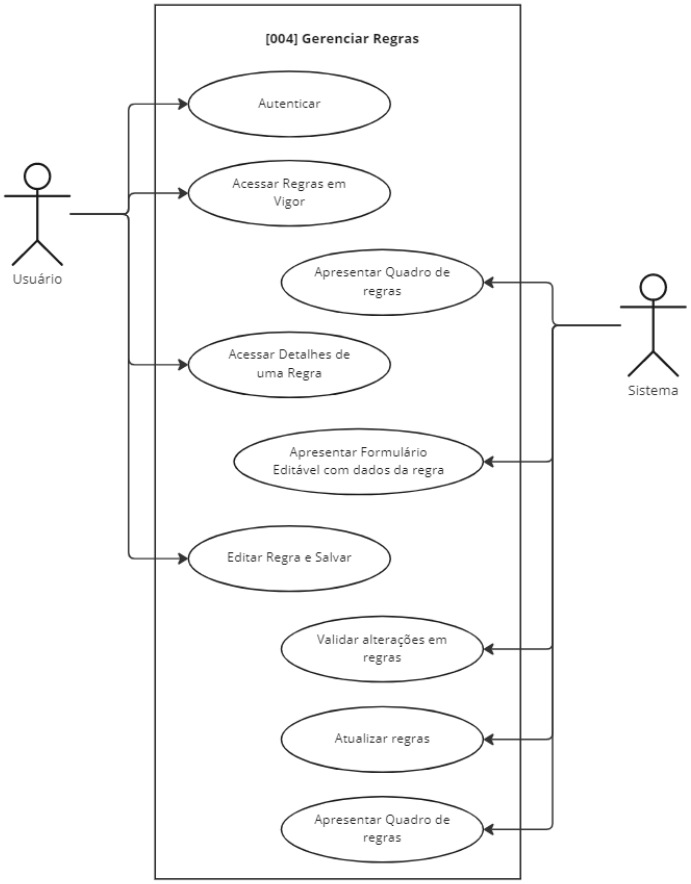

Figura 5 – Diagrama para caso de uso: Gerenciar regras

– Caso de Uso: [005] Cadastrar usuário

  Ator primário: Usuário.

  Meta no contexto:

  Apenas um usuário do sistema pode criar outro usuário para conceder acesso ao sistema de gerenciamento de regras. Para o caso de haver necessidade de mais de uma pessoa gerenciar as regras de alerta no sistema.

  Precondições:

  Usuário deve ter acesso à rede onde a aplicação web encontra-se disponível. A aplicação web deve estar hospedada em um servidor, sendo disponibilizada na rede. Usuário deve estar autenticado no sistema.

Disparador: Usuário acessa funcionalidade de criar usuário. Cenário:

1. Usuário: Autenticar no sistema.
2. Sistema: Concede acesso ao sistema.
3. Usuário: Seleciona criar um usuário.
4. Sistema: Apresenta formulário para cadastro do novo usuário.
5. Usuário: Preenche formulário e salva.
6. Sistema: Valida se dados do novo usuário são válidos.
7. Sistema: Envia e-mail para novo usuário com *link* para ativação do usuário.

Exceções:

1. No caso de dados inválidos o sistema não criará um usuário e, manterá a o formulário de criar usuário.

A figura 6 apresenta o diagrama para este caso de uso.

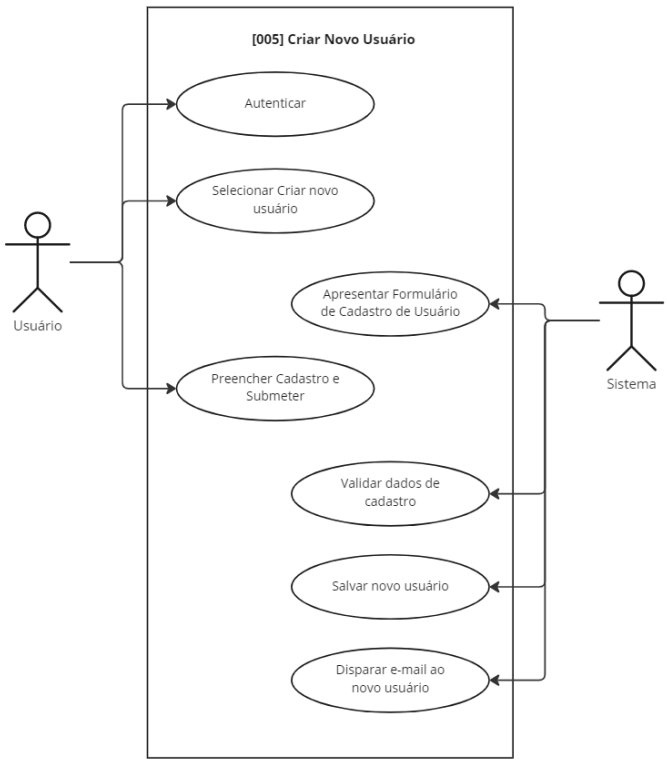

Figura 6 – Diagrama para caso de uso: Cadastrar usuário

- Diagrama de Classes

  Após o levantamento de requisitos e a modelagem de funcionalidades, os dados e informações projetados para o sistema podem ser organizados de modo que representem entidades da vida real que serão manipuladas no sistema. Assim foram identificadas as principais entidades que foram formalmente representadas por uma adaptação do diagrama de classes apresentado em Sommerville (2011) como um modelo semântico de dados para auxiliar na implementação das entidades do sistema. A partir desse modelo foi criada a estrutura do banco de dados do sistema.

  O diagrama de classes da figura 7 apresenta as entidades identificadas como próprias do sistema de alerta de prazos e, que serão necessárias sem seu sua implementação.

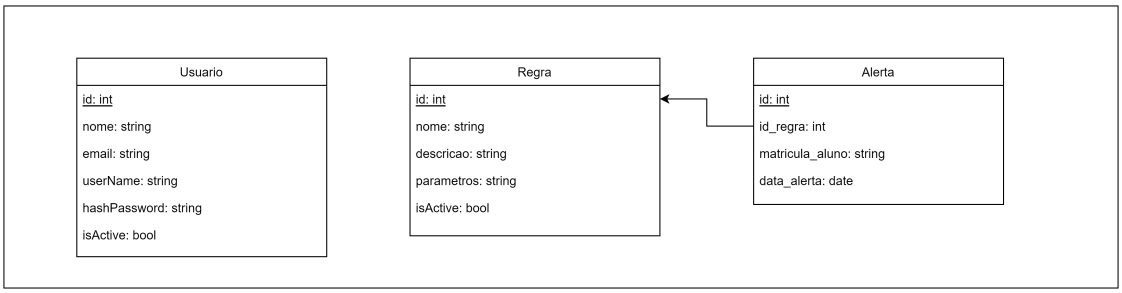

Figura 7 – Diagrama de classes do sistema

O diagrama de classes da figura 8 é fruto da estratégia de representar a estrutura presente nos sistemas da FACOM para controle da graduação. Essa estrutura contém informações referentes a atividades do sistema de graduação que formam o domínio de entidades da faculdade necessários para alimentar a execução do sistema deste trabalho.

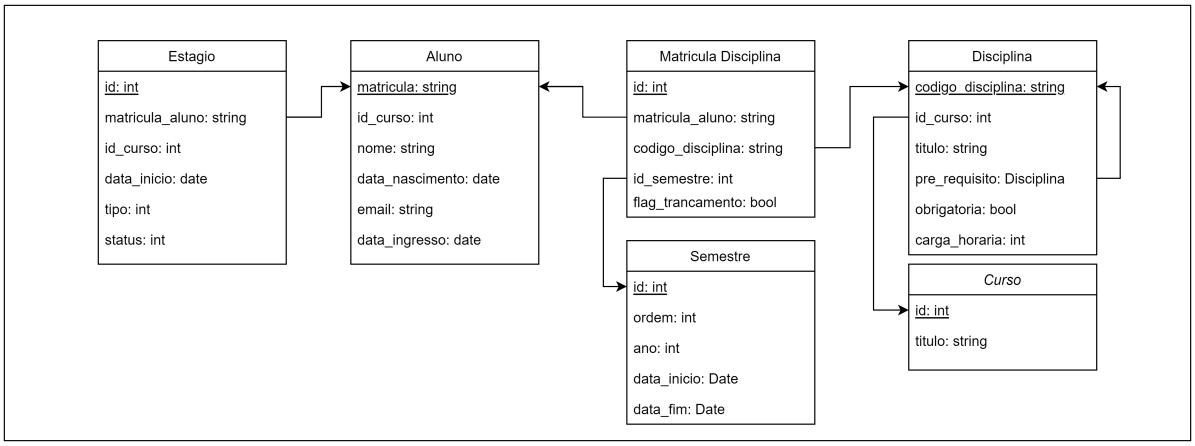

Figura 8 – Diagrama de classes representando a estrutura da FACOM

- Diagrama de atividades

  A linguagem UML dispõe de uma ferramenta de diagramação de atividades para complementar a documentação de caso de uso, representando um caso de uso de uma maneira visual que enfatiza a lógica da funcionalidade e o fluxo de dados. Essa atividade produziu uma interpretação mais próxima da implementação técnica, proporcionando uma visão complementar para identificar integrações do sistema com outros componentes.

  Logo em seguida está a figura 9 ilustrando o diagrama de atividades para o caso de uso [001]. Os diagramas de atividades para os demais casos poderão ser acessados no repositório git (PEREIRA, 2023) com o código-fonte do sistema.

Figura 9 – Diagrama de atividades para o caso de uso [001]

- Diagrama de sequências

  O artefato de diagramação da UML permitiu representar a funcionalidade dos requisitos sob o fator do tempo de maneira técnica. O que complementou a elaboração de requisitos não funcionais e evidenciou etapas menos previsíveis no levantamento

de requisitos preliminares e proporcionando material para revisão do diagrama de atividades e da documentação de caso de uso.

A figura 10 apresenta o diagrama de sequências para o caso de uso [001]. Os diagramas de sequências para os demais casos poderão ser acessados no repositório git Pereira (2023) com o código-fonte do sistema.

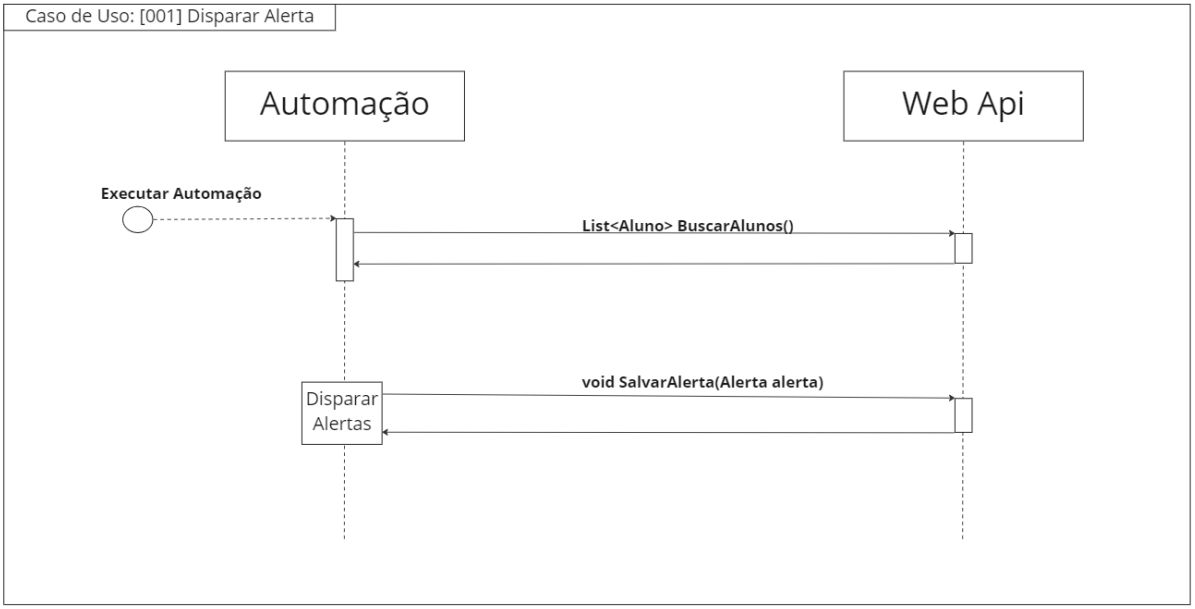

Figura 10 – Diagrama de sequências para o caso de uso [001]

- Diagrama de implantação

  A UML dispõe do artefato de diagrama de implantação para formalizar a estrutura que irá comportar o sistema. Neste trabalho, adotamos todas as informações produzidas pelos demais componentes da engenharia de software aplicados no desenvolvimento do projeto do sistema para a identificar todos os componentes do mesmo a fim de construí-los e implantá-los. Essas informações foram utilizadas como entrada no diagrama que representa a implantação do sistema e produz o conhecimento da organização das aplicações e onde as mesmas serão implementadas, dispondo de informações suficientes para determinar quais requisitos técnicos de ambiente devem ser alocados para hospedar o sistema. A figura 11 apresenta o diagrama de implantação.

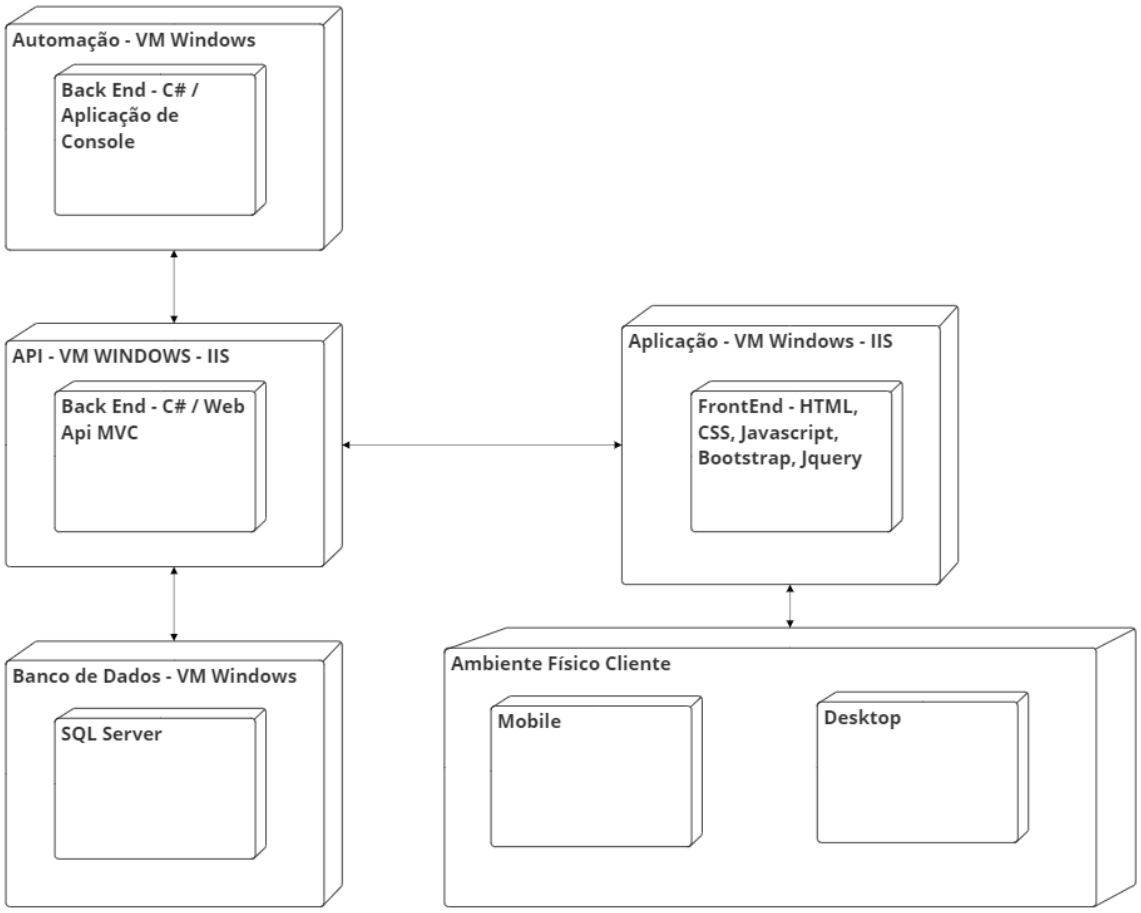

Figura 11 – Diagrama de implantação para caso de uso [001]

3.2 Arquitetura do sistema

A atividade proposta nesse trabalho de conclusão de curso utiliza como entrada informações do sistema de graduação, como, por exemplo, alunos, cursos, disciplinas, estágios e semestres para suprir a execução de um sistema que irá verificar continuamente a situação individual de cada discente, disparando, à medida que necessário, e-mails com o intuito de alertar os discentes para se atentarem a limites e prazos aos quais estão sujeitos conforme às normas de graduação.

Devido à natureza do trabalho, é necessário conciliar funcionalidades de múltiplas estratégias de software para implementar uma solução que abranja desde a análise da situação de cada discente até o disparo de e-mails de alerta, sendo essas funcionalidades: Uma interface web para usuário, rotina de execução, persistência de dados e comunicação com servidor SMTP para disparo de e-mails.

De acordo com as necessidades do projeto, foi adotada a arquitetura cliente-servidor para atingir nosso objetivo ao explorar a facilidade de dispor uma interface gráfica permitindo gerenciamento do sistema por um humano, e também o processamento do volume massivo de dados no lado do servidor.

"Muitas aplicações cliente-servidor podem ser construídas de acordo com três partes diferentes: uma parte que manipula a interação com um usuário, uma parte que age sobre um banco de dados ou sistema de arquivo e uma parte intermediária que, em geral, contém a funcionalidade central de uma aplicação. Essa parte intermediária está localizada logicamente no nível de processamento. Ao contrário de interfaces de usuário e bancos de dados, não há muitos aspectos comuns no nível de processamento."(TANENBAUM, 2007).

Assumindo o ponto de partida arquitetura cliente-servidor, foram organizadas as responsabilidades do sistema em 3 camadas, sendo elas a camada do usuário, camada de aplicação e camada de dados. As camadas são apresentadas pelo diagrama na figura 12.

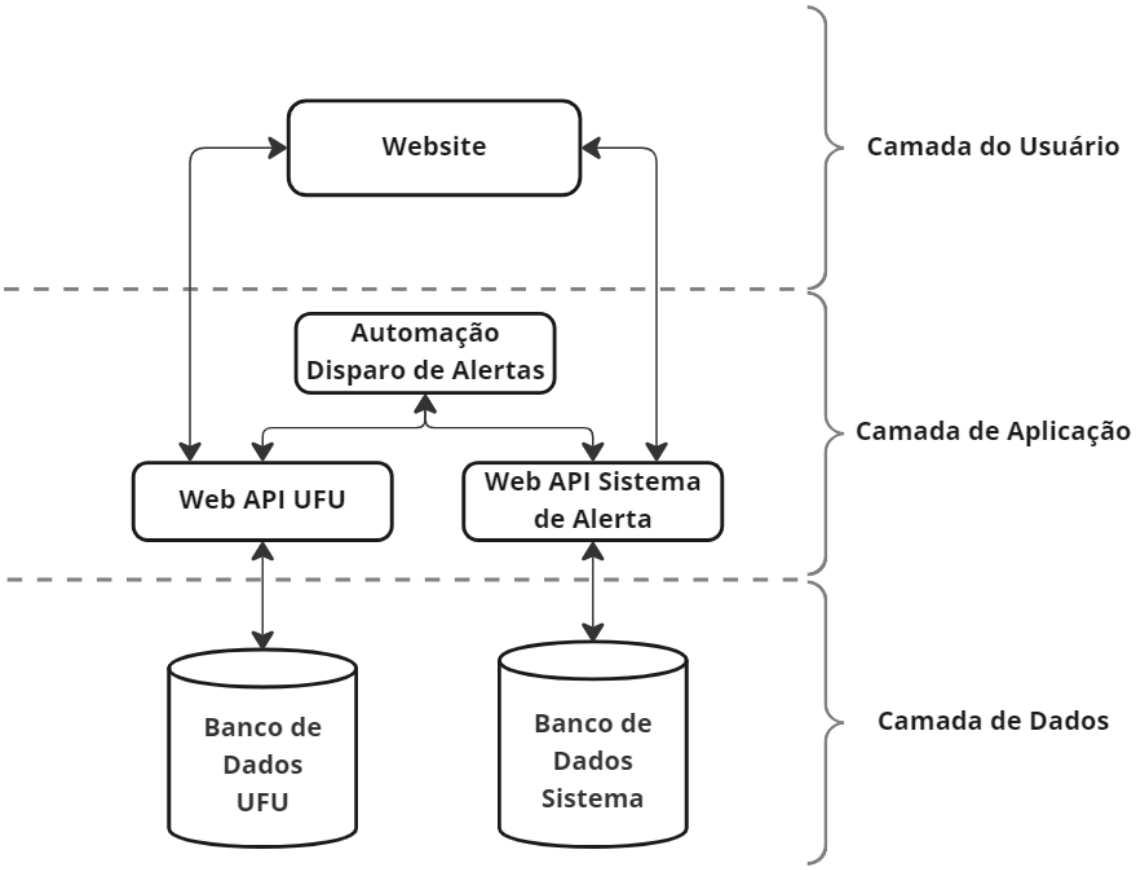

Figura 12 – Diagrama de arquitetura - camadas

3.3 Camada do usuário

Essa camada é composta por uma única aplicação web (web site) implementada em um projeto web app do .Net Core, adotando o estilo arquitetural de camadas modelo-visão-controlador (MVC).

Essa aplicação irá se comunicar com os outros componentes do sistema na camada de aplicação por meio de um *client* HTTP. A aplicação irá se comunicar com as web APIs por meio de requisições do protocolo HTTP para poder acessar e modificar a camada de dados que persistirá os parâmetros de execução da automação.

A implementação do projeto web app do .Net Core será feita em C#, juntamente com também as ferramentas padrão de mercado para desenvolvimento de web sites (HTML, CSS e Javascript, JQuery e Bootstrap) para construir as páginas web.

Para proporcionar acessibilidade para a aplicação, o web site de gerenciamento foi projetado para ser responsivo tanto para ser acessado por aparelhos desktop quanto mobile, por meio da framework do bootstrap.

Esse modelo de projeto do .Net Core é multiplataforma, logo há muitas possibilidades de hospedar esse sistema para disponibilizá-lo na web. Duas possíveis abordagens foram eleitas para a implantação deste trabalho.

A primeira e mais acessível abordagem seria implantar a aplicação na infraestrutura de servidores da FACOM que já são utilizados para disponibilizar os sistemas atuais da UFU.

A segunda seria consumir o serviço de aplicação como serviço da Azure, onde a *cloud* se encarrega de hospedar e dispor a aplicação na rede.

Seu objetivo é fornecer uma interface gráfica para um usuário humano ter acesso e gerenciar a execução da rotina (automação) que realizará o trabalho contínuo de disparar alertas para os discentes. Isso irá permitir que a coordenação acompanhe o sistema e também dará uma flexibilidade no gerenciamento da execução da automação fornecendo cadastros onde o usuário pode editar alguns parâmetros para cada regra responsável por disparar um alerta.

3.4 Camada de Aplicação

A camada de aplicação assumirá a responsabilidade de fornecer o acesso à camada de dados e também por executar as regras de negócio do sistema.

Sua organização se dá por duas aplicações web APIs e uma aplicação de console. As duas APIs irão fornecer o acesso à camada de dados. Essas APIs serão consumidas pelo web site na camada do usuário e também pela aplicação de console que fica nessa

camada de dados agendada como uma rotina (automação).

3.4.1 Web API alerta de prazos

Essa aplicação será responsável por fornecer acesso à camada de dados. Esse modelo de aplicação adota o padrão RESTfull, implementada em C#. A questão de hospedagem desse aplicativo segue os mesmos padrões planejados para o web site de gerenciamento.

Seu objetivo é dispor o acesso aos dados do banco de dados do sistema de alerta de prazos. Essa API será consumida por duas aplicações:

- Aplicação web (web site)

  O web site de gerenciamento irá consumir a API de alerta de prazos para buscar os dados de parâmetro para a execução do sistema de alerta e dados de controle de usuário da plataforma.

- Automação

  A automação irá consumir essa API para acessar os dados persistidos no banco de dados que definem os parâmetros de controle que são gerenciados pela aplicação web de gerenciamento.

3.4.2 Web API UFU

A API da UFU irá intermediar o acesso ao banco de dados referente à organização do sistema de graduação da FACOM. Seu modelo de aplicação adota o padrão RESTfull, implementada em C#. A questão de hospedagem desse aplicativo segue os mesmos padrões planejados para o web site de gerenciamento.

Essa API será consumida por duas aplicações:

- Aplicação web (web site)

  O web site de gerenciamento irá consumir a API da UFU para buscar dados referentes aos cursos para auxiliar no gerenciamento entre regras e cursos dentro do site.

- Automação

  A automação irá consumir essa API para buscar dados de todos os alunos, matrículas em disciplinas, cursos, disciplinas, estágios e semestres. Esses são os principais dados de entrada para a automação, necessários para execução da regra de negócio e disparo dos alertas.

Essa API não é o foco do projeto, mas teve um papel muito importante em seu desenvolvimento. Ela foi desenvolvida para sustentar o desenvolvimento do sistema principal ao representar a base de dados da UFU e o acesso a esses dados. Pois essa estrutura já existe e está em vigor constituindo toda a estrutura digital que gere o sistema da FACOM, mas não é disponível para os discentes por motivos de segurança.

Como essa API é uma representação da estrutura real da UFU, para cumprir seu objetivo ela não precisa possuir todos os recursos de sua versão real implementada na UFU hoje, basta atingir a um nível de similaridade suficiente para construir a ponte de acesso aos dados que o sistema de alerta precisa.

Uma vez aprovado projeto para ser implementado em produção, integrando com o sistema da UFU, é esperado que haja alguma divergência técnica de comunicação, porém serão divergências que poderão ser corrigidas tanto na estrutura real da UFU como na parte do sistema de alerta de prazos que consome os dados da UFU sem prejudicar a lógica que move o sistema de alerta de prazos, pois o sistema foi implementado de uma maneira similar ao funcionamento do sistema da FACOM.

Um trunfo que essa aplicação proporcionou para o desenvolvimento do sistema de alerta de prazos foi a implementação de um *endpoint* para atender ao comando de uma requisição que popula o banco de dados que representa a base de dados da UFU com dados realísticos e massivos de alunos, cursos, disciplinas, matrículas, estágios e trancamentos. Foram implementados algoritmos randômicos para criação de nomes de discentes e eventos como estágios, trancamentos e aprovação e reprovação de disciplinas. Isso proporcionou uma base de dados realística o suficiente para apoiar no desenvolvimento e testes de execução do sistema de uma maneira produtiva, uma vez que se fez possível popular uma base inteira representando dezenas de semestres em menos de 5 minutos.

Porém, deve ser considerado que essa API e seu banco de dados tem sua utilidade durante o desenvolvimento desse trabalho. Uma vez que esse projeto seja considerado a ser levado adiante e adotado pela FACOM, essa API e seu banco de dados serão descontinuados, pois não haverá necessidade de representar a estrutura do sistema da FACOM uma vez que seja fornecido acesso aos dados de produção e homologação necessários para o sistema de alerta de prazos consumir.

3.4.3 Automação

Há a necessidade de um componente para disparar os e-mails como uma rotina que executará diariamente, logo essa implementação deve adotar um modelo de automação.

Sua implementação foi planejada como um projeto do tipo aplicação console do .Net Core, implementado em C#. Para transformar essa aplicação em uma rotina ela será agendada em uma aplicação agendadora de rotinas. Uma opção para implantação é agen-

dar esse programa em um aplicativo agendador de tarefas em uma máquina com sistema operacional Windows ou baseado em Linux, contando que disponha de acesso a mesma rede que o resto do sistema. O agendador de tarefas é um aplicativo nativo das distribuições dos sistemas operacionais Windows mais recentes e, nele é possível criar uma rotina que executa um programa da máquina, de modo que seja possível configurar a frequência de repetição da execução de uma maneira bem completa. Uma alternativa nativa para ambiente Linux seria agendar a automação por meio do aplicativo Crontab. Apesar de não ser um caminho adotado para esse projeto, esse tipo de aplicação é multiplataforma e também pode ser implantado em um sistema operacional MacOS ou sistemas operacionais baseados em kernel Linux com outros métodos de agendamento de rotina.

Essa Automação assume o papel principal na regra de negócio do sistema desse projeto. Consumindo dados das duas APIs (API da UFU e API do sistema de alerta de prazos).

Seu objetivo é buscar as informações necessárias da base de dados da FACOM para poder validar a situação de cada discente, verificando se o discente atende a algum dos eventos previamente implementados na plataforma que identificam que o aluno se enquadra em alguma situação que implique o cumprimento de um prazo.

Além de buscar a base de dados da graduação, essa automação consome a API do sistema de Alerta de Prazos, para acessar os dados de parametrização da execução. Parte desses dados são gerenciados pelo usuário no web site de gerenciamento e persistidos na base de dados do sistema de alerta de prazos, a outra parte são registros de alertas gerados pela própria automação e persistidos na base para manter o controle de qual discente já recebeu, qual tipo de alerta e quando isso aconteceu, para evitar *spam* na caixa de e-mail dos discentes.

A execução ocorre em um loop que percorre toda a lista de discentes da FACOM, e quando é verificado que algum alerta deve ser disparado para um discente, a aplicação o envia um e-mail com uma mensagem informativa sobre a situação do discente e os prazos aos quais precisa se atentar.

O envio de e-mail feito pela automação utiliza a classe MailMessage da biblioteca biblioteca System.Net.Mail, uma solução padrão da ferramenta que dispõe um SmtpClient que permite enviar mensagens por meio do protocolo SMTP para servidores SMTP.

Para este projeto foi escolhido um Servidor SMTP da Microsft com o seguinte endereço smtp.office365.com usando a porta 587. Por meio do SmtpClient da automação, basta passar as credenciais de uma conta Outlook para acessar esse servidor que irá enviar o e-mail para caixa do discente automaticamente. Essa solução foi adotada por esse servidor fornecer seu serviço gratuitamente numa quantidade limitada de envios. Para uma solução definitiva o serviço pode ser contratado pela Microsoft para fornecer

permissão para uma conta Outlook que consiga disparar e-mails suficientes para todos os discentes da base da FACOM.

Após cada alerta, o sistema persiste um registro no banco de dados do sistema de alerta, contendo informações como matrícula do aluno, tipo de alerta e data de envio do alerta. Todos os registros de envio serão buscados no início da execução da automação, para verificar se e quando um tipo de alerta foi enviado.

3.5 Camada de dados

Para este trabalho, foi necessário construir dois bancos de dados relacionais distintos no SQL Server para sanar a necessidade de persistir as informações devidas para o funcionamento do sistema de alerta de prazos.

Ambos os bancos de dados tiveram sua estrutura gerada automaticamente pela Entity Framework Core, uma ferramenta de mapeamento objeto-relacional (*ORM - Object Relational Mapping*) padrão da Microsoft para projetos .NET Core que segue a abordagem *code first* e gera um banco de dados completo com base nas classes de entidades do sistema sem a necessidade de codificação em SQL.

Os bancos de dados deste trabalho serão acessados diretamente por web APIs da camada de aplicação. Sua criação, acesso e manipulação são realizados dentro das APIs por meio do Entity Framework Core.

A implantação do banco de dados deve ser feita nos servidores da FACOM juntamente com as outras aplicações, bastando apenas possuir o SQL Server instalado e, gerar a instância do banco de dados do sistema por meio da ferramenta Entity Framework Core.

Outra solução moderna planejada para implantação é gerar os dois bancos de dados na Azure utilizando o serviço de base de dados como serviço.

Cada banco de dados nesse projeto tem seu propósito, sendo eles:

3.5.1 Banco de dados do sistema de alerta de prazos

Este banco de dados possuirá três tabelas baseadas no modelo das entidades do sistema de alerta de prazos. Essas tabelas são:

- Usuários: Responsável por persistir as credenciais do usuário utilizadas na autenticação do web site de gerenciamento.
- Regras: Uma tabela para armazenar os dados de parâmetros para cada tipo de validação de alertas na automação.

- Alertas: Essa tabela armazena o registro de cada alerta disparado para cada discente da FACOM para ser consultado na execução da automação de disparo de alertas.

3.5.2 Banco de dados representando a estrutura de dados da FACOM

Esse banco de dados foi gerado com base nas entidades definidas na API e representa os dados fornecido pela FACOM. Sua estrutura conta com as seguintes tabelas simulando como o sistema de graduação persiste os dados para gestão da faculdade: Aluno, Curso, Disciplina, MatriculaDisciplina, Semestre e Estágio.

3.6 Versionamento

O controle do versionamento da aplicação foi feio por meio da plataforma do GitHub, repositório de código fonte de onde é possível acessar a implementação da aplicação em https://github.com/Erick-Cris/SistemaAlertaDePrazos.

4 Apresentação e Resultados

Esse capítulo apresentará o funcionamento do sistema aos olhos do usuário.

É importante ao leitor deste trabalho de conclusão de curso se atentar à terminologia regra adotada na exibição da interface do usuário. As regras se referem a uma validação que a automação faz para cada discente durante sua execução. Validação essa que indica se o usuário se encontra ou não em uma situação onde seja necessário alertá-lo por e-mail sobre os prazos aos quais deve se atentar.

4.1 Tela: Autenticação

A tela de autenticação é a primeira tela com a qual o usuário irá se deparar ao acessar o web site de gerenciamento em seu navegador web. Por meio dela o usuário pode inserir suas credenciais de usuário (e-mail e senha) para ter acesso à aplicação.

Após a autenticação bem sucedida, o usuário é redirecionado para a tela de listagem de regras que é a principal tela do sistema. Tela apresentada na figura 13.

Figura 13 – Tela de autenticação da plataforma

4.2 Tela: Lista de regras

A tela de listagem das regras apresenta uma tabela listando cada regra que é executada pela automação de disparo de e-mail. Sua função é proporcionar uma visão geral sobre o status de processamento de todas as regras, e fornecer um meio para editar cada regra individualmente. Tela apresentada na figura 14.

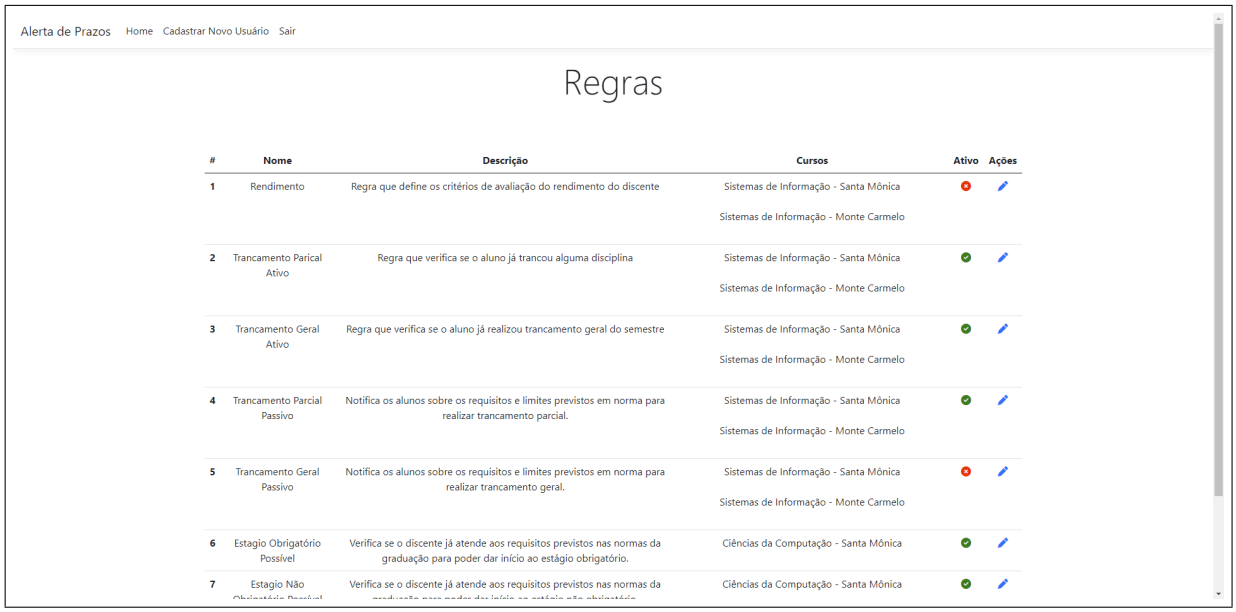

Figura 14 – Tela de listagem de regras

Sobre a tabela:

- Nome e descrição

  Através das colunas nome e descrição o usuário pode identificar e compreender o propósito para cada regra.

- Cursos

  A coluna cursos indica para quais cursos uma regra está habilitada para executar a verificação de alerta na automação.

- Ativo

  A coluna ativo apresenta o status do parâmetro de ativo/inativo que define se uma regra pode ou não ser validada na execução da automação.

- Ações

  A coluna ações apresenta um ícone de lápis para cada regra. Esse ícone é um *link* que direciona o usuário para a tela de edição da regra, onde o usuário pode editar os parâmetros de configuração.

4.3 Tela: Editar regra

Essa tela dispõe de um formulário que permite editar dados que servirão de parâmetro para a execução dessa regra pela automação de alerta de prazos.

Nesse formulário o usuário gerenciador do sistema pode selecionar para quais cursos a validação da regra na automação vai entrar em vigor, definir a descrição da regra e habilitar ou desabilitar a regra. Tela apresentada na figura 15.

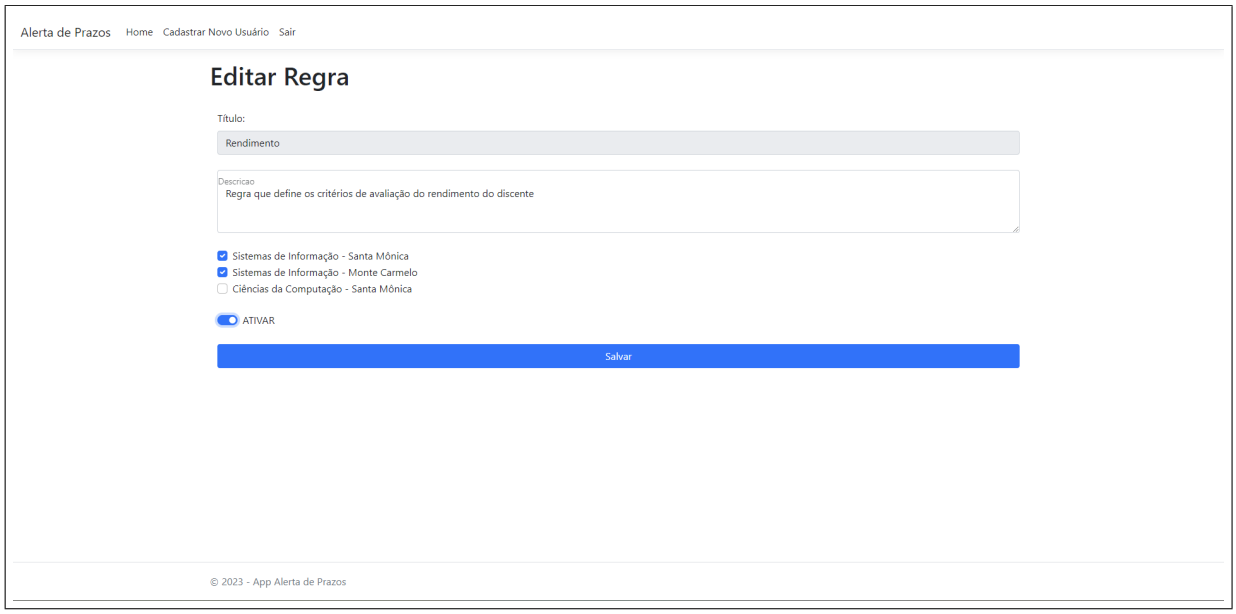

Figura 15 – Tela de edição de uma regra

4.4 Tela: Criar Usuário

A funcionalidade de cadastro de novos usuários foi implementada de modo que só possa ser acessada por um usuário já autenticado na plataforma. Ou seja, para criar um novo usuário, é necessário primeiro acessar o sistema com uma conta que já existente. Essa tela dispõe de um pequeno formulário onde o usuário logado vai definir parte das credenciais do novo usuário, sendo essa parte o nome do usuário e seu e-mail.

Ao submeter esse formulário, um novo usuário é cadastrado no banco de dados, porém, inicialmente o usuário é criado com status inativo e sem senha definida. Assim, no momento da submissão desse formulário o sistema envia um e-mail para o novo usuário contendo um *link* de ativação desse usuário.

Assim que o novo usuário receber o e-mail de validação em sua caixa de entrada ele poderá acessar o sistema e definir uma nova senha para ativar sua conta e poder acessar as funcionalidades do sistema. Tela apresentada na figura 16.

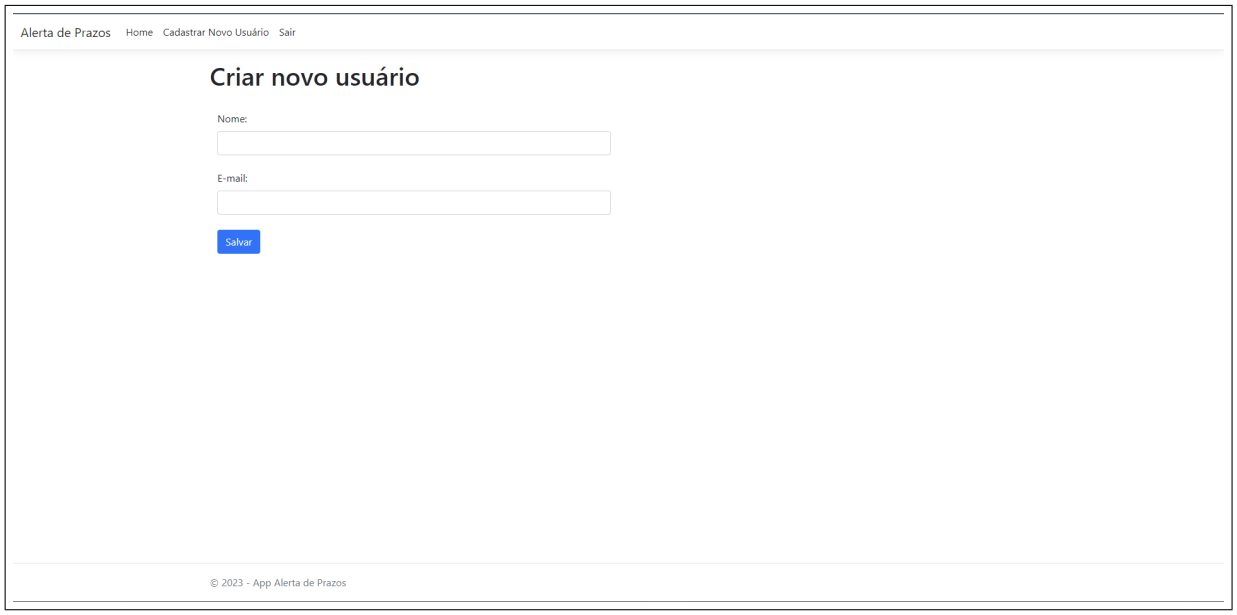

Figura 16 – Tela de criação de usuário

4.5 Etapa: Email de ativação

A figura 17 representa o *template* do e-mail de ativação de usuário que é enviado para a caixa de e-mail do novo usuário cadastrado no sistema.

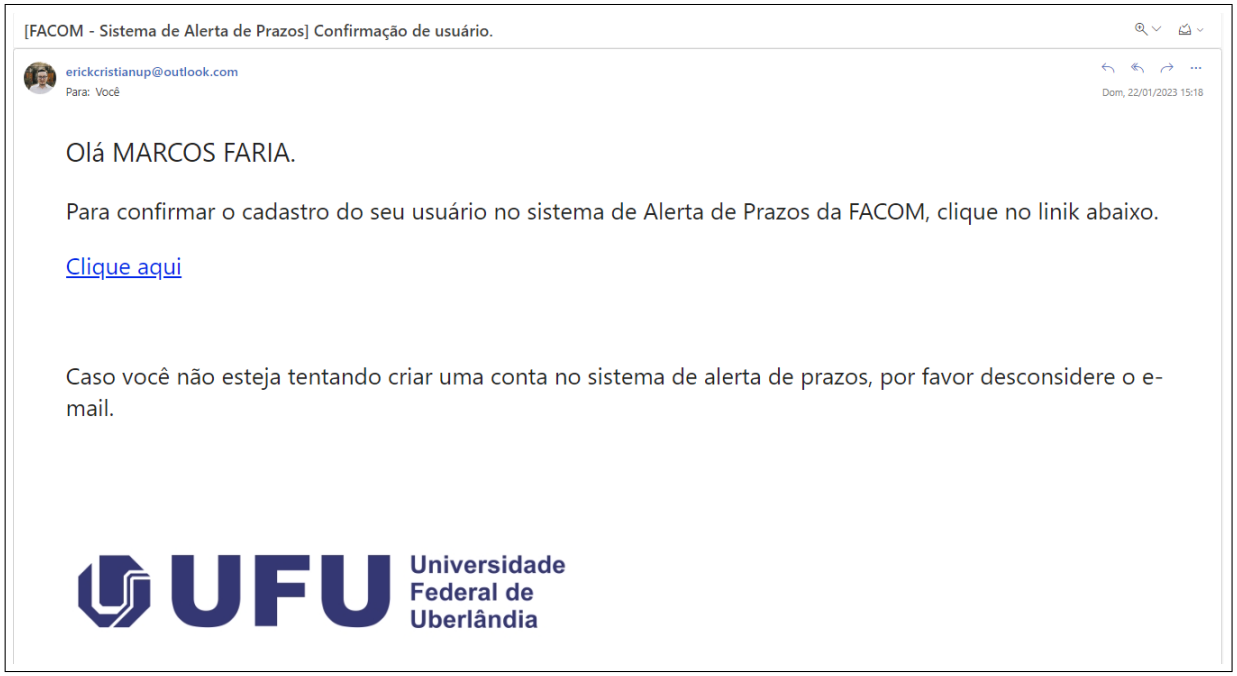

Figura 17 – *Template* do e-mail de ativação de usuário

4.6 Tela: Ativar usuário e criar senha

Essa é a tela para qual o usuário é redirecionado após clicar no *link* de ativação presente no corpo do e-mail de ativação que recebeu em sua caixa de entrada.

Há um token codificado em BASE64 sendo passado como parâmetro nesse *link* de ativação. Esse token contém informações necessárias para identificar o usuário na base de dados para poder ser ativado. É importante ressaltar que, adotar a técnica de codificar em base64 é uma estratégia preliminar para proteger os dados de identificação do usuário. Para futuras pesquisas e evoluções será necessário mudar a forma de proteger esses dados para um método mais seguro, pois o base64 é facilmente decodificável.

Através do token presente no *link* de ativação, a tela busca os dados do usuário e em seguida apresenta um pequeno formulário para permitir que o novo usuário crie a sua senha. Assim, ao submeter o formulário com a nova senha, a mesma é codificada com um algoritmo de *hash* seguro SHA2 disposta pela biblioteca System.Security.Cryptography que é uma biblioteca padrão de *hash* do C#.

Após a submissão, um *hash* da nova senha é gerado e persistido no banco de dados do sistema de alerta de prazos para ser usado nas futuras autenticações do usuário no sistema e o status do usuário muda para ativo. Tela apresentada na figura 18.

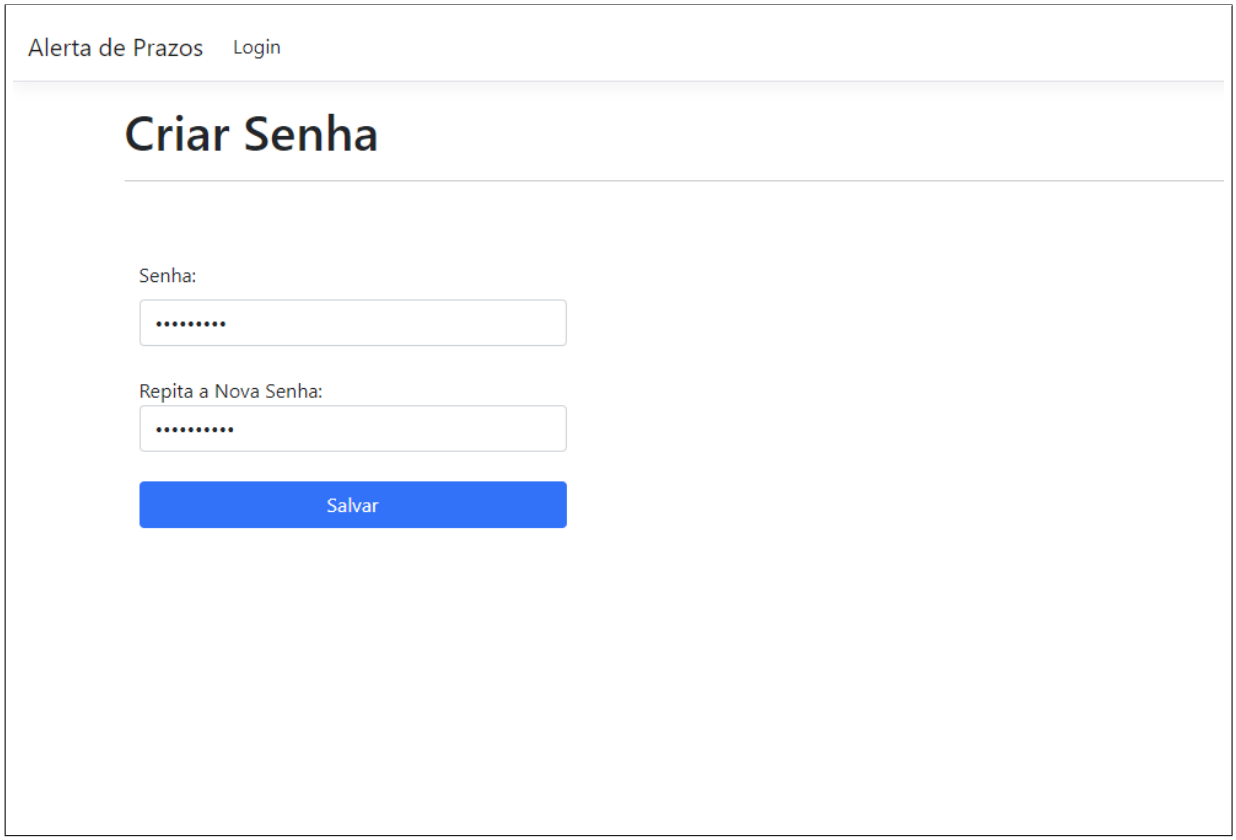

Figura 18 – Tela de ativação / criação de senha

4.7 Responsividade

Para aprimorar a experiência com o usuário gerenciador do sistema, as telas da aplicação foram implementadas dispondo de responsividade, por meio da framework do bootstrap. Assim, o website de gerenciamento pode ser acessado por meio de diversos dispositivos sem complicações por conta de dispositivos com telas de diferentes dimensões. Um exemplo da responsividade implementada está na imagem 19 que apresenta a mesma tela da imagem 14, porém sob a perspectiva de um dispositivo mobile.

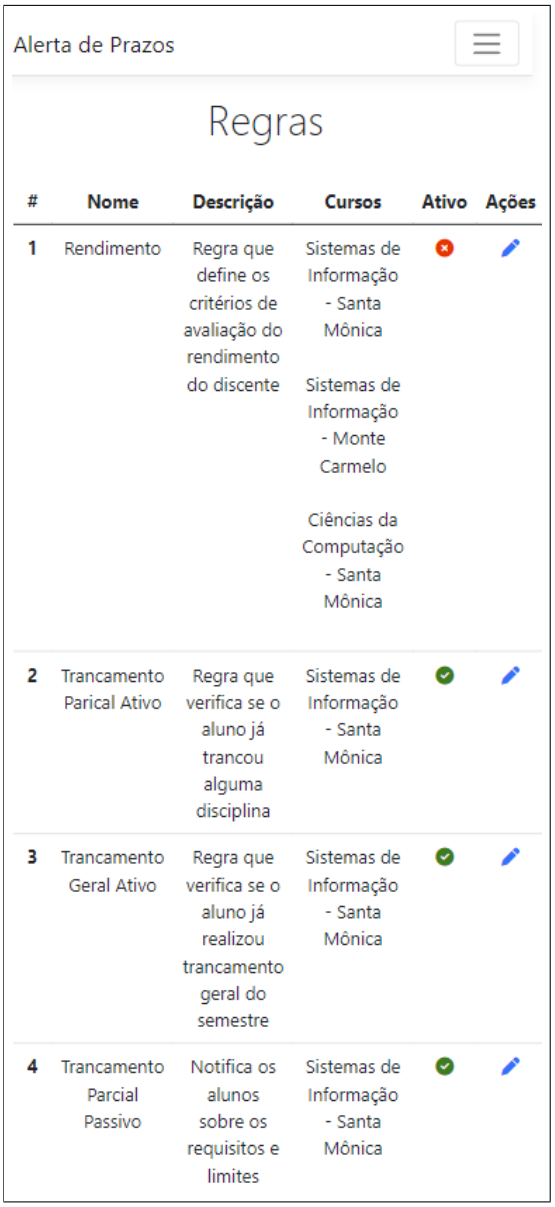

Figura 19 – Tela de listagem de regras

4.8 Execução e desempenho da automação

Para medir o custo computacional e tempo de execução para a automação que será responsável por executar todo o trabalho de validação da situação dos alunos e, pelo disparo de alerta de prazos, o sistema foi submetido a uma série de testes de execução.

Os testes do sistema foram realizados em ambiente local, com todos os programas executando em uma única máquina com as seguintes configurações: Sistema operacional: Windows 11 Home, processador: Intel(R) Core(TM) i7-10700F, memória: 2x 8 GB 3200 MHz DDR4 e armazenamento: SSD WD Green 480 GB.

Com resultado de múltiplos testes de execução no ambiente descrito acima, foi possível identificar que o tempo de execução para cada aluno seguiu um padrão bem consistente, permitindo mensurar quanto tempo em média um aluno leva para ser validado e seus respectivos alertas disparados com uma variação desprezível (menos de 0,01 segundos).

É importante resaltar que os dados obtidos através da execução foram obtidos de forma empírica e estão ligados ao ambiente ao qual os testes foram submetidos.

Foi identificado que a validação para um aluno sem disparo de alertas leva em média 0,018 segundos e, quando um alerta é identificado e disparado para o aluno, cada alerta leva em média 5 segundos.

Para estimar o tempo médio de execução completa da automação de alerta, basta assumir um valor de entrada X representando a quantidade de alunos a serem validados, um valor N para a quantidade de alertas que serão identificados e disparados e, assumir que independente de ser identificado um caso de alerta, cada aluno consome em média 0,018 segundos da execução da automação. Assim podemos definir que a fórmula que representa o tempo médio (em segundos) de execução para uma entrada X se dá por: (X * 0,018) + N * 5.

Como exemplo, consideremos uma entrada de 3000 alunos para serem validados pela automação e, que a validação irá identificar e disparar 5000 alertas. Logo é possível calcular que a execução dessa entrada levará em torno de 6,96 horas para concluir sua execução no ambiente em que o sistema está sendo testado.

O custo computacional para execução da automação se manteve próximo de:

- Consumo de CPU: 3%.
- Consumo de Memória: 1.760 MB.
- Consumo do SSD: 0,1MB/s.

A partir dos dados coletados, é possível observar que o sistema é capaz de processar um alto volume de entrada em menos de 24 horas com baixo custo de processamento.

5 Conclusão

O conjunto de objetivos descritos neste trabalho de conclusão de curso pode ser resumido em identificar prazos importantes, implementar técnicas de engenharia de software no desenvolvimento do projeto e desenvolver um sistema que envie alertas por e-mail para os discentes se atentarem ao cumprimento de tais prazos.

Foram identificados prazos na documentação oficial de normas de graduação da FACOM que especificam situações nas quais os discentes devem se atentar para evitar complicações como perda de aproveitamento de componentes e até mesmo jubilação. Situações que colocam os discentes em situação de recorrer a solicitações de reconsideração para solucionar transtornos gerados por falta de informação e de atenção aos limites impostos pelas normas de graduação.

Algumas definições de limites e prazos identificados nas normas de graduação e que cabem na proposta de alertar o discente foram implementadas no sistema de alerta que valida essas situações para todos os discentes da base de alunos da FACOM, e envia um e-mail conscientizando o discente.

As técnicas de engenharia de software aplicadas no desenvolvimento se provaram eficazes ao proporcionarem um escopo sólido para desenvolvimento do software permitindo enxergar a responsabilidades e como dividi-las entre diferentes processos. Outro benefício que pode ser observado é que durante o desenvolvimento alterou-se muito pouco o modelo criado na documentação produzida pelas técnicas de engenharia de software, o que mostra que projetar o sistema permitiu um desenvolvimento objetivo e com poucos desvios, minimizando o custo do desenvolvimento e mantendo as funcionalidades dentro do objetivo planejado.

Os componentes desenvolvidos para representar de forma realística os dados da FACOM (web API e banco de dados da FACOM) se provaram uma ferramenta muito útil que deu suporte durante todo o desenvolvimento do projeto. A abordagem também foi uma ótima oportunidade para explorar como a lógica de gestão da faculdade funciona. Por mais que a ferramenta seja projetada para ser descontinuada ao implantar o sistema de alerta em produção, esses componentes se provaram valiosos e proporcionaram produtividade, profundidade e enriquecimento para o desenvolvimento deste trabalho que pode inclusive ser usado para dar suporte em outros trabalhos que necessitem de trabalhar com a base de dados do sistema de graduação.

Uma observação relevante pode ser feita a um obstáculo que foi encontrado durante o desenvolvimento do software a respeito de uma proposta de parametrização para execução do disparo de alertas de maneira que seja possível criar novas regras de alerta

pelo próprio usuário. Ao cogitar a possibilidade de implementar tal funcionalidade, se materializou um obstáculo. Uma abordagem dessas implica que um usuário sem conhecimento de programação pode cadastrar uma regra de disparo de alerta para qualquer circunstância da graduação, porém os requisitos e limites previstos em norma não seguem sempre o mesmo padrão e podem ser definidos por variados tipos de recursos, componentes, dados e situações. O que se apresenta como uma proposta lógica desafiadora para ser explorada em futuras pesquisas e evoluções do sistema ou trabalhos relacionados.

A estratégia de disparar e-mails foi o meio de comunicação adotada nesse trabalho. Para futuras pesquisas ou evoluções deste tema, podem ser exploradas outras oportunidades de meios de comunicação como Whatsapp, Microsoft Teams e aplicações mobile para tornar a informação ainda mais acessível para os discentes. Da maneira como o sistema foi desenhado e construído, já dispõe de recursos de comunicação entre aplicações, o que torna a estratégia adaptável a comunicação com serviços de terceiros. A UFU já dispõe de um aplicativo mobile para divulgar informações para discentes, o que pode ser uma boa oportunidade de integração com o sistema de limites e alertas de prazos.

Por fim, a conclusão do desenvolvimento do software e o sucesso em seus testes colaboram para o que se faz uma solução eficaz para atingir aos objetivos especificados, proporcionando um sistema capaz de disparar alertas a todo o corpo discente dos cursos da FACOM com informações a respeito de prazos e situações a serem cumpridos.

Referências

COLEGIADO. **Normas de Estágio Curricular do Bacharelado em Sistemas de Informação**. Uberlândia, 2022. Disponível em: <https://facom.ufu.br/legislacoes/normas-de-estagio-curricular-do-bacharelado-em-sistemas-de-informacao>. Acesso em: 30 junho. 2022. Citado na página 11.

GRADUAÇÃO, C. de. **RESOLUÇÃO CONGRAD Nº 46**. Uberlândia, 2022. Disponível em: <http://www.prograd.ufu.br/legislacoes/resolucao-congrad-no-46-de-28-de-marco-de-2022-normas-gerais-da-graduacao-da-ufu>. Acesso em: 30 junho. 2022. Citado 2 vezes nas páginas 11 e 12.

LOCATELLI, J. A. **Sistema Online para Distribuição de Disciplinas**. Monografia (Aplicação para gestão de processos da Universidade Federal de Uberlândia) — Faculdade de Comutação, Universidade Federal de Uberlândia, Uberlândia, 2015. Citado na página 17.

MDN, M. D. N. **MDN**. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 14 fevereiro. 2023. Citado na página 15.

MICROSOFT, c. **C**. 2022. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>. Acesso em: 14 fevereiro. 2023. Citado na página 13.

MICROSOFT, d. **Projeto de API Web RESTful**. 2023. Disponível em: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>. Acesso em: 14 fevereiro. 2023. Citado na página 14.

NET, c. M. D. D. **.Net**. 2022. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/core/introduction>. Acesso em: 14 fevereiro. 2023. Citado na página 13.

PEREIRA, E. C. de O. **Repositório de código fonte do Sistema de Alerta de Prazos**. 2023. Disponível em: <https://github.com/Erick-Cris/SistemaAlertaDePrazos>. Acesso em: 14 fevereiro. 2023. Citado 2 vezes nas páginas 28 e 29.

PRESSMAN, R. S. **Engenharia de Software - Uma Abordagem Profissional**. 8. ed. AMGH: [s.n.], 2016. Acesso em: 30 jun 2022. Citado 4 vezes nas páginas 4, 9, 12 e 18.

SCANDIFFIO, M. G. **Sistema de controle de prazos para alunos de pós-graduação.** Monografia (Sistema de controle de prazos) — Faculdade de Comutação, Universidade Federal de Uberlândia, Uberlândia, 2017. Disponível em: <https://repositorio.ufu.br/handle/123456789/25266>. Acesso em: 14 fevereiro. 2023. Citado na página 16.

SILVA, T. R. **Protótipo de aplicação web para gestão de estoque de um laboratório de pesquisa da Universidade Federal de Uberlândia**. Monografia (Sistema de gestão de estoque de um laboratório de pesquisa) — Faculdade de Comutação, Universidade Federal de Uberlândia, Uberlândia, 2021. Disponível em: <https://repositorio.ufu.br/handle/123456789/35042>. Acesso em: 14 fevereiro. 2023. Citado na página 17.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. AMGH: Pearson Education, 2011. Acesso em: 16 aug 2022. Citado 2 vezes nas páginas 16 e 27.

SQLSERVER, c. M. D. **SQL Server**. 2022. Disponível em: <https://docs.microsoft.com/pt-br/sql/relational-databases/databases/databases?view=sql-server-ver16>. Acesso em: 14 fevereiro. 2023. Citado na página 13.

TANENBAUM, A. S. **Sistemas Distribuídos - Princípios e Paradigmas**. 2. ed. AMGH: [s.n.], 2007. Acesso em: 30 jun 2022. Citado na página 31.

TEAM, B. **Bootstrap**. 2022. Disponível em: <https://getbootstrap.com/docs/5.2/getting-started/introduction/>. Acesso em: 14 fevereiro. 2023. Citado na página 15.

W3C, c. **C**. 2022. Disponível em: <https://www.w3.org/html/>. Acesso em: 14 fevereiro. 2023. Citado na página 14.

_____. **C**. 2022. Disponível em: <https://www.w3.org/standards/webdesign/htmlcss>. Acesso em: 14 fevereiro. 2023. Citado na página 14.