Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Desenvolvimento de uma plataforma para o gerenciamento de comissões

Bruno Henrique Magno Tomaz

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Rafael Frederico Alexandre

Junho, 2022
João Monlevade–MG

Bruno Henrique Magno Tomaz

Desenvolvimento de uma plataforma para o gerenciamento de comissões

Orientador: Rafael Frederico Alexandre

Monografia apresentada ao curso de Engenharia da Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Junho de 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

T655d Tomaz, Bruno Henrique Magno.
Desenvolvimento de uma plataforma para o gerenciamento de comissões. [manuscrito] / Bruno Henrique Magno Tomaz. - 2022.
78 f.

Orientador: Prof. Dr. Rafael Frederico Alexandre.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia de Computação .

1. Engenharia de software. 2. Experiência do usuário - Usabilidade. 3. Serviço público - Comissões. I. Alexandre, Rafael Frederico. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.41

Bibliotecário(a) Responsável: Sione Galvão Rodrigues - CRB6 / 2526

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Bruno Henrique Magno Tomaz

Desenvolvimento de uma plataforma para o gerenciamento de comissões

Monografia apresentada ao Curso de Engenharia da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de bacharel em Engenharia da Computação

Aprovada em 22 de junho de 2022

Membros da banca

Doutor - Rafael Frederico Alexandre - Orientador (Universidade Federal de Ouro Preto)
Doutor - Bruno Pereira dos Santos - (Universidade Federal de Ouro Preto)
Doutor - Mateus Ferreira Satler - (Universidade Federal de Ouro Preto)
Bacharel - Weverton Costa Peixoto - (Universidade Federal de Ouro Preto)

[Digite o nome do orientador (apenas a primeira letra de cada nome maiúscula)], orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em XX/XX/XXXX

Documento assinado eletronicamente por **Rafael Frederico Alexandre**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 30/01/2023, às 14:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0465928** e o código CRC **CA782CC2**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado a Deus, aos meus familiares e amigos.

Agradecimentos

Em primeiro lugar, agradeço a Deus por me sustentar e guiar durante toda a minha trajetória.

Agradeço aos meus pais, Geraldo Magela e Maura, por todo o amor, cuidado e apoio ao longo das etapas da minha vida. Vocês são maravilhosos!

Ao meu irmão, Felipe Tomaz, por me apresentar o mundo da computação, além de ser o meu tutor e melhor amigo. O seu companheirismo foi imprescindível para que eu chegasse até aqui.

À minha namorada, Gleiciely, pelo carinho, tranquilidade e paciência em todos os momentos.

Aos grandes amigos que me foram apresentados através da graduação. Foi um prazer compartilhar tantos momentos com cada um de vocês.

Direciono também meus agradecimentos a todos os professores e colaboradores da Universidade Federal de Ouro Preto. Em especial, ao meu orientador, Prof. Dr. Rafael Alexandre, por todos *feedbacks*, dicas e conselhos na elaboração deste trabalho.

"No man is an island entire of itself; every man is a piece of the continent, a part of the main."

— John Donne (1572 – 1631),
in: No Man Is an Island.

Resumo

O gerenciamento de comissões universitárias centralizado, apoiado por ferramentas de *software* modernas, pode constituir-se como elemento fundamental na administração dos processos de uma instituição de ensino. No entanto, atualmente, o Departamento de Computação e Sistemas utiliza um modelo de planilhas eletrônicas para a gestão das comissões, que apresenta problemas de usabilidade, além de impossibilitar a centralização das informações. Sendo assim, este trabalho apresenta a plataforma intitulada Insider, que introduz um meio moderno, versátil e focado na experiência do usuário, para a unificação dos dados das comissões. Como resultado, este trabalho conclui que é possível escalar a utilização do *software* para todos os departamentos do instituto e, em última instância, da universidade. Além disso, observou-se que uma plataforma personalizada, com foco na experiência do usuário, pode contribuir consideravelmente nos processos de tomada de decisão, aquisição de dados e resolução de problemas no contexto da gerência de comissões em comparação com o uso de planilhas eletrônicas.

Palavras-chaves: Gerenciamento de Comissões. Usabilidade. Experiência do Usuário.

Abstract

A centralized management of university commissions, supported by modern software tools, can be a fundamental element in the administration of the processes of an educational institution. However, currently, the departments of the Institute of Exact and Applied Sciences use an electronic spreadsheet model for the management of commissions, which presents usability problems, in addition to making it impossible to centralize information. Therefore, this work presents the platform called Insider, which introduces a modern, versatile and user-experience-focused way to unify data from the Committees of the Department of Computing and Systems. As a result, this work concludes that it is possible to scale the use of software to all departments of the institute and, ultimately, of the university. In addition, it was observed that a customized platform, with a focus on user experience, can contribute considerably to decision-making, data acquisition and problem solving processes in the context of commission management compared to the use of electronic spreadsheets.

Key-words: Commissions Management. Usability. User Experience.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API *Application Programming Interface*

AWS *Amazon Web Services*

CA *Clean Architecture*

CSS *Cascading Style Sheets*

DECSI Departamento de Computação e Sistemas

DER Diagrama Entidade-Relacionamento

HTML *HyperText Markup Language*

HTTP *HyperText Transfer Protocol*

IDE *Integrated Development Environment*

IES Instituições de Ensino Superior

IoT *Internet of things*

JSON *Javascript Object Notation*

JS *JavaScript*

JWT *JSON Web Token*

MER Modelo Entidade-Relacionamento

OO Orientação a Objetos

QR *Quick Response*

REST *Representational state transfer*

SGBD Sistema de Gerenciamento de Banco de Dados

TDD *Test Driven Development*

TICs Tecnologias de Informação e Comunicação

UFOP Universidade Federal de Ouro Preto

UI *User Interface*

URI *Uniform Resource Identifier*

UX *User Experience*

XML *eXtensible Markup Language*

Sumário

1 Introdução

O uso intensivo de Tecnologias de Informação e Comunicação (TICs) nos ambientes organizacionais tem evoluído constantemente ao longo dos anos. De fato, a digitalização de processos em instituições de diferentes portes ou áreas de atuação lograram êxito em inúmeras conjunturas. Seguindo essa linha, Prado e Souza (2014) atestam que sistemas de informação tem auxiliado o registro e análise de dados em quase todas as atividades institucionais na atualidade.

Souza e Monteiro (2015) destacam que as Instituições de Ensino Superior (IES) têm adotado o uso de plataformas digitais para o aperfeiçoamento gerencial dos seus processos internos, em virtude de que estas ferramentas disponibilizam informações, dados e recursos que cooperam significativamente nas tomadas de decisão e resolução de problemas.

Dentre os processos internos importantes em um IES, pode-se citar a administração de suas comissões. As comissões universitárias são estruturas de caráter consultivo, pedagógico ou organizativo, que reúnem colaboradores, como técnicos e servidores, para a realização de estudos, acompanhamento de atividades, organização de tarefas, elaboração de eventos e desenvolvimento de ações em torno de determinados temas. Exemplos de comissões comuns em IES públicas são: Comissão de Ética no Uso de Animais, Comissão de Pessoal Docente, Comissão de Monitorias, entre outras.

Neste sentido, o desenvolvimento de um *software* para o gerenciamento das comissões dentro de uma IES, baseado em uma aplicação *web*, afigura-se como um artifício para o aprimoramento organizacional. Como potenciais benefícios de uma implantação bem-sucedida de uma aplicação desta natureza, pode-se mencionar a melhoria na visibilidade das informações, eficiência na tomada de decisões e assertividade na resolução de problemas. Ainda, vale-se ressaltar o aumento na consistência e confiabilidade das informações, em função da centralização dos dados em um único ambiente.

1.1 Identificação do problema

O modelo atual para a gestão das comissões no Departamento de Computação e Sistemas (DECSI) é feito com o auxílio de planilhas eletrônicas. A Figura 1 apresenta um exemplo de uma planilha eletrônica para a Comissão de Laboratórios. Constata-se que uma comissão é composta por vários membros, em mandatos com durações distintas. Por intermédio da legenda, verifica-se que os mandatos são classificados como ativos, inativos ou pendentes.

Embora seja uma solução barata, a manipulação de planilhas no plano gerencial

COMISSÃO DE LABORATÓRIOS							Legenda
MEMBRO	Entrou em	Sai em	Prorrogado em	Vence em	Observações		Ativo
Professor 1	18/03/2016						Pendente
Professor 2	18/03/2016	**16/02/2017 (37ª)**					Encerrado
Professor 3	18/03/2016						
Professor 4	11/04/2017	11/04/2018					
Professor 5	14/02/2017	31/03/2017					
Professor 6	16/02/2017	Exorneração					
Professor 7	16/02/2017	16/02/2018					
Professor 8	18/09/2018	18/09/2020					
Professor 9							
Professor 10							

Figura 1 – Planilha eletrônica da Comissão de Laboratórios

Fonte: Produzido pelo autor

acarreta várias inconveniências para os usuários. Por exemplo:

- **Segurança:** com o armazenamento dos dados em uma planilha, existe um menor controle relacionado às políticas de acessos às informações. Ademais, os dados e arquivos das planilhas podem ser prejudicados caso ocorra algum problema à nível de *software* ou *hardware* no dispositivo;
- **Facilidade de uso:** para uma pequena quantidade de dados, o uso de planilhas é satisfatório. No entanto, à medida que o volume de informações cresce, a administração dos dados torna-se cada vez mais complexa e contraprodutiva;
- **Limitações na centralização dos dados:** o uso de diversas planilhas para a gestão das informações favorece a descentralização dos dados. Por conseguinte, a validade, atualidade e coesão de suas informações podem não refletir a realidade.

Além dos malefícios supracitados, a metodologia utilizada para o controle das comissões no DECSI não propicia uma validação adequada para inserções, modificações e deleções na base de informações. Vale-se destacar, ainda, que a usabilidade das planilhas, além da disposição visual dos seus dados, não favorecem uma jornada atrativa para o usuário durante o manuseio do *software*.

Seguramente, o cenário atual de gerenciamento das comissões não suporta requisitos de usabilidade, escalabilidade, integrações com outros sistemas, confiabilidade e robustez em níveis satisfatórios para esta incumbência tão crucial para o departamento.

1.2 Justificativa

O emprego de um sistema gerencial concernente às comissões do DECSI potencialmente agregará valor para o departamento. O aumento da celeridade nos processos de criação, modificação e visualização dos dados, são exemplos de ganhos para todas as partes envolvidas com estas estruturas.

Em relação a experiência de uso do *software*, uma aplicação *web* aprimora a acessibilidade, uma vez que, encontrando-se *online*, os usuários podem acessar os dados das comissões em qualquer lugar, seja em um dispositivo *desktop* ou *mobile*. Mais do que isso, o sistema apoiará os usuários nos processos de validação dos dados, evitando com que possíveis incongruências sejam armazenadas na base.

1.3 Objetivos

O objetivo principal deste trabalho é criar uma plataforma *web* que apoie o procedimento de gerenciamento das comissões do DECSI em substituição ao uso de planilhas eletrônicas.

1.3.1 Objetivos específicos

Os objetivos específicos deste trabalho são apresentados abaixo:

- Analisar as dificuldades da metodologia atual de gestão das comissões;
- Modelar uma plataforma para o gerenciamento das comissões do DECSI de acordo com as especificações dos usuários;
- Projetar estruturas que promovam a escalabilidade do sistema para sua utilização em outros departamentos da Universidade Federal de Ouro Preto (UFOP);
- Validar a aplicação proposta por meio de testes de *software*.

1.4 Metodologia

Com o propósito de alcançar os objetivos elucidados, foram adotados os seguintes passos:

- Levantamento de requisitos juntos as partes interessadas pelo *software*;
- Definição das tecnologias utilizadas para a prototipação e desenvolvimento do projeto;
- Implementação do *software* de acordo com as necessidades dos usuários;
- Realização de testes e levantamento de possíveis melhorias.

1.5 Organização do trabalho

O restante deste trabalho está estruturado como se segue. O Capítulo 2 apresenta a revisão bibliográfica utilizada para o desenvolvimento da solução. O Capítulo 3 discorre sobre a arquitetura e tecnologias do *software* desenvolvido. O Capítulo 4 apresenta os resultados obtidos. Por fim, o Capítulo 5 contém as conclusões obtidas acerca da construção da plataforma.

2 Revisão bibliográfica

Este capítulo apresenta a revisão da literatura sobre os conceitos abordados no desenvolvimento da plataforma.

2.1 Informatização de processos no ambiente universitário

A expansão dos sistemas de informação, sobretudo, os sistemas de informação *web*, atinge constantemente diversas organizações do âmbito público ou privado. De acordo com Laudon e Laudon (2011), um sistema de informação pode ser conceituado como um conjunto de componentes inter-relacionados que recuperam, armazenam, processam e distribuem informações com o objetivo de apoiar a tomada de decisões, controle e coordenação de uma organização. Em adição, esses sistemas também ajudam os colaboradores a analisar os problemas, criar novos produtos e visualizar assuntos complexos.

Segundo Davenport (2000), a tecnologia perpassou o papel de uma ferramenta auxiliar para a administração, mas tornou-se um setor vigoroso em si mesmo. Múltiplas esferas sociais, como as IES, utilizam as TICs de forma vertiginosa. Irawan, Foster e Tanner (2018) corroboram com o fato de que IES de todo mundo têm aderido aceleradamente o uso de serviços tecnológicos para o gerenciamento das suas atividades internas.

De acordo com Rocha Neto e Lima (2009), um sistema de informação amigável consolida-se como uma base para uma boa administração da universidade como um todo. Esses sistemas objetivam atender as necessidades de planejamento e gestão das IES. De forma geral, o uso dessas aplicações contribuem para a mitigação de esforços para o controle dos processos, além de reduzir drasticamente o tempo gasto para cadastramento e atualização de tarefas processuais. Por último, esses *softwares* podem incrementar a qualidade das tomadas de decisão e recuperação de informações para os diretores, gestores e secretários das instituições.

Conforme Pineda e Medina (2014), sistemas informacionais, como um meio conciliador entre as tecnologias disponíveis e as necessidades administrativas das instituições, são essenciais no contexto universitário. Para as autoras, esses sistemas devem ser práticos e fáceis de usar. Em vista disso, suas interfaces devem ser amigáveis e promotoras de uma boa experiência para o utilizador. Pineda e Medina (2014) também destacam que os bons *softwares* agregam benefícios como a promoção de uma maior qualidade de serviço oferecidos às comunidades interna e externa, bem como expandem a qualidade dos processos de avaliação de programas acadêmicos.

Como manifestado no tópico 1.1, a solução proposta por este trabalho visa a

construção de um ambiente usual e prático para os colaboradores do DECSI, com a substituição das planilhas eletrônicas por uma plataforma *web*. Consoante com Pineda e Medina (2014), a plataforma buscará mecanismos para facilitar a sua utilização e oferecer uma boa experiência para o usuário.

2.2 Sistemas correlatos

Esta seção aborda algumas aplicações relacionadas à plataforma desenvolvida. Foram feitas pesquisas de sistemas de gerenciamento de comissões no repositório de artigos do Simpósio Brasileiro de Sistemas de Informação[1] e na *web*, mas não foram encontrados resultados satisfatórios. Por conseguinte, os fatores para a seleção de produtos digitais correlatos baseou-se nas funcionalidades de gerenciamento de equipes.

2.2.1 ClickUp

O ClickUp[2] é uma ferramenta para gestão de projetos. Esta ferramenta visa a gestão de projetos aliada ao gerenciamento de equipes. Com este produto, é possível associar atividades para cada membro da equipe, estabelecer prazos para as atividades, estabelecer suas prioridades e obter relatórios sobre o progresso do projeto. Para uso pessoal, pode-se utilizar a versão gratuita do ClickUp. Entretanto, o acesso às funcionalidades para o gerenciamento de equipes estão disponíveis nos planos pagos[3], com a mensalidade mínima de 5 dólares/mês.

2.2.2 Asana

A Asana[4] é uma plataforma para o gerenciamento de projetos fundada em 2008. Este *software* possui o enfoque no planejamento de projetos, permitindo a criação de tarefas para membros de uma equipe e a possibilidade de estabelecer restrições de tempo para cada tarefa individualmente. A plataforma também permite a criação de regras personalizadas para a execução de cada tarefa, além do estabelecimento de prioridades entre as mesmas. O plano básico da Asana - direcionado para indivíduos e equipes que estão iniciando a gestão de projetos - é gratuito para sempre. Já os planos mais completos que contam, por exemplo, com a criação de regras para as tarefas, são pagos.

[1] Disponível em: <https://sol.sbc.org.br/index.php/sbsi/issue/archive>
[2] Disponível em: <https://clickup.com/>
[3] Disponível em: <https://clickup.com/pricing>
[4] Disponível em: <https://asana.com/pt>

2.2.3 Comparativo

Os *softwares* supramencionados não são focados na administração de comissões universitárias, mas no gerenciamento de projetos e equipes no geral. Desta maneira, muitas regras de negócio específicas das comissões não são contempladas por estes produtos. A Tabela 1 apresenta um quadro comparativo entre as ferramentas apresentadas nesta seção e a Insider.

Tabela 1 – Quadro comparativo

Funcionalidade	**ClickUp**	**Asana**	**Insider**
Controle de tarefas	X	X	
Controle de notificações	X	X	X
Controle de mandatos			X
Atribuição de pontuação para as comissões			X
Geração de certificado de participação			X
Manifestação de interesse em comissões			X

Fonte: Produzido pelo autor

2.3 Arquitetura de *software*

Apesar de não existir uma conceituação uniforme e amplamente adotada pela literatura em geral, o termo arquitetura de *software* aborda a divisão de um sistema em componentes, a responsabilidade destes e como eles interagem. A visão da arquitetura do *software* limita-se à estrutura macro do sistema. Destarte, um componente refere-se a um conjunto de classes, funções ou módulos relacionados.

Arquitetura de um sistema computacional ou programa é a estrutura ou um conjunto de estruturas que abarca os componentes do *software*, as propriedades externamente visíveis de cada um deles e o inter-relacionamento entre as partes (BASS; CLEMENTS; KAZMAN, 2003).

No livro *Fundamentals of Software Architecture: An Engineering Approach*, Richards e Ford (2020) teorizam que arquitetura de *software* fundamenta-se na combinação de 4 aspectos: (1) estrutura; (2) características; (3) princípios de projeto; e (4) decisões de projeto.

A estrutura resume-se ao estilo arquitetural usado no sistema (por exemplo, arquitetura em camadas ou microsserviços). Essa propriedade está associada com a topologia do sistema, com enfoque na distribuição dos componentes.

O segundo ponto compreende as características do *software*, tais como sua escalabilidade, confiabilidade, testabilidade, disponibilidade, manutenibilidade, segurança, etc.

O terceiro ponto engloba os princípios de projeto empregados para nortear a confecção da arquitetura. Por exemplo, os arquitetos podem definir que a comunicação entre os componentes do sistema só ocorrerão por meio de interfaces e que não existirão módulos compartilhados entre os componentes, de maneira a construir um sistema com baixo acoplamento.

Por último, as decisões arquitetônicas do projeto envolvem detalhes tecnológicos e conceituais da solução, como a separação dos componentes responsáveis pelo acesso ao banco de dados, pelo processamento das regras de negócio e apresentação dos dados para o usuário final.

Tendo em vista que a volatilidade é uma característica marcante nos produtos digitais atuais, é natural perceber a primacialidade na escolha de uma arquitetura que atenda os objetivos de negócio e suporte às mudanças exigidas pelos usuários. Bass, Clements e Kazman (2003) afirmam que a arquitetura destacará desde o início quais serão as decisões de projeto que impactarão intensamente o trabalho de engenharia de *software*. Arquiteturas bem elaboradas são premissas para sistemas relevantes e manuteníveis.

2.3.1 Arquitetura em Três Camadas

A arquitetura em três camadas é largamente adotada no desenvolvimento de sistemas corporativos. Segundo Valente (2020), as três camadas dessa arquitetura são:

- **Interface com o Usuário:** essa camada, também chamada de camada de apresentação, responsabiliza-se pela interação entre o usuário e o *software*. Ela trata os eventos de apresentação e coleta de dados com o utilizador do sistema. O aplicativo desta camada pode ser executado, por exemplo, em em um ambiente *desktop* ou na *Web*, geralmente baseado em *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e *JavaScript* (JS).
- **Banco de Dados:** denominada como camada de dados, possui a função de armazenar e manipular os dados do sistema. A camada de dados pode conter um Sistema de Gerenciamento de Banco de Dados (SGBD), como PostgreSQL[5], MySQL[6] ou Oracle[7] ou um servidor de banco de dados NoSQL, como MongoDB[8] ou Cassandra[9].
- **Lógica de Negócio:** por vezes chamada de camada de aplicação, é responsável pela implementação das lógicas de negócio do *software*. Por exemplo, uma regra de negócio na plataforma Insider especifica que uma comissão não pode ser criada sem uma pontuação.

[5] Disponível em: <https://www.postgresql.org>
[6] Disponível em: <https://www.mysql.com>
[7] Disponível em: <https://www.oracle.com/br/database>
[8] Disponível em: <https://www.mongodb.com>
[9] Disponível em: <https://cassandra.apache.org>

O modelo de arquitetura em três camadas favorece a separação lógica e física das funcionalidades do sistema. Assim, o desenvolvimento dos componentes em cada uma das camadas pode ser feito de um jeito ágil e independente. Outrossim, os aplicativos de apresentação, lógica de negócios e banco de dados podem ser providos por servidores distintos, de maneira que o ajuste da escala para cada uma das camadas é feito individualmente. Por conseguinte, a indisponibilidade de uma camada não impacta diretamente no funcionamento e performance das outras.

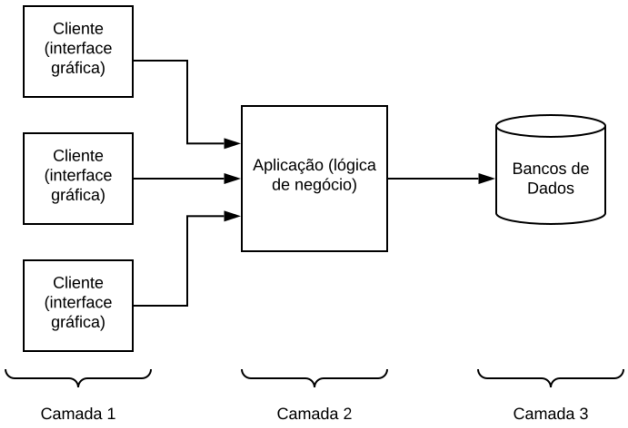


Figura 2 – Arquitetura em três camadas

Fonte: Valente (2020, p. 235)

A Figura 2 apresenta um exemplo de arquitetura em três camadas onde a interface exibida ao cliente é uma interface gráfica.

2.3.2 Arquitetura REST

A larga adoção de APIs que seguem o estilo arquitetural REST no *design* de aplicações *web* é uma realidade. De fato, o projeto de APIs intuitivas e que estejam de acordo com as necessidades dos seus clientes é um fator fundamental na elaboração de sistemas digitais.

O conceito do estilo arquitetural REST para sistemas hipermídia distribuídos foi formulado por Roy Fielding, em sua tese de doutorado (FIELDING, 2000). De acordo com o autor, a motivação para o desenvolvimento do REST foi construir um modelo arquitetural padrão para a estruturação de protocolos da *Web*.

Os componentes principais de um sistema que segue a arquitetura REST são os clientes e servidores. O componente cliente deve providenciar interfaces para o usuário, enquanto o componente servidor é encarregado pelos dados necessários para o cliente. Em sistema *web*, um aplicativo cliente é renderizado no *browser* e envia solicitações *HyperText Transfer Protocol* (HTTP) para um servidor, que acessa/manipula os dados e entrega uma

resposta para a solicitação do cliente. A Tabela 2 explana sobre os métodos HTTP mais utilizados em um API REST.

Tabela 2 – Métodos HTTP

Método	**Descrição**
GET	Solicita um recurso. Esse recurso pode ser um arquivo HTML, *eXtensible Markup Language* (XML), *Javascript Object Notation* (JSON), etc.
POST	Cria um recurso. Os dados do recurso são enviados no corpo da requisição.
PUT	Requisita que um recurso seja armazenado na *Uniform Resource Identifier* (URI) fornecida. Caso o recurso exista, ele é atualizado. Caso contrário, ele é criado.
DELETE	Deleta o recurso apontado.
PATCH	Requisita a aplicação de modificações parciais em um recurso especificado.

Fonte: Antunes (2019)

Fielding (2000) destaca a importância do particionamento dos papéis de clientes e servidores em uma arquitetura REST, pois tal característica adiciona a flexibilidade para implementação independente dos módulos do sistema. Todavia, o autor explana que, para o sucesso de um projeto que se orienta nas diretrizes do REST, as interfaces de comunicação entre os componentes devem ser estáveis e uniformes, de modo que haja um contrato consistente entre os clientes e servidores.

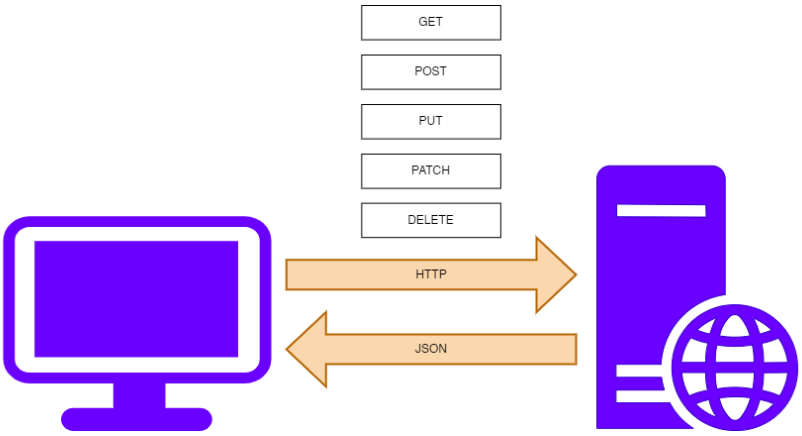


Figura 3 – Arquitetura REST

Fonte: Produzido pelo autor

A Figura 3 exemplifica a comunicação inter-sistemas seguindo o modelo REST. Pode-se observar que, mediante uma requisição HTTP, o cliente solicita uma ação no servidor. O servidor processará a solicitação e entregará uma resposta em um formato cognoscível ao solicitante, como JSON ou XML.

2.3.3 Arquitetura Limpa

A entrega de valor e geração de impactos positivos para as partes interessadas em um sistema são verdadeiros regentes em equipes construtoras de *software*. Para Lemos (2022), uma questão crucial para o desenvolvimento de *software* centra-se na minimização

do *lead time*, ou seja, o tempo de entrega de uma funcionalidade ou correção demandadas pelos *stakeholders*. Ainda, a equipe deve buscar estratégias de geração de valor para o negócio por um longo tempo, sustentavelmente. Entende-se por sustentável a construção de um *software* que não ultrapasse as restrições de tempo e custo calculados para o projeto.

Perante o exposto, Martin, Grenning e Brown (2018) idealizaram a Arquitetura Limpa, ou *Clean Architecture* (CA). Martin, Grenning e Brown (2018) postulam que uma arquitetura de qualidade deve assegurar o desacoplamento do sistema com estruturas externas, como *frameworks*, tecnologias de interface gráfica ou banco de dados. Em outros termos, as políticas internas de funcionamento do *software*, como suas entidades e regras de negócio, não devem estar associadas diretamente com elementos externos, mas, sim, auxiliada por eles. Para este propósito, a CA propõe a segmentação de um sistema em quatro camadas: ***Enterprise Business Rules***, ***Application Business Rules***, ***Interface Adapters*** e ***Framework & Drivers***, conforme a Figura 4.

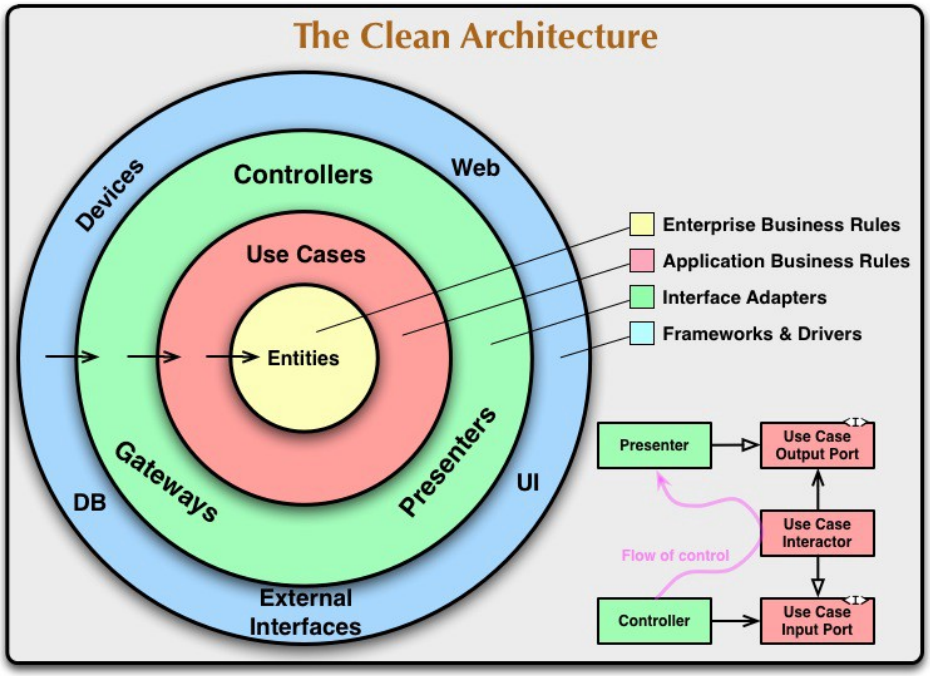


Figura 4 – Camadas da *Clean Architecture*

Fonte: Martin, Grenning e Brown (2018, p. 196)

2.3.3.1 *Enterprise Business Rules*

De acordo com Martin, Grenning e Brown (2018), a camada *Enterprise Business Rules* - Regras de Negócios da Empresa, em tradução livre - deve acomodar as regras críticas e sensíveis de um domínio. As entidades, especificadas no centro do diagrama da Figura 4 devem implementar as regras de mais alto nível do sistema, isto é, as regras com menor probabilidade de sofrer alterações frente uma mudança exterior. Em outras palavras,

as entidades não precisam sofrer uma mutação caso seja empregada uma nova tecnologia de banco de dados ou interface gráfica do *software*. Uma entidade do sistema pode ser uma classe com métodos, no Paradigma de Orientação a Objetos, ou uma coleção de funções e estruturas de dados, conquanto que as políticas mais gerais do projeto estejam nelas. A camada *Enterprise Business Rules* não deve depender de implementações de nenhuma das outras camadas.

2.3.3.2 *Application Business Rules*

A camada *Application Business Rules* - Regras de Negócios da Aplicação, em tradução livre - contém as regras de negócio basilares. Essa camada age na orquestração das entidades da camada *Enterprise Business Rules* para implementação dos casos de uso do sistema, ou seja, nos procedimentos de alto nível que os atores do *software* podem efetivar. Os casos de uso efetuam a *dança das entidades*, visto que esses agentes atuam na instanciação e nas chamadas de operações das entidades para alcançarem os desígnios das regras de negócio (MARTIN; GRENNING; BROWN, 2018). Semelhantemente à camada *Enterprise Business Rules*, espera-se que mudanças em componentes externos ao sistema não impactem a camada *Application Business Rules*. Observa-se, pela Figura 4, que a camada dos casos de uso deve depender, unicamente, da camada das entidades.

2.3.3.3 *Interface Adapters*

O código da *Interface Adapters* - Adaptadores de Interface, em tradução livre - deve prover um grupo de adaptadores que transformam os dados advindos de fontes externas para os formatos mais apropriados para os casos de usos e entidades, além de traduzir os dados provenientes das camadas mais internas do sistema para a sua borda. Exemplificativamente, essa camada pode conter um adaptador que mapeia os dados resultantes da execução de um caso de uso e os disponibiliza apropriadamente para um *gateway* de pagamentos online ou um serviço de *e-mails*, de tal modo que as regras de negócio do sistema não estejam acopladas com a infraestrutura.

2.3.3.4 *Frameworks & Drivers*

A camada mais exterior da CA é formada por bibliotecas, *frameworks*, banco de dados, entre outros apetrechos. As ferramentas dessa camada dão sustentação à operação do *software*. Segundo Martin, Grenning e Brown (2018), os componentes dessa camada são apenas detalhes para o funcionamento da aplicação. Em relação à volatilidade, esses módulos são os mais sujeitos à mudança. Logo, devem ser implementados de forma que sejam facilmente substituíveis.

2.3.3.5 A Regra de Dependência

Um conceito muito importante da Arquitetura Limpa é a Regra de Dependência. Essa regra enuncia que o código-fonte de uma camada mais externa só pode depender de classes ou estruturas das camadas mais internas, em consonância com as setas exibidas na Figura 4. Dessarte, os trechos de código das camadas mais internas não podem conter referências para as camadas mais externas. Isso viabiliza o baixo acoplamento entre as políticas de alto e baixo nível da aplicação.

Os projetos que seguem a CA fazem uso intensivo de princípios de projeto como **Inversão de Dependência** e **Responsabilidade Única**. Para tal, a comunicação de códigos-fonte das camadas internas e externas do sistema ocorre via interface, onde as estruturas mais externas implementam os contratos estabelecidos pelas classes mais internas.

Lemos (2022) disserta que, além de todas as particularidades supracitadas, a Arquitetura Limpa favorece a testabilidade das regras de negócio, pois essas não são dependentes da Interface Gráfica, servidor *web*, banco de dados ou qualquer outra unidade. Ademais, a CA fundamenta-se no conceito de arquitetura "gritante", onde o tema da aplicação e os casos de uso que ela suporta devem ser evidentes em sua estrutura.

2.4 Experiência do Usuário

O termo Experiência do Usuário, do inglês *User Experience* (UX), foi originado por Don Norman com o intuito de englobar todos os detalhes da experiência do usuário no uso de algum produto (BULEY, 2013). A concepção de sistemas digitais intuitivos, focados em usabilidade, assumiu um papel fundamental na indústria moderna. O conceito de UX perpassou a ideia de simplesmente desenvolver belas interfaces. Seu objetivo é viabilizar, de forma clara e agradável, todas as informações necessárias para o usuário em toda a sua cadeia de eventos durante a utilização do aplicativo.

Ocasionalmente, os termos UX e Usabilidade são identificados como sinônimos. No entanto, a Usabilidade pode ser tipificada como um subdomínio da UX, cuja conceituação é a facilidade de utilização de um serviço ou produto, onde os atores do sistema consigam realizar suas atividades sem dificuldades ou induzidos ao erro. Outra definição para Usabilidade é:

> Usabilidade é uma medida de quanto um sistema computacional... colabora com o aprendizado; auxilia os aprendizes a se lembrarem sobre o conteúdo que aprenderam; mitiga a probabilidade de erros; permite que se tornem eficientes; e os deixa satisfeitos com o *software*. (PRESSMAN; MAXIM, 2016),

Para Morville (2004), uma boa experiência do usuário é evidenciada em sete atributos: útil (*useful*), encontrável (*findable*), utilizável (*usable*), credibilizável (*credible*),

desejável (*desirable*), valiosa (*valuable*) e acessível (*accessible*).

- **Útil**: um produto ou serviço deve oferecer um benefício ou facilitar a execução de uma atividade. Nesse sentido, o conhecimento das necessidades do usuário é extremamente necessário para adequação do produto para cada perfil de utilizador.
- **Encontrável**: um produto ou serviço deve possuir uma interface clara, com objetos distinguíveis e localizáveis, para que os usuários possam encontrar o que precisam. A hierarquia e disposição dos elementos visuais devem ser orgânicos e assimiláveis pelos utilizadores da aplicação.
- **Utilizável**: este atributo está relacionado à facilidade de uso do sistema. O usuário deve conseguir concluir suas atividades com eficiência e eficácia. Caso a utilização do *software* seja confusa e/ou difícil, o utilizador pode se sentir frustrado, o que prejudica gravemente a sua experiência.
- **Credibilizável**: um produto ou serviço deve passar confiança para o usuário. Esse atributo está associado com a convicção do cliente de que suas informações sensíveis não serão disponibilizadas sem a sua permissão e que os seus dados permanecerão seguros.
- **Desejável**: a estética, imagem, identidade, entre outros elementos do *design* emocional, devem despertar uma boa impressão no usuário e, consequentemente, estimulá-lo à utilização do produto ou serviço.
- **Valiosa**: a experiência do usuário será valiosa quando o *software* oferecer mecanismos para aquisição de dados, tomadas de decisão e resolução de problemas. O produto ou serviço deve agregar valor para as partes interessadas.
- **Acessível**: um produto digital deve diligenciar-se para o aumento da sua usabilidade, fazendo com que uma comunidade ainda maior de usuários consigam usufruir da solução.

Sumariamente, é possível perceber a importância da UX na construção de um *software*. A negligência dessa área na fabricação de uma aplicação pode ser prejudicial para o seu sucesso.

2.5 Testes

Segundo Valente (2020), a introdução das atividades de teste no ciclo de desenvolvimento de *software* é fundamental para a valorização do sistema computacional. Para o autor, por ser uma construção inerentemente complexa, *softwares* precisam ser testados

para que eventuais erros não cheguem aos usuários finais e causem prejuízos incalculáveis. Nessa mesma linha de raciocínio, Feathers (2004) enuncia que "código sem testes é um código ruim" (tradução própria).

Valente (2020) enumera alguns benefícios da utilização de testes em um sistema computacional:

1. Localização de *bugs*: uma ampla cobertura de testes favorece a localização de *bugs* em um *software* ainda na fase de desenvolvimento;
2. Rede de proteção contra regressões: a escrita de bons testes pode prevenir que regressões sejam anexadas ao software. Logo, a equipe de desenvolvimento tem a liberdade de modificar um projeto tranquilamente, sem a preocupação de que mudanças no código danificarão outras partes do sistema;
3. Documentação e especificação do código de produção: testes bem escritos são fontes de informações valiosas sobre as funcionalidades do sistema em produção, pois especificam detalhadamente o funcionamento do código.

De acordo com Pressman e Maxim (2016), uma estratégia de testes de *software* deve comportar testes de baixo nível, responsáveis pela verificação da implementação de pequenos segmentos de código, e testes de alto nível, que validam os principais casos de uso do sistema, em comum acordo com os requisitos do cliente.

Conforme Valente (2020), os testes podem ser divididos em três grupos: **testes de unidade**, **testes de integração** e **testes de sistema**. A Figura 5 apresenta a **pirâmide de testes**, que os particiona em concordância com sua granularidade.

Ao observar a Figura 5, depreende-se que os testes de unidade formam a base da pirâmide. Desta maneira, eles devem representar a maioria dos casos de teste de um sistema, sendo também os mais rápidos, de menor custo e menor granularidade no *software*. Em contrapartida, os testes de sistema representam a minoria dos casos de teste de uma aplicação, sendo os mais caros, lentos e de maior granularidade.

2.5.1 Testes de Unidade

Testes de unidade são testes automatizados de pequenos segmentos de código-fonte que são testados isoladamente (VALENTE, 2020). Testes de unidade devem testar todos os caminhos independentes dentro da estrutura de controle, de forma que seja assegurado que todas as instruções em um módulo tenham sido processadas pelo menos uma vez (PRESSMAN; MAXIM, 2016). As condições limite do componente testado também devem ser avaliadas no teste para garantir que o módulo opere apropriadamente nas fronteiras.

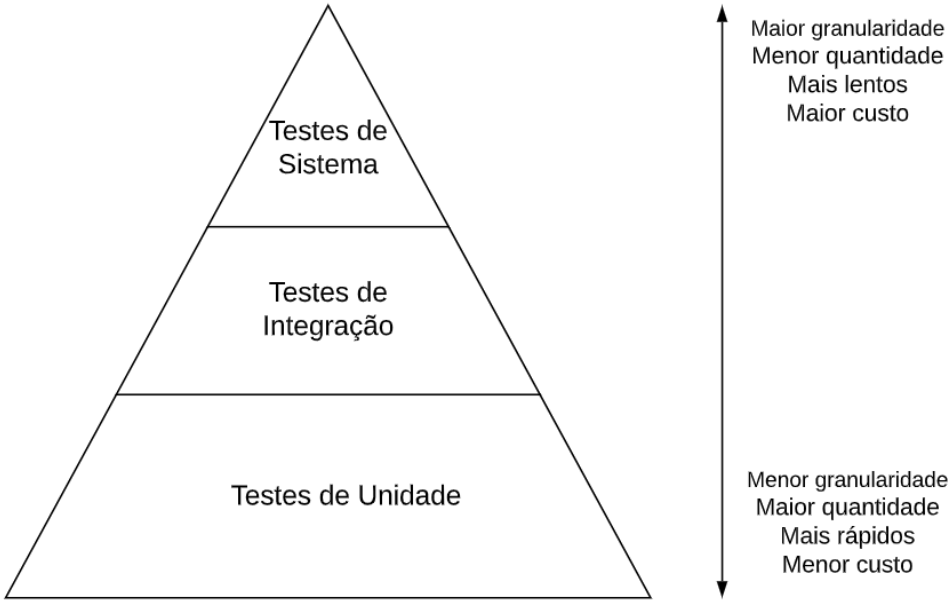


Figura 5 – Pirâmide de Testes

Fonte: Valente (2020, p. 259)

Por fim, os autores orientam que todos os caminhos de manipulação de erro sejam cobertos pelos testes unitários. A Figura 6 sintetiza as temáticas de casos de teste para um módulo.

2.5.2 Testes de Integração

Segundo Valente (2020), os testes de integração são os responsáveis pela verificação completa de uma funcionalidade ou transação de um sistema computacional. Após a fase de construção dos testes de unidade, que garantem o funcionamento individual dos módulos da aplicação, os testes de integração objetivarão certificar que os componentes do sistema funcionem quando estiverem em conjunto. Por serem maiores e abordarem mais de um módulo, os testes de integração são mais complexos de serem implementados e demoram mais tempo para serem executados. Em decorrência disso, eles são chamados com menor frequência.

2.5.3 Testes de Sistema

Os testes de sistema estão localizados no topo da pirâmide de testes. Eles simulam a utilização de um sistema por um cliente real. Essa classe de testes também é denominada por testes de interface ou *end-to-end* (ponta-a-ponta, em tradução livre).

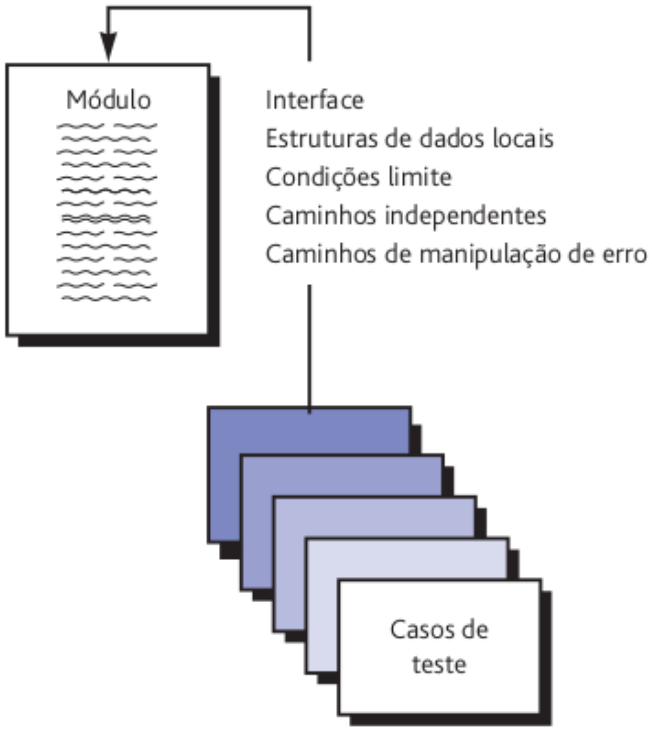


Figura 6 – Testes de unidade

Fonte: Pressman e Maxim (2016, p. 474)

2.6 *Test-Driven Development*

De acordo com Sommerville (2011), o TDD (Desenvolvimento Dirigido a Testes, em tradução própria), é uma abordagem para o desenvolvimento de *software* em que são intercaladas as fases de testes e desenvolvimento de código. TDD é uma das práticas de programação preconizadas por Beck e Andres (2004). Fundamentalmente, uma funcionalidade e o seu respectivo teste são desenvolvidos de maneira incremental.

Conforme (VALENTE, 2020), o TDD divide a implementação de uma funcionalidade em um ciclo de três estados: testes falhando, testes passando e refatoração. Este ciclo é exposto na Figura 7.

O primeiro estado do ciclo (vermelho) é escrever um teste unitário para a funcionalidade, mas que falhará, pois a implementação funcional ainda não foi realizada. O teste especificará as regras e caminhos para a escrita da funcionalidade. O segundo estado (verde) é escrever um código-fonte para o módulo sob teste. Por fim, o estado de refatoração (amarelo) focaliza a melhoria da qualidade do código, removendo duplicações, aplicando princípios e padrões de projeto, etc.

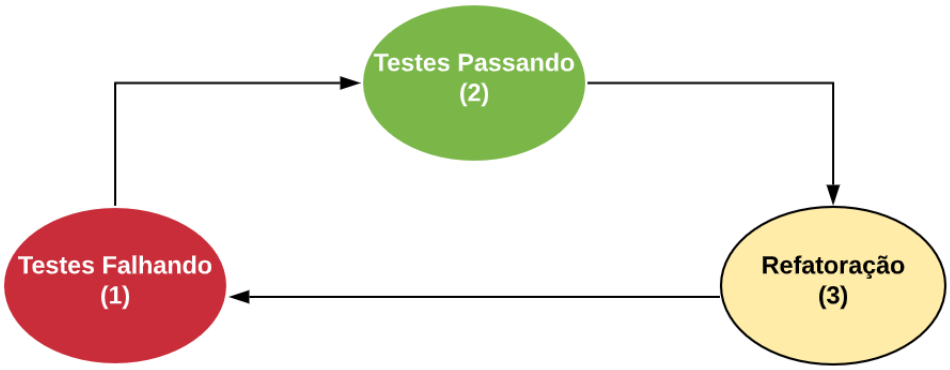


Figura 7 – Ciclos do TDD

Fonte: Valente (2020, p. 291)

2.7 Considerações finais

O presente capítulo abordou os conceitos teóricos que fundamentaram a elaboração deste trabalho. No Capítulo 3, serão apresentados os requisitos, casos de uso e as tecnologias utilizadas para o desenvolvimento da plataforma.

3 Desenvolvimento

Este capítulo descreve o desenvolvimento da aplicação proposta, abrangendo os detalhes de implementação, tecnologias e arquitetura do sistema computacional. Além disso, são mostrados os casos de uso da aplicação e a estrutura do banco de dados.

3.1 Visão geral

A plataforma Insider, *software* desenvolvido neste trabalho, foi concebida com o intuito de facilitar o cadastro e acompanhamento das comissões do DECSI. As dificuldades do processo atual de gerenciamento das comissões, por meio de planilhas eletrônicas, mostra-se insatisfatório para diversas necessidades dos usuários, além de ser contraproducente.

O domínio da solução está sistematizada nos seguintes atores: **Administradores**, **Secretários** e **Membros**. Os administradores e secretários possuem as prerrogativas de cadastro, atualização e deleção das comissões. Porém, os administradores possuem um acesso generalizado, de tal modo que conseguem gerenciar as comissões com membros de departamentos distintos, enquanto os secretários estão restringidos às comissões internas dos seus respectivos departamentos. Um membro pode visualizar os seus mandatos, manifestar interesse em comissões e obter comprovantes de participação.

A criação da Insider baseou-se nas diretrizes da Engenharia de Usabilidade. Conforme Nielsen (1994), a Engenharia de Usabilidade é um ramo da ergonomia aplicada ao desenvolvimento de *software* e relaciona-se ao modo de construção de uma aplicação de fácil uso. Nesta perspectiva, a Engenharia de Usabilidade pode ser tipificada nos seguintes pontos:

- **Facilidade de aprendizado:** deve ser descomplicado para o usuário aprender a utilizar o sistema. Esta peculiaridade está diretamente atrelada ao tempo e esforço requerido para que o utilizador do *software* atinja um boa performance no uso da aplicação;
- **Retenção do aprendizado com uso intervalado:** a interface gráfica deve possibilitar que um usuário - neste contexto, esporádico - seja capaz de usar o sistema informatizado adequadamente;
- **Produtividade no exercício das atividades:** a interface do sistema de *software* deve permitir uma boa performance ao utilizador no exercício de suas atividades. Entretanto, ressalta-se que o termo performance adotado nesta esfera não correlaciona-se ao desempenho computacional, mas com a interação do usuário com a aplicação;

- **Prevenção de erros do usuário:** o sistema deve oferecer mecanismos de prevenção de erros. Ratifica-se que esses são potenciais erros que poderiam ser cometidos pelo usuário;
- **Satisfação:** o usuário deve gostar de usar o sistema. O *software* deve prover os requisitos necessários que satisfaçam aos anseios do seu utilizador. Embora a satisfação seja caracterizada pela subjetividade e pessoalidade, é muito importante avaliar esse aspecto na concepção de um produto digital.

3.2 Requisitos da plataforma

Segundo Sommerville (2011), requisitos de *software* são as funcionalidades e restrições previstas para um sistema computacional. Estes requisitos são qualificados em duas categorias: requisitos funcionais e não-funcionais:

- Requisitos funcionais: são as funções, ou casos de uso, que o produto deve oferecer aos usuários explicitamente.
- Requisitos não-funcionais: estabelecem as restrições para o funcionamento da aplicação, como o desempenho, robustez, portabilidade, confiabilidade, etc.

3.2.1 Requisitos funcionais

Os requisitos funcionais para a plataforma Insider foram categorizados de acordo com os atores do sistema. Em conformidade com o exposto em 3.1, as funcionalidades de cada um dos atores foram sintetizadas na Tabela 3.

Tabela 3 – Atores da Insider

Ator	**Descrição**
Administrador	Gerenciar as comissões de todos os departamentos do sistema. Este gerenciamento está associado com o cadastro, atualização, inativação e deleção das comissões.
Secretário	Controlar as comissões específicas do seu departamento.
Membro	Visualizar os seus mandatos, manifestar interesse em comissões e obter comprovantes de participação.

Fonte: Produzido pelo autor

3.2.1.1 Administradores

Os requisitos para os administradores do sistema são listados abaixo:

- **Gerenciamento das comissões interdepartamentais**: um administrador deve ser capaz de cadastrar, editar, encerrar e deletar comissões de um ou mais departamentos;

- **Gerenciamento dos mandatos**: um administrador deve estar habilitado para adicionar, prorrogar e encerrar mandatos das comissões;
- **Controle dos interesses**: um administrador deve ser estar apto para aceitar ou negar a entrada de um membro que está na lista de interessados em uma comissão;
- **Controle de acessos**: um administrador deve ser capaz de alterar o papel de um usuário para membro, secretário ou administrador.

3.2.1.2 Secretários

De forma análoga aos requisitos apresentados no tópico anterior - 3.2.1.1, os secretários são responsáveis pelo gerenciamento das comissões, mandatos e interesses dos seus departamentos. Todavia, o secretário só pode alterar o papel de um usuário para membro ou secretário.

3.2.1.3 Membros

Os requisitos para os membros do sistema são listados abaixo:

- **Obtenção de comprovantes**: os membros devem ser capazes de solicitar comprovantes de participação em comissões;
- **Controle dos interesses**: os membros devem ser capazes de manifestar ou retirar o seu interesse em participar de determinadas comissões.

3.2.2 Compartilhados

Os requisitos compartilhados entre todos os usuários do sistema são:

- ***Login***: um usuário deve ser capaz de realizar o *login* na aplicação;
- **Edição de perfil**: um usuário deve estar habilitado para alterar as informações do seu perfil;
- **Notificações**: um usuário deve ser capaz de visualizar ou excluir suas notificações;
- **Listagem das comissões**: um usuário deve ser capaz de visualizar todas as comissões do sistema;
- **Listagem dos usuários**: um usuário deve ser capaz de visualizar o perfil de todos os outros usuários do sistema.

Por último, um usuário externo deve ser capaz de se cadastrar na plataforma. Contudo, o papel estabelecido para o usuário em seu contato inicial com o sistema será o de membro.

3.2.3 Requisitos não-funcionais

A especificação dos requisitos não-funcionais da aplicação são apresentados abaixo:

- **Usabilidade**: a interface deve ser intuitiva e de fácil uso;
- **Responsividade**: a interface deve ser adaptável para telas de computadores, *notebooks* e dispositivos móveis.

3.2.4 Diagrama de Casos de Uso

Casos de uso e diagramas de casos de uso auxiliam na determiniação das funcionalidades e características do *software* sob o prisma do usuário (PRESSMAN; MAXIM, 2016). Os casos de uso gerais da Insider são apresentados na Figura 8. Salienta-se que o acesso aos casos de uso apresentados no diagrama está condicionado à autenticação do usuário na plataforma.

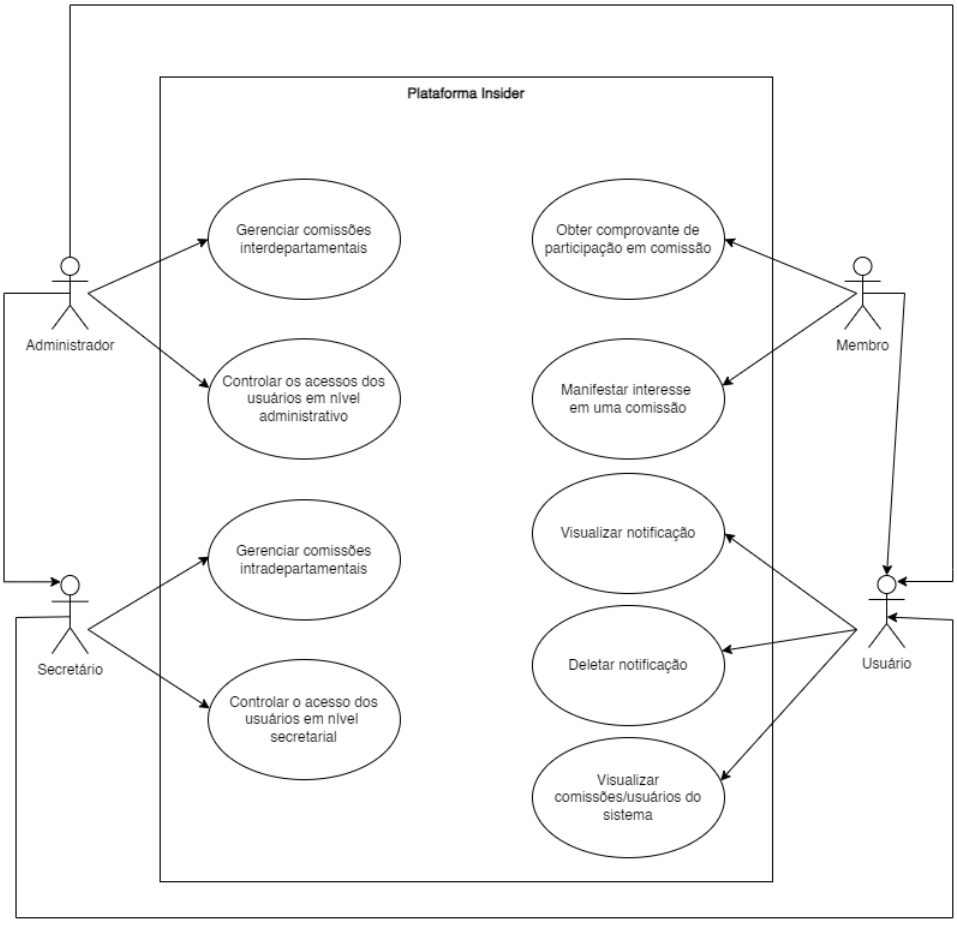


Figura 8 – Diagrama de casos de uso

Fonte: Produzido pelo autor

3.3 *Back-end*

Em um sentido macro, a arquitetura da Insider está divida em três camadas. O *back-end* está correlacionado com a camada de **Lógica de Negócios**.

Executado no lado do servidor, o *back-end* da Insider é responsável por operacionalizar as regras de negócio, bem como comunicar com o banco de dados e assegurar a integridade das informações. Nesta seção são apresentadas as tecnologias, organização do código e metodologia de desenvolvimento usadas no *back-end*.

3.3.1 *C#* e .NET

O *back-end* da aplicação foi implementado com a linguagem C#[1] em conjunto com .NET[2]. O .NET é uma plataforma de desenvolvimento de *software* livre gratuita para criação de aplicativos móveis, APIs, aplicativos em nuvem, *Internet of things* (IoT), entre outros[3].

A justificativa para a escolha destas tecnologias baseou-se na facilidade de codificação proporcionados pelo C# e .NET, além do seu suporte à Orientação a Objetos (OO). Adicionalmente, a comunidade e o ecossistema para esta linguagem estão estabelecidos em diversas soluções de mercado[4]. Outro fator importante é a possibilidade de estender as funcionalidades de uma solução em .NET, com a sua disponibilidade de uma ampla quantidade de pacotes.

3.3.2 SonarQube

O gerenciamento da qualidade do código é uma responsabilidade basilar para a melhoria contínua de um projeto de *software*. Em seu trabalho, Pina (2013) traça a correlação entre uma boa gestão dos débitos técnicos em sistema computacional com a qualidade, facilidade de manutenção e evolução do produto digital.

A adoção de ferramentas para a análise do código-fonte auxilia a gerência da qualidade do *software*. De forma geral, uma ferramenta de análise estática procura trechos de código que não estejam de acordo com as regras de qualidade predefinidas para o projeto.

Nesta perspectiva, a opção empregada pela plataforma Insider foi o SonarQube[5]. O SonarQube é um servidor *open source* para a análise estática do código. Esta ferramenta provê um relatório sobre *bugs*, cobertura de testes, vulnerabilidades e *code smells* em uma

[1] Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/>
[2] Disponível em: <https://dotnet.microsoft.com/en-us/>
[3] Disponível em: <https://docs.microsoft.com/pt-BR/dotnet/core/introduction>
[4] Disponível em: <https://dotnet.microsoft.com/en-us/platform/customers>
[5] Disponível em: <https://www.sonarqube.org/>

aplicação. A Figura 9 apresenta a interface do SonarQube com as métricas analisadas no *back-end*.

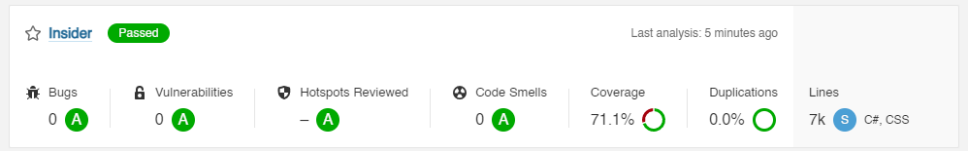


Figura 9 – SonarQube do *back-end*

Fonte: Produzido pelo autor

Pode ser observado que não foram apresentadas *bugs*, *code smells* ou duplicações no projeto analisado. Vale ressaltar que arquivos de configuração da solução também foram analisados e, por isso, a cobertura de testes ficou em 71.1%. Os testes da aplicação focaram nas camadas de casos de uso e nos endpoints da API.

A Figura 10 destaca a cobertura de testes da camada de casos de uso do *back-end*, o que cria uma rede de proteção contra introdução de *bugs* nas regras de negócio do sistema. Adicionalmente, foram incluídos testes em torno dos *endpoints* do *back-end*, protegendo os contratos de entrada/saída da API. A Figura 11 apresenta a cobertura de testes dessa camada. Pode ser observado que a cobertura de testes nessas camadas foi de 100%.

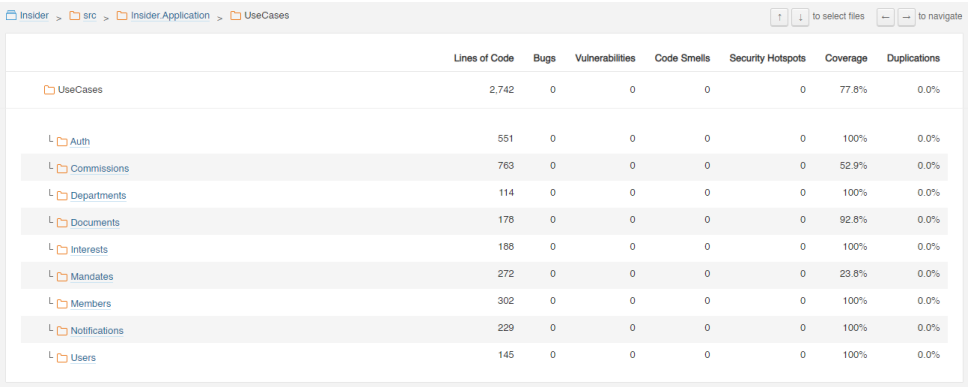


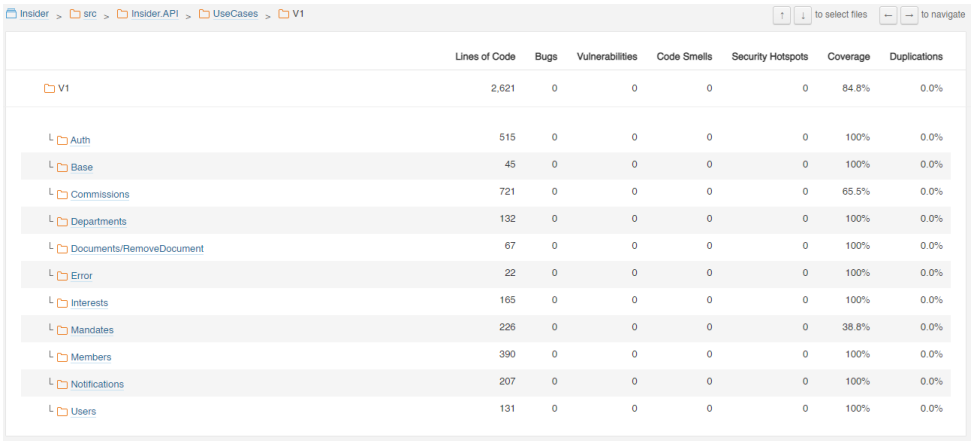
Figura 10 – Análise da camada de casos de uso

Fonte: Produzido pelo autor

3.3.3 Amazon Simple Storage Service

O uso da Computação em Nuvem tem sido objeto de estudo recorrente na computação moderna. Segundo Varella (2019), os recursos computacionais neste modelo são adquiridos sob demanda por meio da Internet e conforme as necessidades do negócio. Por esta perspectiva, não existe a obrigatoriedade da compra de servidores físicos, mas o aluguel destes mecanismos de provedores em nuvem, como a *Amazon Web Services* (AWS)[6].

[6] Disponível em: <https://aws.amazon.com/pt/>



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
V1	2,621	0	0	0	0	84.8%	0.0%
Auth	515	0	0	0	0	100%	0.0%
Base	45	0	0	0	0	100%	0.0%
Commissions	721	0	0	0	0	65.5%	0.0%
Departments	132	0	0	0	0	100%	0.0%
Documents/RemoveDocument	67	0	0	0	0	100%	0.0%
Error	22	0	0	0	0	100%	0.0%
Interests	165	0	0	0	0	100%	0.0%
Mandates	226	0	0	0	0	38.8%	0.0%
Members	390	0	0	0	0	100%	0.0%
Notifications	207	0	0	0	0	100%	0.0%
Users	131	0	0	0	0	100%	0.0%

Figura 11 – Análise na camada dos *endpoints* da API

Fonte: Produzido pelo autor

A Computação em Nuvem é disruptiva, visto que ela traz mudanças profundas em termos de tecnologia e custo para uma organização (VERAS, 2015). Varella (2019) atesta que níveis satisfatórios de escalabilidade, alta disponibilidade, elasticidade e confiabilidade podem ser alcançados com esta nova forma de computação.

A solução empregada para o armazenamento dos recursos estáticos da plataforma Insider, como os documentos advindos das comissões e as imagens dos usuários, foram armazenados em nuvem no serviço *Amazon Simple Storage Service*, também chamado de *Amazon S3*[7].

A AWS oferece um plano gratuito com um período de duração de 12 meses para o acesso ao *Amazon S3*. Após este período, pode-se escolher um plano pago adequado com as necessidades do cliente. Por exemplo, o plano ***S3 Standard***, focado no armazenamento de uso geral e para dados acessados frequentemente, possui o preço de 0,0405 dólares por gigabyte para os primeiros 50 terabytes/mês.

3.3.4 Tarefas em segundo plano

As verificações das datas de expiração das comissões e mandatos são feitas com o auxílio de um serviço agendador de tarefas, ou *cron job*. Este recurso é ofertado gratuitamente pelo .NET[8].

As tarefas em segundo plano são executadas diariamente, na madrugada, onde espera-se que o número de acessos ao sistema seja menor. Caso um mandato ou comissão esteja expirando, o serviço altera automaticamente o estado destas entidades para pendente. Entidades com esse estado são destacadas no *front-end* da aplicação, para que

[7] Disponível em: <https://aws.amazon.com/pt/s3/>

[8] Disponível em: <https://docs.microsoft.com/pt-br/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-6.0&tabs=visual-studio>

os responsáveis pelas comissões tomem as devidas providências para resolução dessas pendências.

3.3.5 *Logs*

Clemente (2008) define o termo *log* como um classe de registros com marcação temporal, que suporta apenas inserção, para representação de eventos que aconteceram em um equipamento de rede ou computador. Esta definição também pode ser estendida para sistemas computacionais. De acordo com Müller (2013), a partir das informações advindas dos *logs*, é possível detalhar o comportamento de uma solução, bem como identificar vulnerabilidades e elaborar planos para resoluções de problemas.

Os *logs* da aplicação *back-end* da Insider são administrados pelo *KissLog*[9]. Esta ferramenta capta os detalhes de uma requisição para o sistema, como duração, fonte e sua resposta. Os *logs* gerados são granularizados nas categorias de informação, aviso, erro ou crítico. As tratativas para os *logs* de erro ou críticos podem ser estabelecidas pelos administradores do *software*. Destaca-se que informações sensíveis, como senhas de usuário em uma operação de *login*, são criptografadas no *KissLog*. A Figura 12 apresenta o uso desta ferramenta pela Insider.

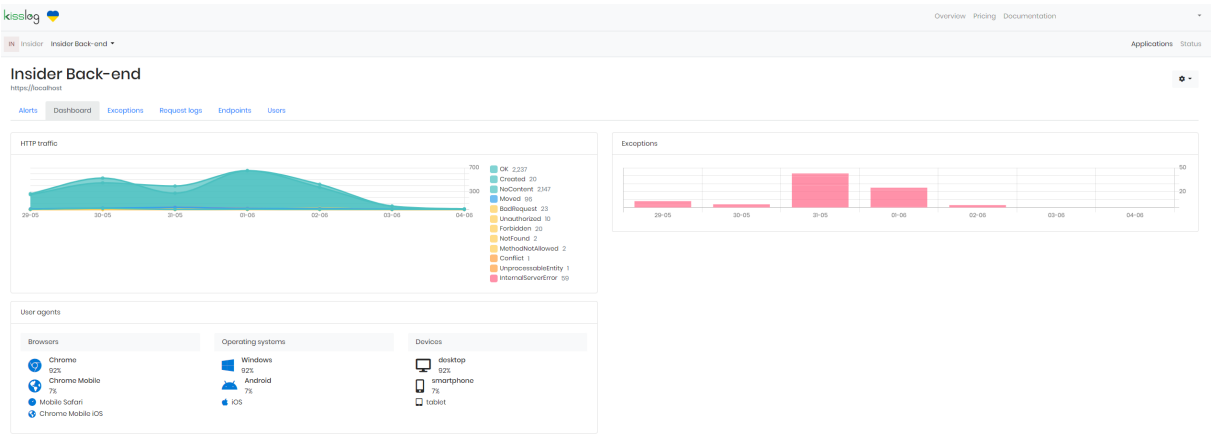


Figura 12 – *KissLog*

Fonte: Produzido pelo autor

3.3.6 Saúde da aplicação

Aplicações em .NET fornecem mecanismos integrados para verificação da saúde e integridade dos seus componentes infraestruturais. Esses aparatos verificam, por exemplo, a conexão com serviços de *e-mail* e banco de dados. A Figura 13 apresenta a interface gráfica do monitoramento da Insider.

[9] Disponível em: <https://kisslog.net/>

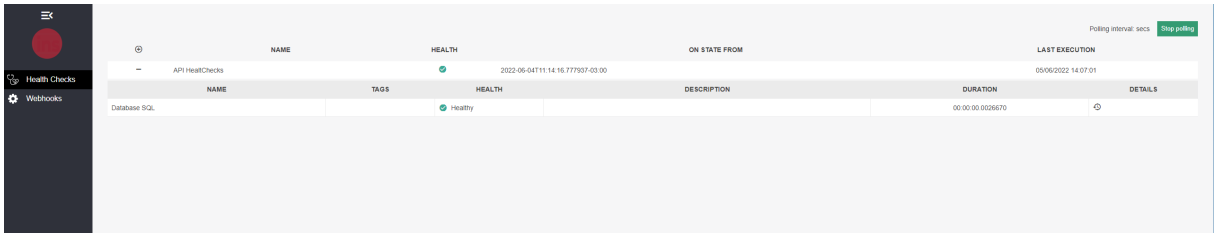


Figura 13 – Interface do monitoramento da saúde da Insider

Fonte: Produzido pelo autor

3.3.7 Swagger

O uso de APIs REST no contexto de desenvolvimento de *software* está bastante difundido em aplicações de todos os portes (vide 2.3.2). Conforme Stylos e Myers (2007), APIs são coleções de código-fonte implementadas para uso de outros desenvolvedores. Para os autores, uma API facilita a integração inter-sistemas, além de propiciar o reúso de componentes. Esta visão corrobora com o manifesto por Ofoeda, Boateng e Effah (2019), que argumentam que o uso de APIs permite a comunicação entre os *softwares* com diferentes tecnologias.

Geralmente, a comunicação com uma API ocorre mediante o acesso aos seus *endpoints*, onde objetos são enviados/recebidos por intermédio dos métodos HTTP. Depreende-se, portanto, que o contrato de uma API deve ser claro e estável para um uso eficiente por parte dos seus consumidores. Por este ângulo, dispor de uma interface gráfica que contenha todos os *endpoints* de uma aplicação incorpora muito valor ao *software*.

Concebido em 2010 como um empreendimento *open source*, o Swagger[10] permite a definição das rotas, modelos de requisição/resposta, formato de *headers* de uma requisição, padrões de autenticação, entre outros. Além disso, o Swagger pode ser usado como uma interface para testes das rotas do sistema através da *Swagger UI*[11]. A Figura 14 contém a especificação de alguns *endpoints* da API Insider.

3.3.8 Organização do projeto

A solução *back-end* foi construída seguindo os preceitos da CA. A justificativa para adoção deste modelo arquitetural está no desacoplamento das regras de negócio dos *frameworks* e ferramentas externas, assim como uma maior facilidade para a realização dos testes. A Figura 15 apresenta a estrutura de diretórios da aplicação *back-end* da Insider. Em consonância com Martin, Grenning e Brown (2018), observa-se que os casos de uso suportados pela *software* são evidenciados em sua organização, ou seja, uma arquitetura "gritante".

[10] Disponível em: <https://swagger.io/>
[11] Disponível em: <https://swagger.io/tools/swagger-ui/>

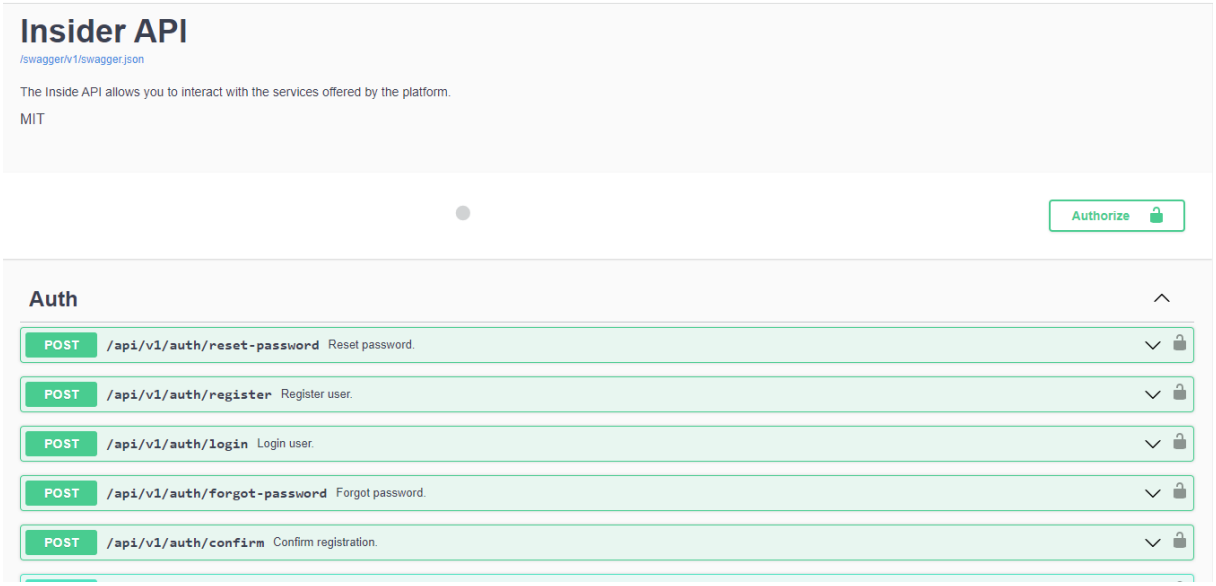


Figura 14 – Documentação do *back-end* com Swagger

Fonte: Produzido pelo autor

3.3.9 Testes

Para a garantia da segurança e qualidade durante o desenvolvimento das *features*/*refactoring*, foram implementados múltiplos testes. A abordagem para a elaboração dos testes foi o TDD, ou seja, os testes foram escritos antes mesmo das funcionalidades. Consequentemente, a escrita do código de produção segue rigorosamente as diretrizes estabelecidas pelo ***design for testability***, incrementando a testabilidade da aplicação (VALENTE, 2020).

A ferramenta para a construção dos testes automatizados foi a xUnit[12]. Foram criados, ao todo, 176 casos de teste. A Figura 16 apresenta a saída da execução dos testes realizados na *Integrated Development Environment* (IDE) Visual Studio[13].

[12] Disponível em: <https://xunit.net/>
[13] Disponível em: <https://visualstudio.microsoft.com/pt-br/vs/>

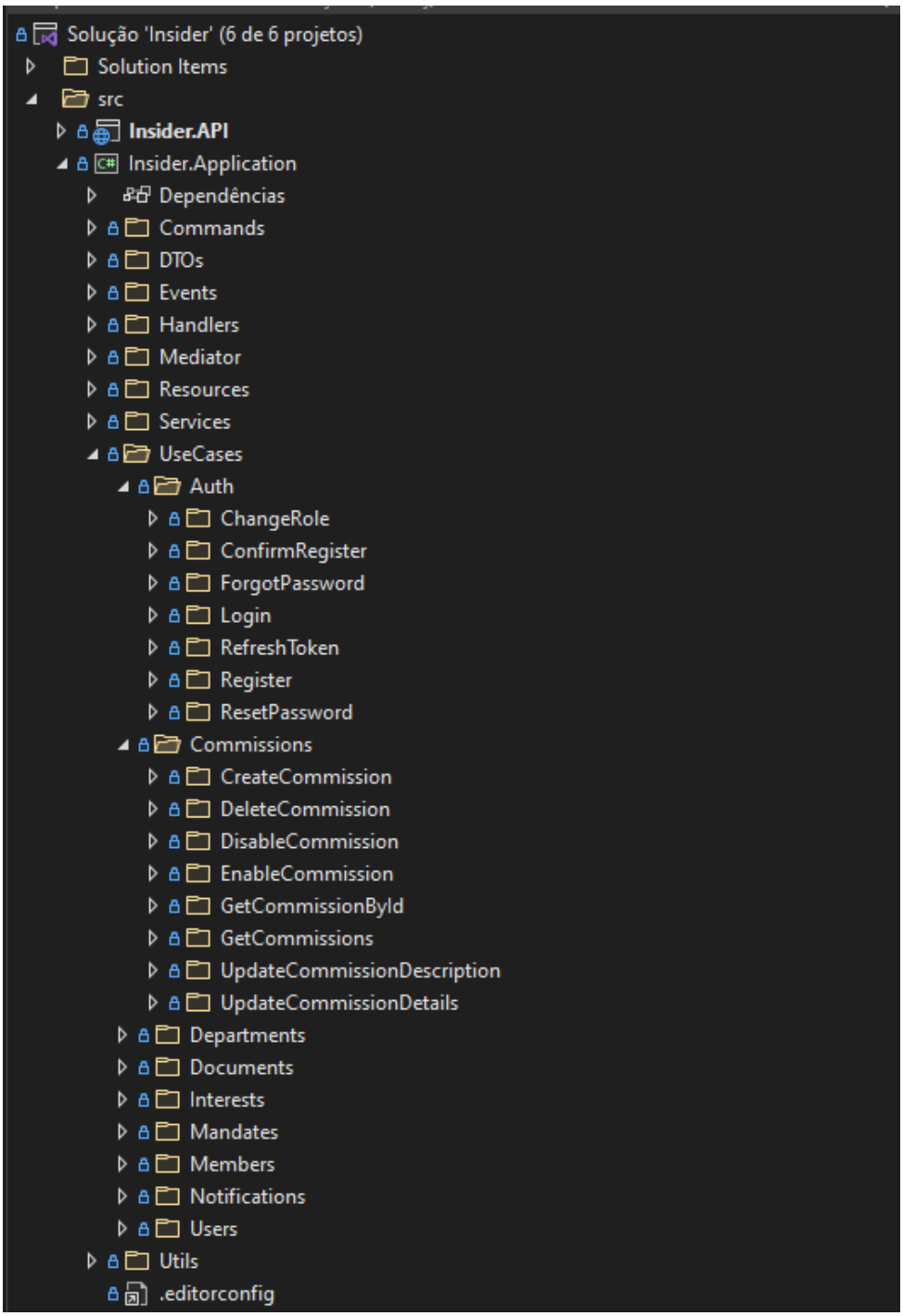


Figura 15 – Estrutura de diretórios do *back-end* da Insider

Fonte: Produzido pelo autor

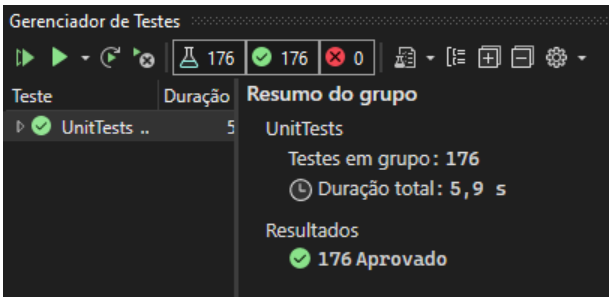


Figura 16 – Testes da solução *back-end*

Fonte: Produzido pelo autor

3.4 Banco de dados

Banco de dados e SGBDs são elementos essenciais em *softwares*. Esta tecnologia pode ser caracterizada da seguinte maneira:

> Um banco de dados tem alguma fonte da qual o dado é derivado, algum grau de interação com eventos no mundo real e um público que está ativamente interessado em seu conteúdo. (ELMASRI; NAVATHE, 2010)

O banco de dados da plataforma Insider foi elaborado com base nas entidades que modelam o processo das comissões. Exemplos de entidades são: **Comissão**, **Membro**, **Mandato**, **Departamento**. As entidades e os seus relacionamentos podem ser retratados por meio do Modelo Entidade-Relacionamento (MER). Graficamente, esse modelo é representado pelo Diagrama Entidade-Relacionamento (DER). Os diagramas simplificado e completo da plataforma Insider são exibidos na Figura 17 e Figura 18, respectivamente.

A plataforma Insider acessa o seu banco de dados com o SGBD PostgreSQL[14]. Para a atualização e versionamento do esquema do banco de dados, foram utilizadas *migrations*[15].

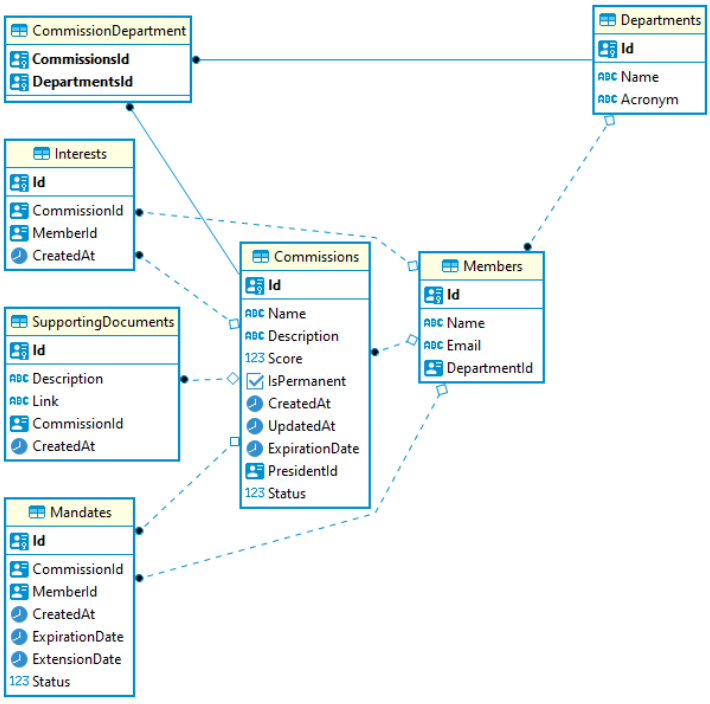


Figura 17 – Diagrama simplificado

Fonte: Produzido pelo autor

[14] Disponível em: <https://www.postgresql.org/>

[15] Disponível em: <https://docs.microsoft.com/pt-br/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli>

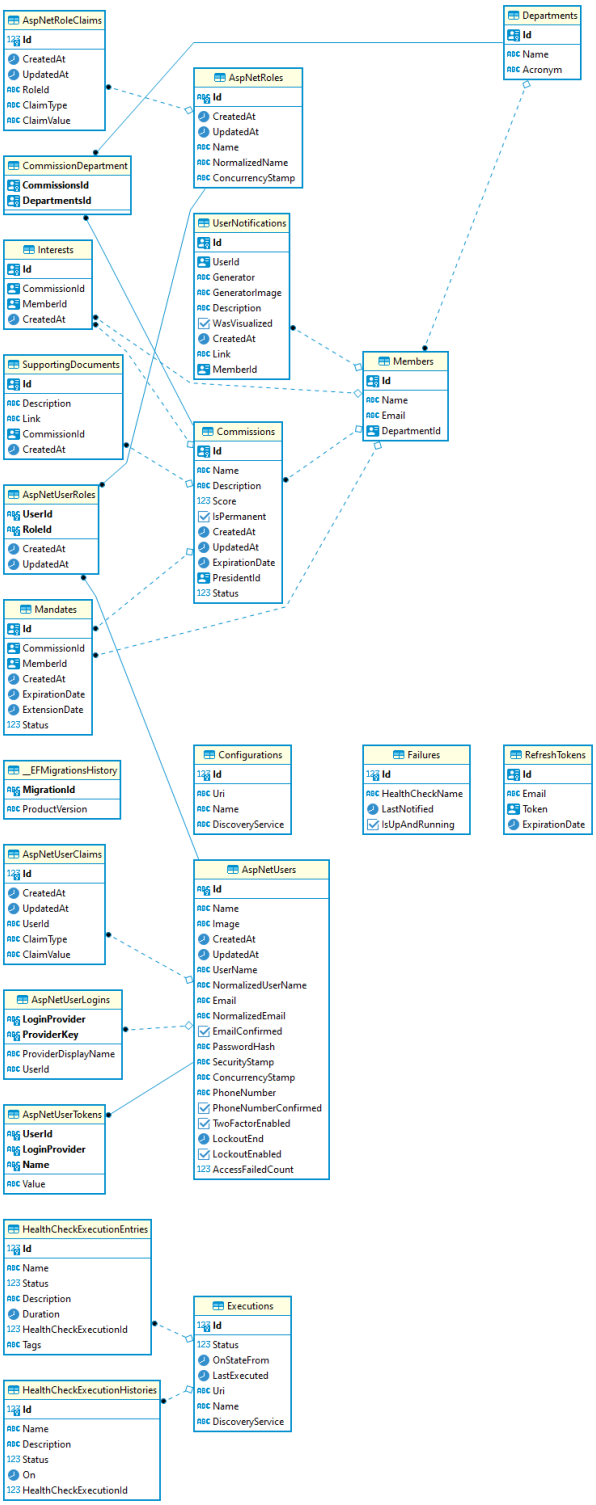


Figura 18 – Diagrama completo

Fonte: Produzido pelo autor

3.5 Plataforma *Web*

A interface com o usuário da plataforma Insider foi desenvolvida com a tecnologia ReactJS[16] em conjunto com o NextJS[17]. O ReactJS é uma biblioteca *JavaScript* elaborada por engenheiros do Facebook[18] para construção de *front-ends* complexos.

Um elemento que simplifica a criação de interfaces gráficas usando o ReactJS reside no fato dele utilizar o paradigma de programação declarativa.

> "A programação lógica (declarativa) permite a um programa modelar um problema declarando qual resultado o programa deve obter, em vez de como ele deve ser obtido.(TUCKER; NOONAN, 2009)

Neste sentido, o código-fonte contém somente as declarações sobre como os elementos devem ser apresentados de acordo com o estado da aplicação. Em adição, a possibilidade de decomposição dos elementos visuais em componentes facilita a construção das telas do sistema.

O NextJS é um *framework web* desenvolvido em ReactJS. A escolha pelo NextJS foi pautada em sua simplicidade de configuração e extensabilidade, além do seu bom desempenho em produção. Grandes empresas do mercado de trabalho utilizam o NextJS em sua esteira de tecnologias, tais como: Uber[19], Netflix[20], Twitch[21] e HBO Max[22].

3.5.1 Documentação

A documentação em projetos *front-end* é um ótimo instrumento consultivo para a equipe de desenvolvimento. Nela, os desenvolvedores podem visualizar as propriedades, variações e comportamentos dos componentes visuais do sistema.

A tecnologia utilizada para este fim na plataforma Insider foi o Storybook[23]. O Storybook é um projeto *open source* para documentação de componentes da *User Interface* (UI). A Figura 19 exibe a interface do Storybook para a aplicação desenvolvida neste trabalho.

[16] Disponível em: <https://pt-br.reactjs.org/>
[17] Disponível em: <https://nextjs.org/>
[18] Disponível em: <https://www.facebook.com/>
[19] Disponível em: <https://www.uber.com/br/pt-br/>
[20] Disponível em:<https://www.netflix.com/br/>
[21] Disponível em: <https://www.twitch.tv/>
[22] Disponível em: <https://www.hbomax.com/br/pt>
[23] Disponível em: <https://storybook.js.org/>

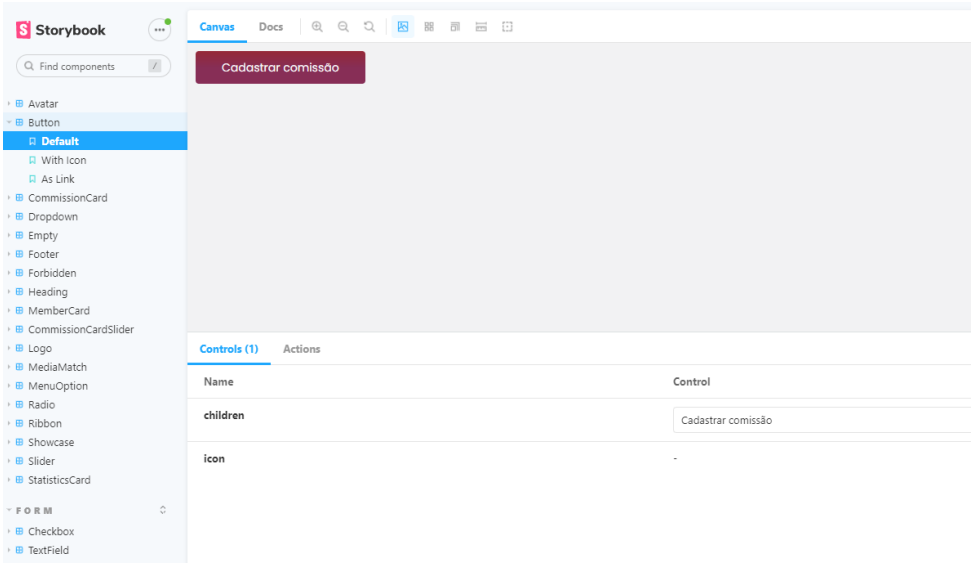


Figura 19 – Storybook

Fonte: Produzido pelo autor

3.6 Considerações finais

Neste capítulo foram discutidos os requisitos, casos de uso e principais tecnologias para o desenvolvimento do sistema. Essencialmente, foram abordadas as arquiteturas, decisões tecnológicas e as ferramentas inseridas em todo o projeto. No Capítulo 4, serão mostradas as telas da aplicação.

4 Resultados

Este capítulo contempla os resultados obtidos com o desenvolvimento do projeto. São abordadas as funcionalidades e telas do *software*.

4.1 Plataforma Insider

A presente plataforma *web* se propõe a atuar como um meio para o gerenciamento das comissões universitárias. A Insider pauta-se em preceitos de harmonia e arquitetura visual em suas interfaces, de tal modo que os usuários sintam-se confortáveis durante a experiência de uso.

4.2 Funcionalidades

Para aumentar a segurança na comunicação cliente-servidor, a sessão de cada usuário autenticado possui um *JSON Web Token* (JWT) associado. JWT é um padrão da indústria para autenticação e transferência de informações (JONES; BRADLEY; SAKIMURA, 2015). Esse *token* é gerado pelo *back-end* e atribuído à sessão do usuário após sua autenticação. Destaca-se que os acessos habilitados para o usuário estão contidos dentro do JWT. Cada solicitação feita ao servidor envia um JWT em seu *header*, de maneira que a resposta para a requisição está condicionada à validade do *token*.

A seguir, são apresentadas as telas constituintes da plataforma Insider.

4.2.1 *Login*

A Figura 20 apresenta a tela de *login* da Insider. Ela é composta por campos para entrada de *e-mail*, senha e botão para submissão da requisição. Caso o usuário esqueça a sua senha, pode-se clicar em “Esqueceu a sua senha?” para obter as informações para redefini-la. Além disso, caso o utilizador ainda não esteja cadastrado na plataforma, basta clicar em “Cadastrar” para navegar até o setor de registro do sistema. A Figura 21 captura a versão *mobile* do *login*.

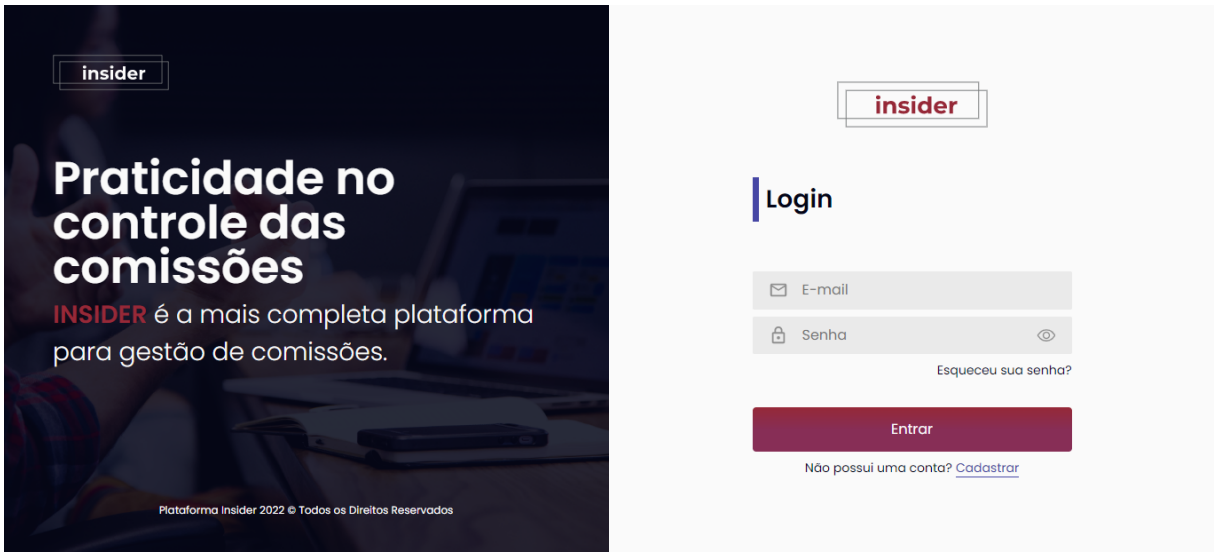


Figura 20 – Tela de *login* (*Desktop*)

Fonte: Produzido pelo autor

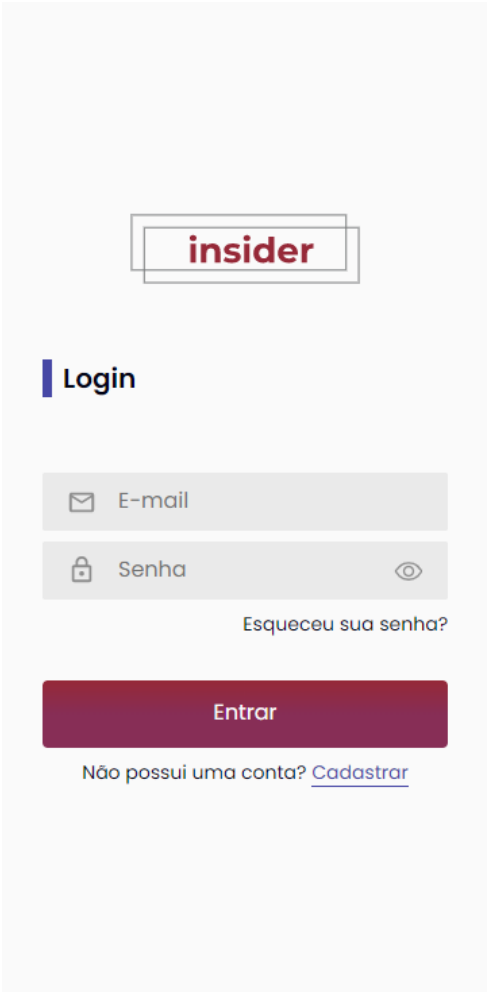


Figura 21 – Tela de *login* (*Mobile*)

Fonte: Produzido pelo autor

4.2.2 Cadastro de Usuário

As telas de cadastro *desktop* e *mobile* são mostradas nas Figuras 22 e 23, nesta ordem. A senha do usuário deve atender a certos requisitos que garantem a criação de senhas fortes, conforme pode ser visualizado na Figura 24.

Figura 22 – Tela de cadastro (*Desktop*)

Fonte: Produzido pelo autor

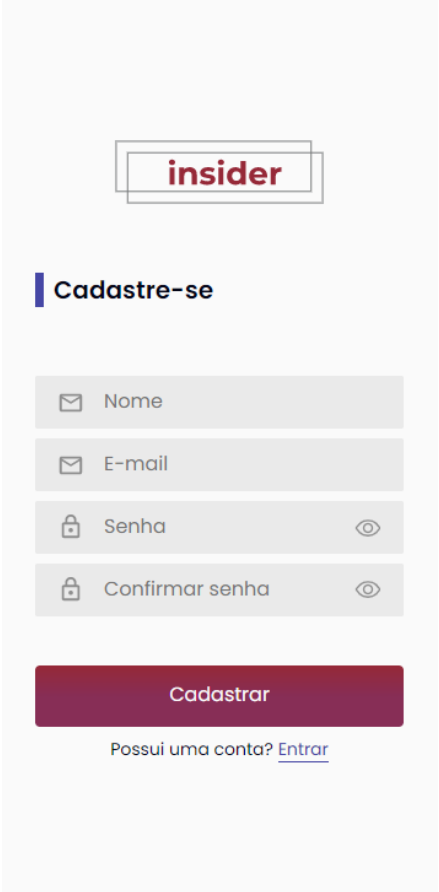


Figura 23 – Tela de cadastro (*Mobile*)

Fonte: Produzido pelo autor

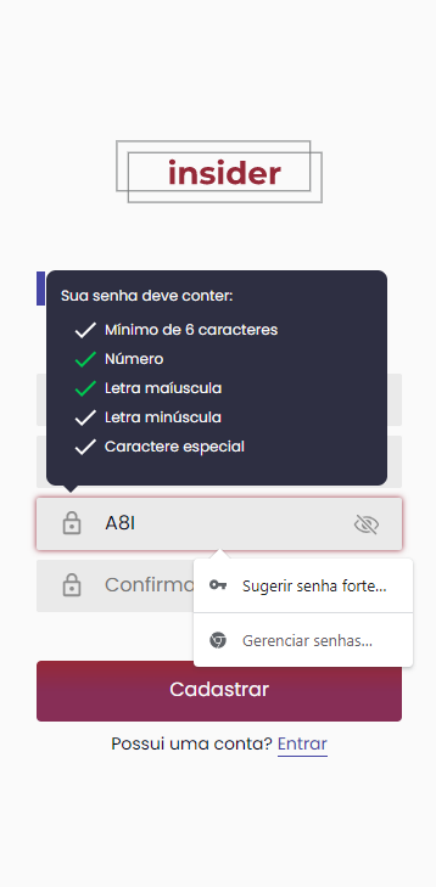


Figura 24 – Criação de senha

Fonte: Produzido pelo autor

Após o cadastro na plataforma, um *e-mail* de confirmação de registro é enviado para o usuário. O conteúdo da mensagem pode ser visto na Figura 25.

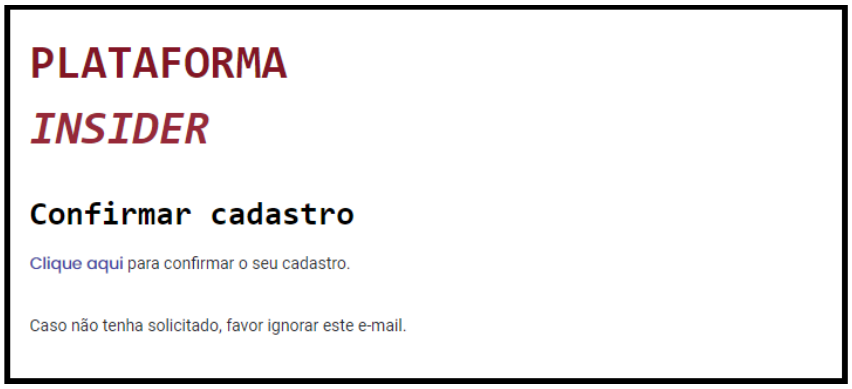


Figura 25 – *E-mail* de confirmação

Fonte: Produzido pelo autor

Ao clicar no *link* enviado no *e-mail*, o usuário é redirecionado para a tela mostrada na Figura 26. Vale-se observar que um novo usuário criado na aplicação deve estar associado a um departamento. Inicialmente, o papel de **Membro** é atribuído ao novo usuário.

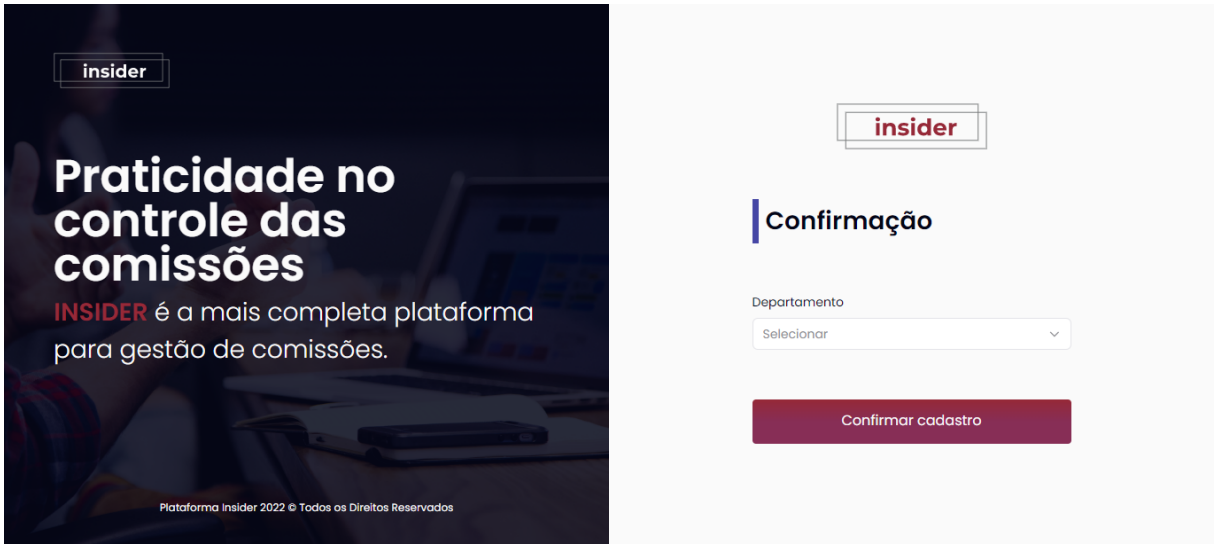


Figura 26 – Confirmação do cadastro

Fonte: Produzido pelo autor

4.2.3 Redefinição de Senha

Ao clicar em "Esqueceu a sua senha?" no *login*, o utilizador é redirecionado até a tela apresentada na Figura 27. Um *link* com um *token* de recuperação é enviado ao *e-mail* do usuário. A Figura 28 contém o *e-mail* de recuperação de senha.

Figura 27 – Tela para redefinição de senha

Fonte: Produzido pelo autor

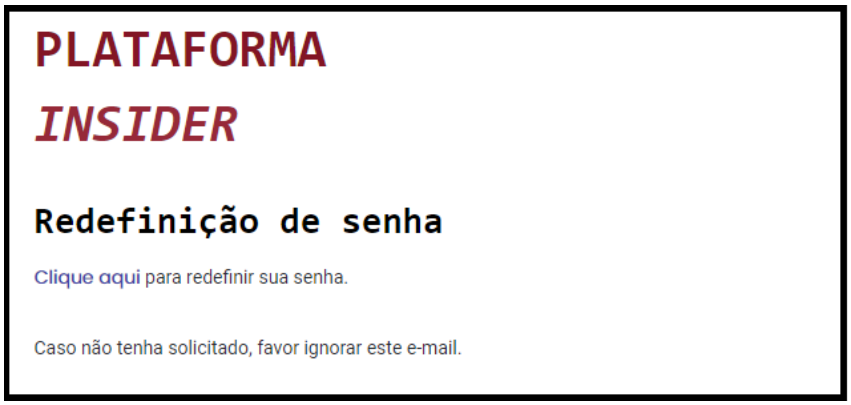


Figura 28 – *E-mail* de recuperação

Fonte: Produzido pelo autor

4.2.4 Meu Perfil

Após autenticar-se na plataforma, o usuário está habilitado para consultar/modificar suas informações pessoais - imagem, nome e senha. As Figuras 29 e 30 representam as telas do perfil em aparelhos *desktop* e *mobile*.

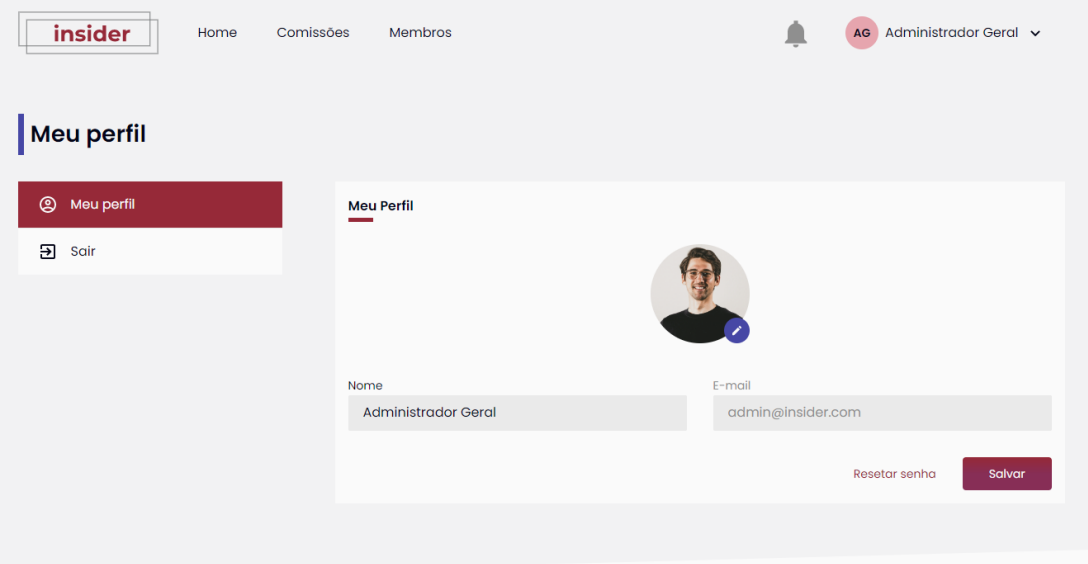


Figura 29 – Perfil (*Desktop*)

Fonte: Produzido pelo autor

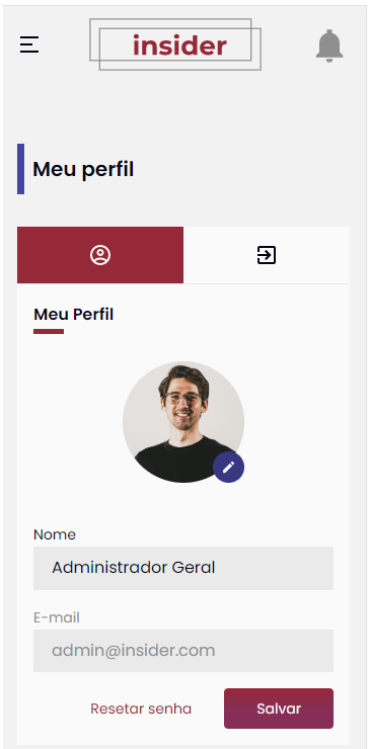


Figura 30 – Perfil (*Mobile*)

Fonte: Produzido pelo autor

4.2.5 Notificações

Eventos, como o registro de novos usuários ou cadastro de comissões, geram notificações para os utilizadores da Insider. As Figuras 31 e 32 apresentam as telas das notificações em duas resoluções diferentes.

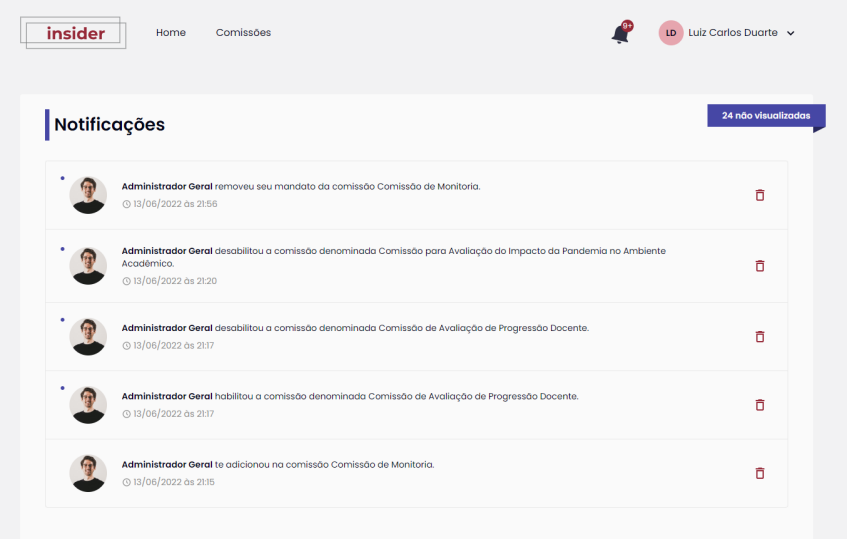


Figura 31 – Notificações (*Desktop*)

Fonte: Produzido pelo autor

Figura 32 – Notificações (*Mobile*)

Fonte: Produzido pelo autor

4.2.6 Tela Inicial

As Figuras 33 e 34 mostram a tela inicial da plataforma vista em um *desktop* e um dispositivo móvel, respectivamente. Vale explicar que o telefone móvel presente na versão *desktop* da página inicial da Insider tenta acentuar a ideia de que a aplicação é responsiva e pode ser acessada de um celular.

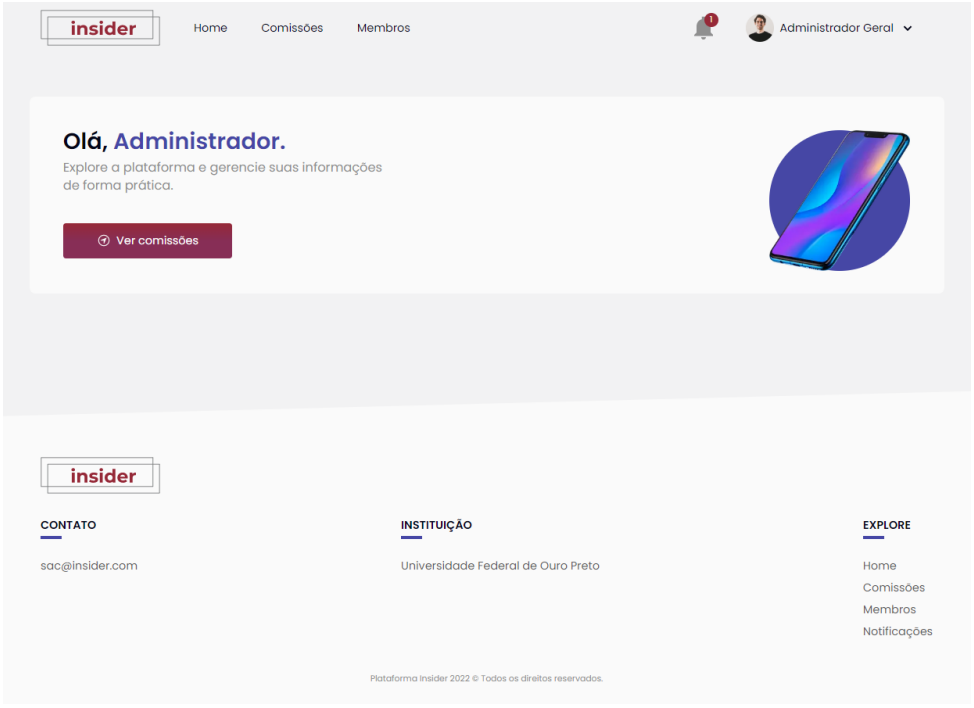


Figura 33 – Tela inicial (*Desktop*)

Fonte: Produzido pelo autor

Figura 34 – Tela inicial (*Mobile*)

Fonte: Produzido pelo autor

4.2.7 Menu

As Figuras 35 e 36 mostram as opções de menu em diferentes resoluções para o Administrador Geral e demais usuários. Já as Figuras 37 e 38 exibem esses itens da aplicação em dispositivos móveis.

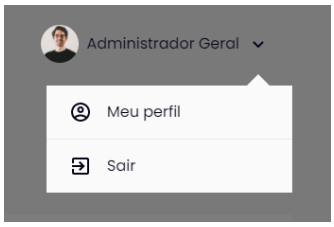


Figura 35 – Menu Administrador (*Desktop*)

Fonte: Produzido pelo autor

Figura 36 – Menu (*Desktop*)

Fonte: Produzido pelo autor

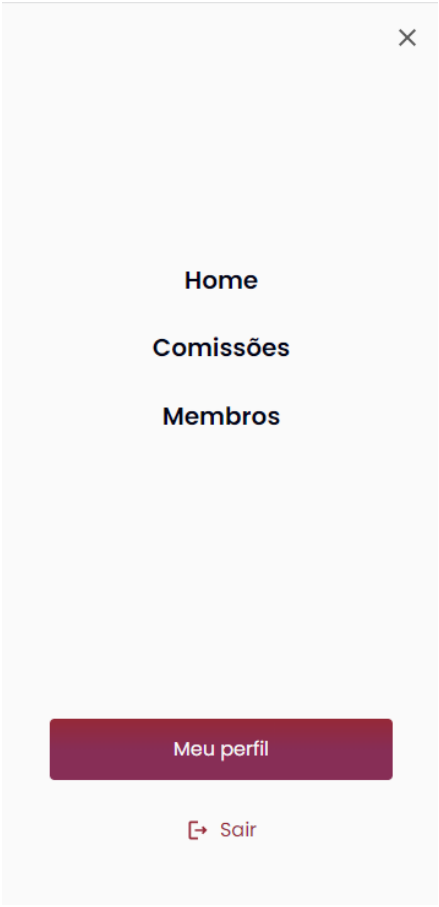


Figura 37 – Menu Administrador (*Mobile*)

Fonte: Produzido pelo autor

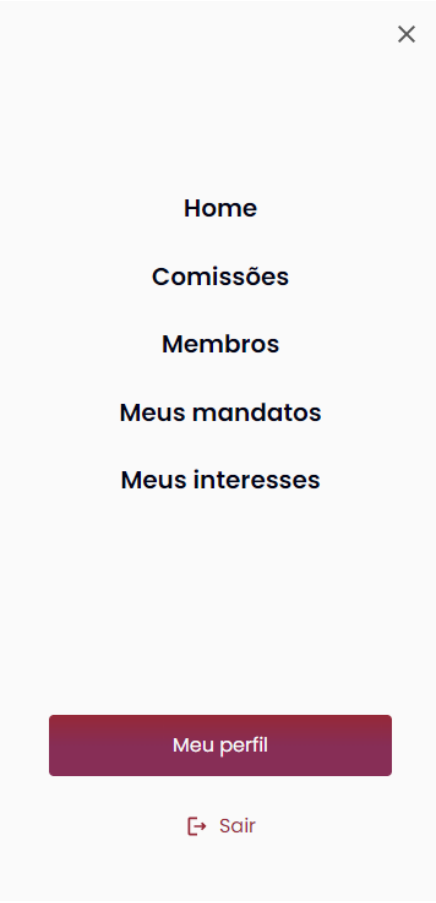


Figura 38 – Menu (*Mobile*)

Fonte: Produzido pelo autor

4.2.8 Cadastro de comissão

O processo de cadastro de uma comissão na plataforma Insider é divido em 4 passos: *informações gerais*, *escolha dos membros*, *detalhes específicos* e *revisão*. Cada uma das etapas inclui uma série de validações, para que não seja possível a criação de uma comissão em um estado inválido.

4.2.8.1 Informações Gerais

A Figura 39 apresenta o *layout* da primeira etapa do cadastro de comissão em um computador. Já a Figura 40 exibe a versão para um dispositivo móvel. A Figura 41 apresenta os validadores pertinentes às informações gerais de uma comissão, isto é, o seu nome, descrição, pontuação e departamentos.

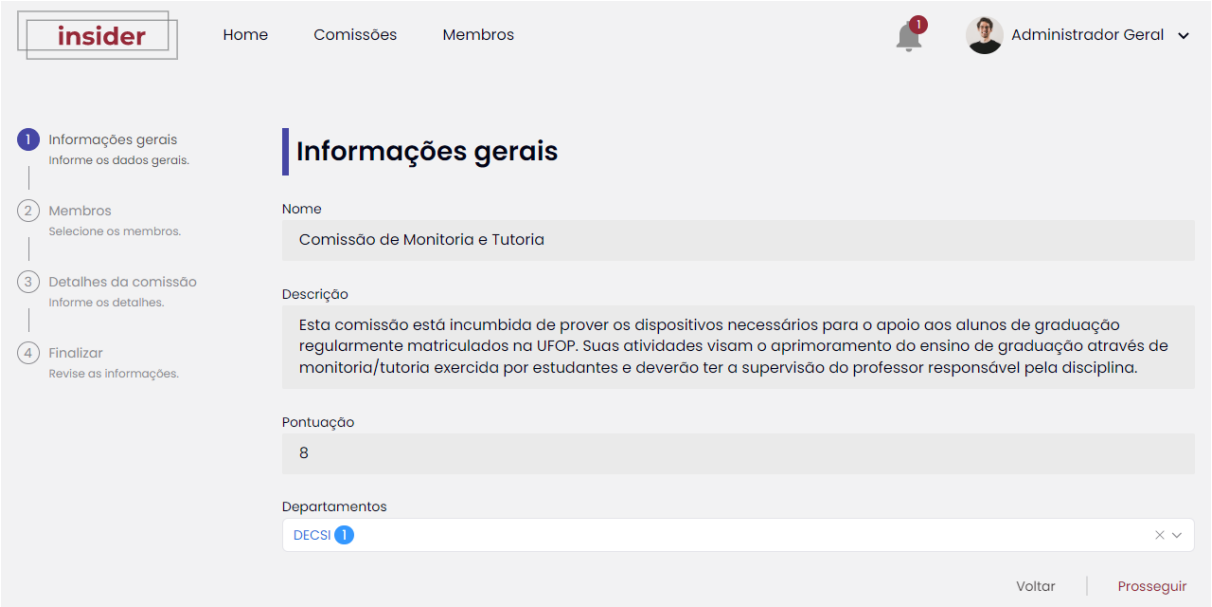


Figura 39 – Informações gerais para o cadastro de comissão (*Desktop*)

Fonte: Produzido pelo autor

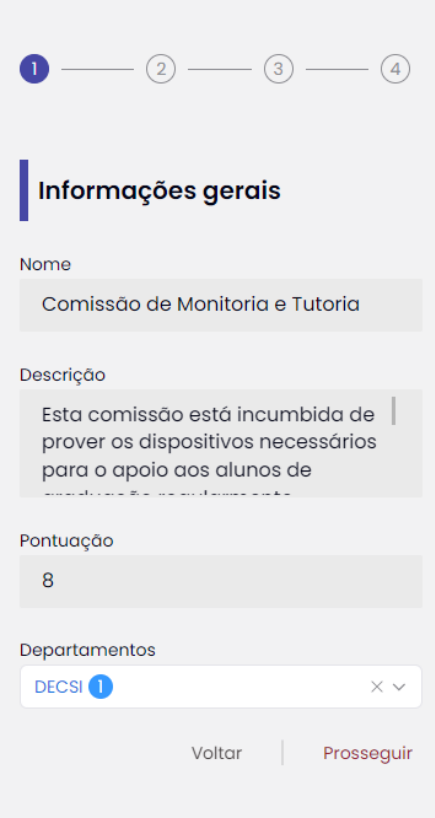


Figura 40 – Informações gerais para o cadastro de comissão (*Mobile*)

Fonte: Produzido pelo autor

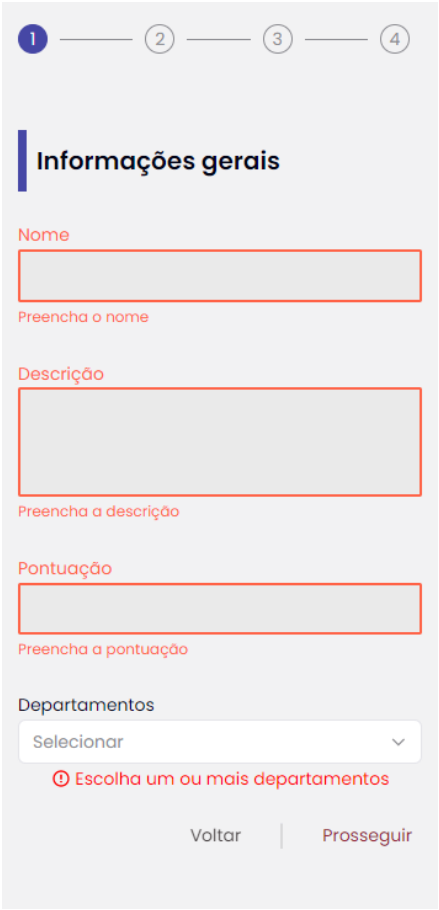


Figura 41 – Validadores das informações gerais

Fonte: Produzido pelo autor

4.2.8.2 Membros

As Figuras 42 e 43 apresentam as capturas de tela para a etapa de escolha dos membros. A Figura 44 evidencia a necessidade de escolha de, pelo menos, um membro para a comissão.

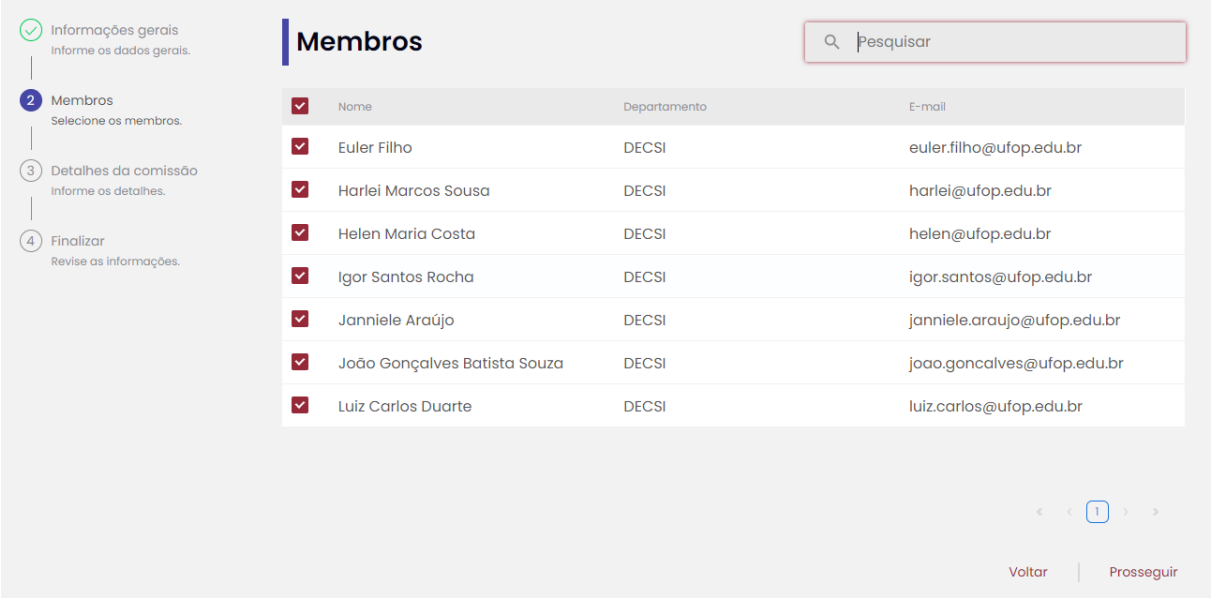


Figura 42 – Escolha dos membros para o cadastro de comissão (*Desktop*)

Fonte: Produzido pelo autor

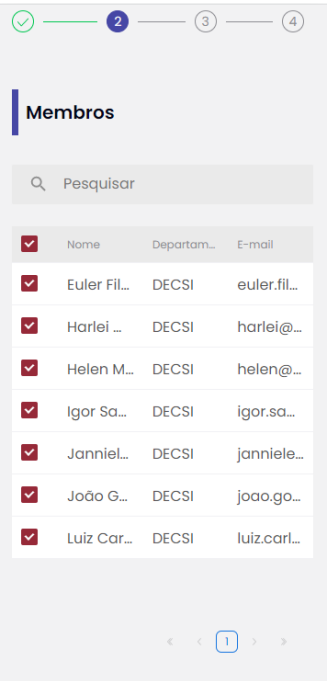


Figura 43 – Escolha dos para o cadastro de comissão (*Mobile*)

Fonte: Produzido pelo autor

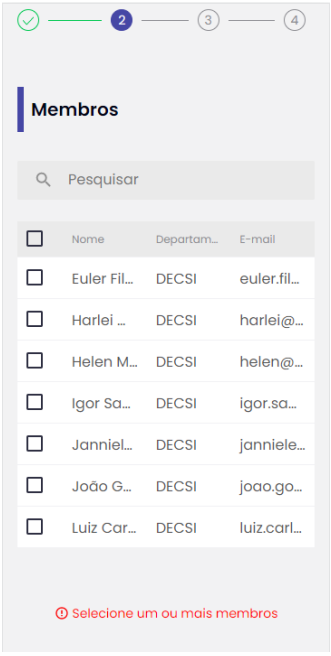


Figura 44 – Validadores na escolha dos membros

Fonte: Produzido pelo autor

4.2.8.3 Detalhes

As Figuras 45 e 46 mostram as telas para a etapa de definição dos detalhes da comissão. Os detalhes da comissão envolvem a escolha do seu presidente, documento comprobatório da criação, datas de início e finalização (caso seja uma comissão temporária). A Figura 47 apresenta exemplos de erros de validação para o formulário.

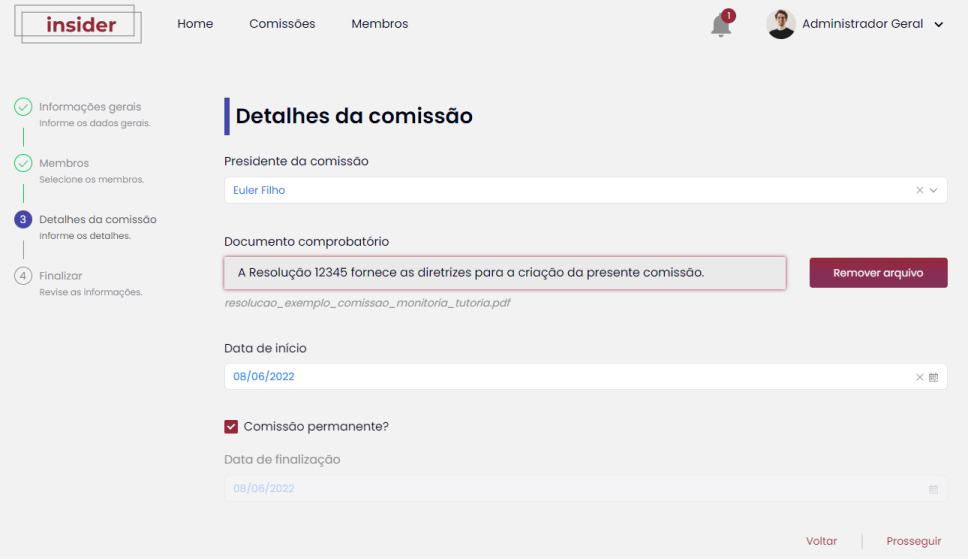


Figura 45 – Detalhes do cadastro de comissão (*Desktop*)

Fonte: Produzido pelo autor

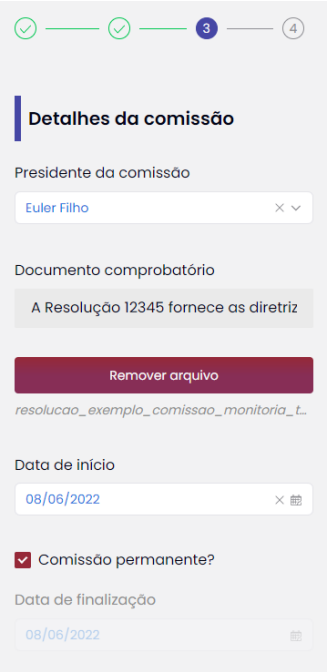


Figura 46 – Detalhes do cadastro de comissão (*Mobile*)

Fonte: Produzido pelo autor

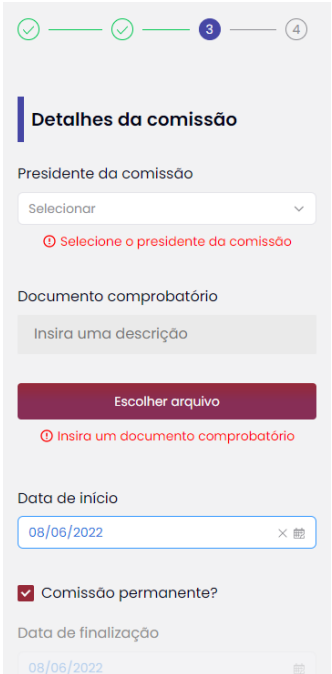


Figura 47 – Validadores na escolha dos membros

Fonte: Produzido pelo autor

4.2.8.4 Revisão

As Figuras 48 e 49 mostram a última etapa do cadastro da comissão. O fim do mandato de cada um dos membros pode ser definido neste momento.

Ao clicar no botão "Finalizar", a comissão é cadastrada no sistema. A tela de sucesso do cadastramento é exibida na Figura 50.

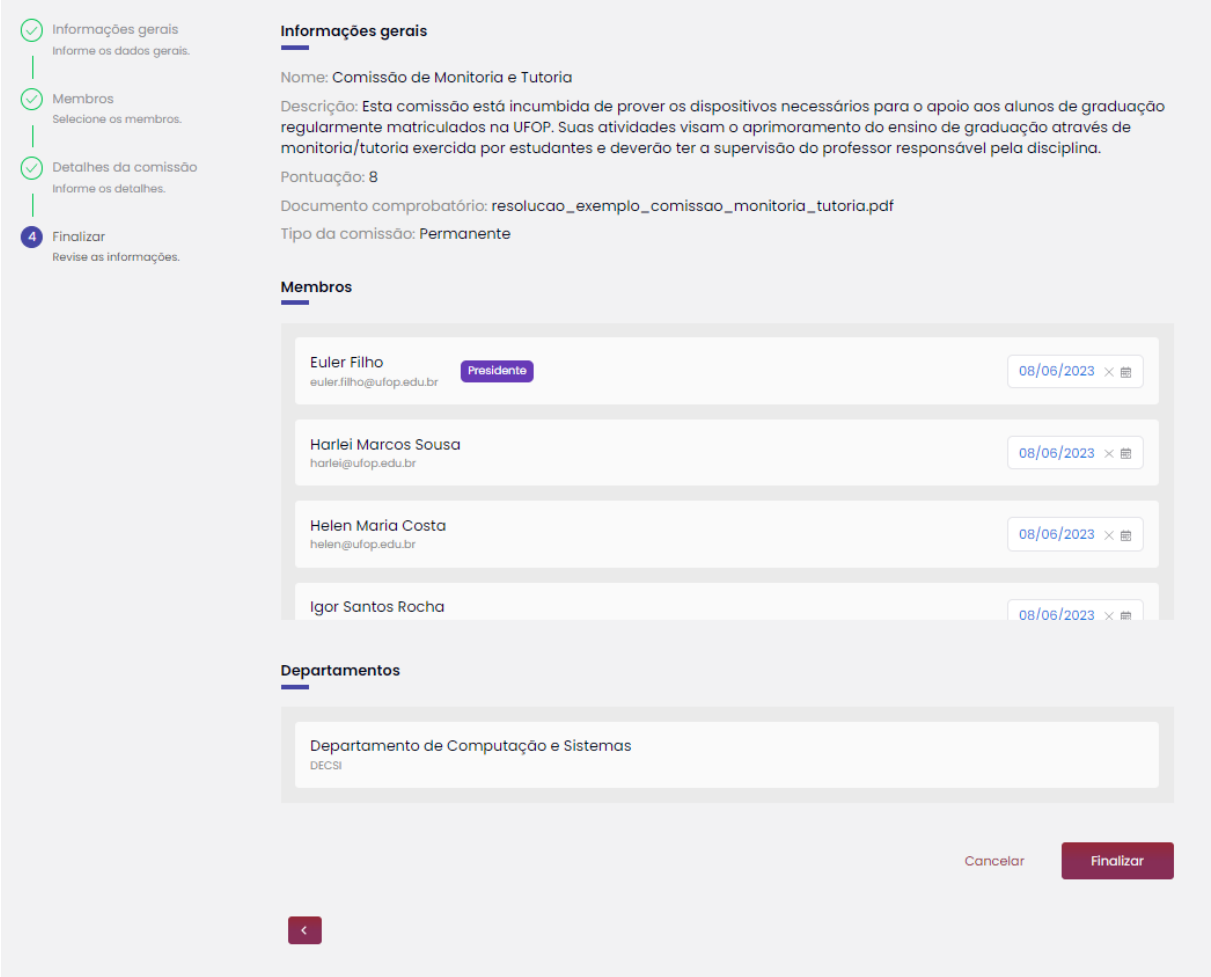


Figura 48 – Revisão do cadastro de comissão (*Desktop*)

Fonte: Produzido pelo autor

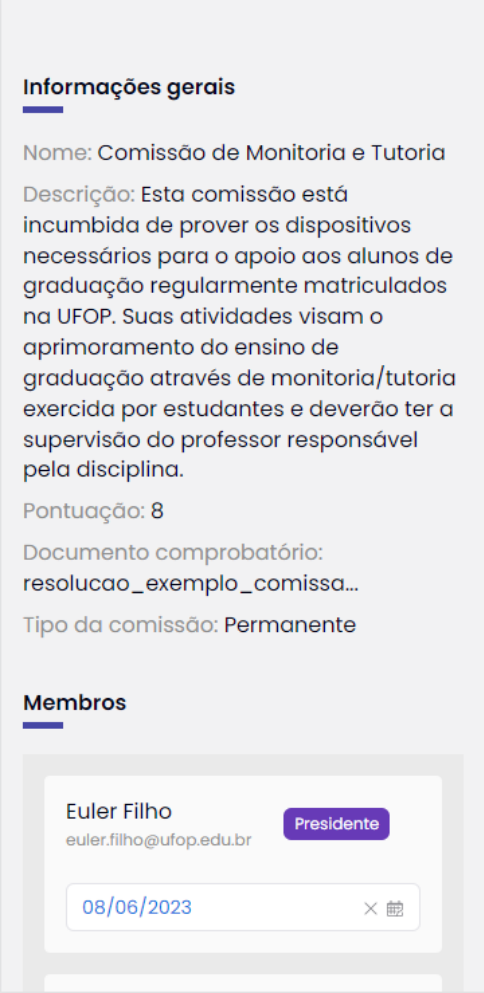


Figura 49 – Revisão do cadastro de comissão (*Mobile*)

Fonte: Produzido pelo autor

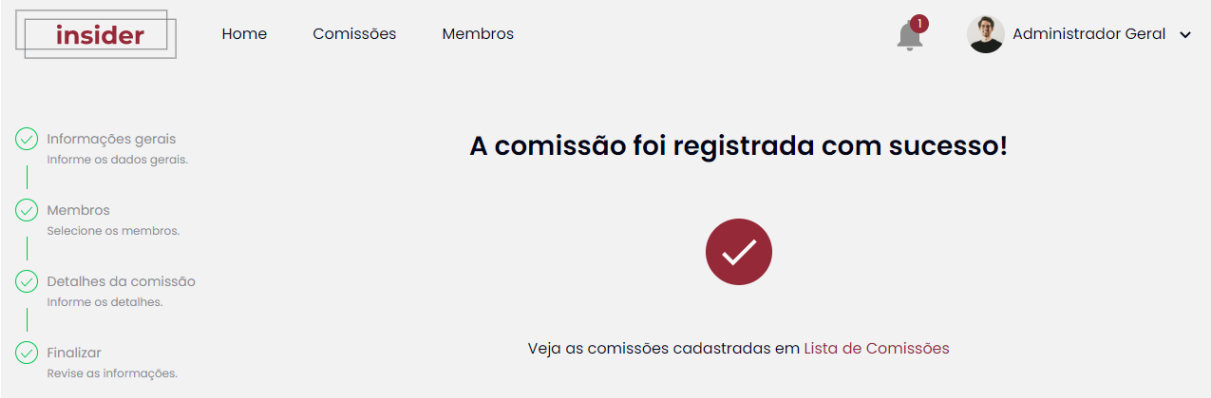


Figura 50 – Sucesso no cadastro da comissão

Fonte: Produzido pelo autor

4.2.9 Listagem das comissões

A Figura 51 exibe a tela da listagem das comissões. A busca das comissões pode ser otimizada com a utilização de múltiplos filtros. A Figura 52 apresenta um exemplo de pesquisa com a aplicação do filtro de nomes. Por último, a Figura 53 apresenta a listagem em um ambiente *mobile*. Observa-se que os botões para adicionar e exportar para *excel* estão disponíveis somente para os administradores e secretários.

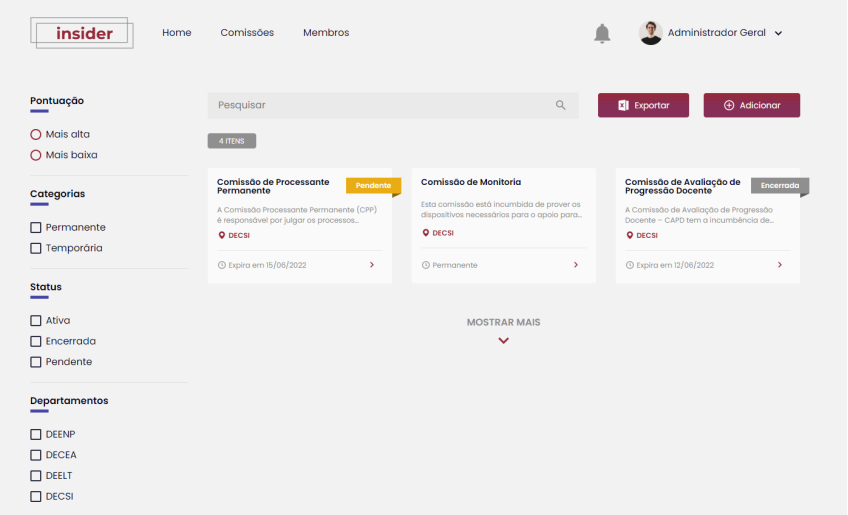


Figura 51 – Listagem das comissões (*Desktop*)

Fonte: Produzido pelo autor

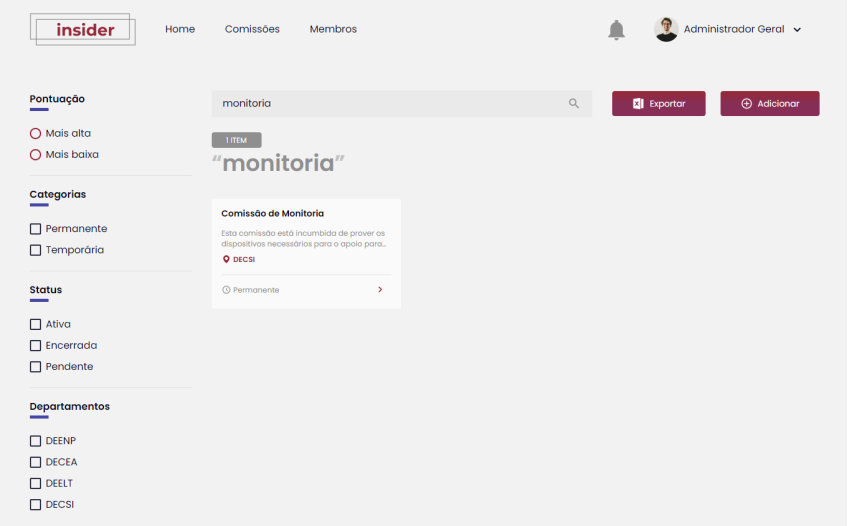


Figura 52 – Exemplo de filtragem na listagem das comissões

Fonte: Produzido pelo autor

Figura 53 – Listagem das comissões (*Mobile*)

Fonte: Produzido pelo autor

4.2.10 Comissão

As Figuras 54 e 55 revelam a composição da tela de uma comissão específica. Mediante as Figuras, averigua-se que as comissões também podem ser editadas neste espaço. A título de exemplo, a Figura 56 apresenta o componente de modificação dos detalhes da comissão, onde é possível alterar o presidente da comissão, sua pontuação e categoria - permanente ou temporária.

Todas as opções de remoção precisam ser confirmadas pelo usuário, como exposto na Figura 57, prevenindo eventuais erros.

Somente usuários com acessos para edição conseguem ver a lista de interessados e modificar o estado de uma comissão. Neste sentido, vale-se ressaltar que os **membros** tem acesso somente de leitura nesta parte do sistema, de forma que os botões de modificação e deleção não são renderizados para esses atores.

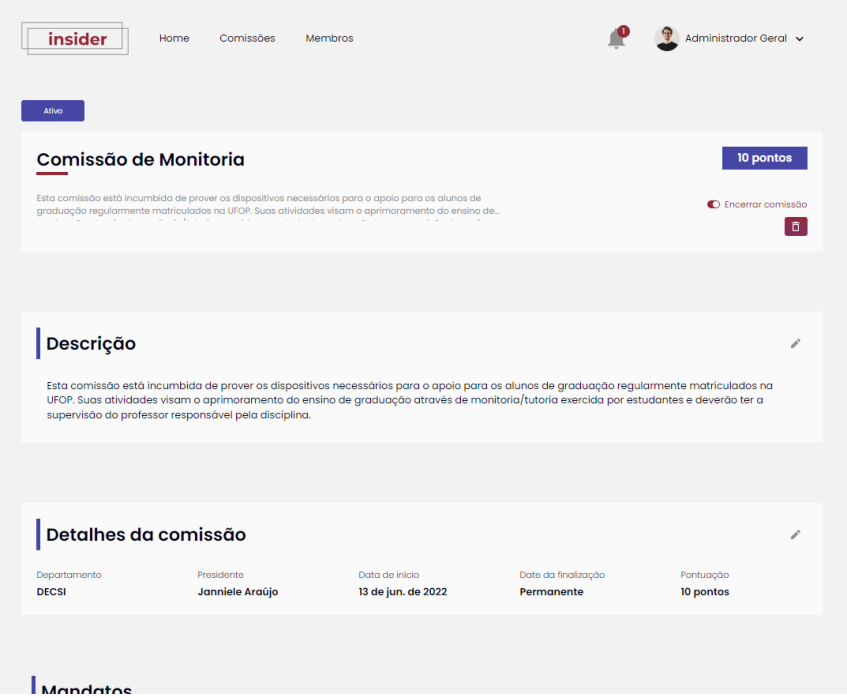


Figura 54 – Comissão - Parte 1

Fonte: Produzido pelo autor

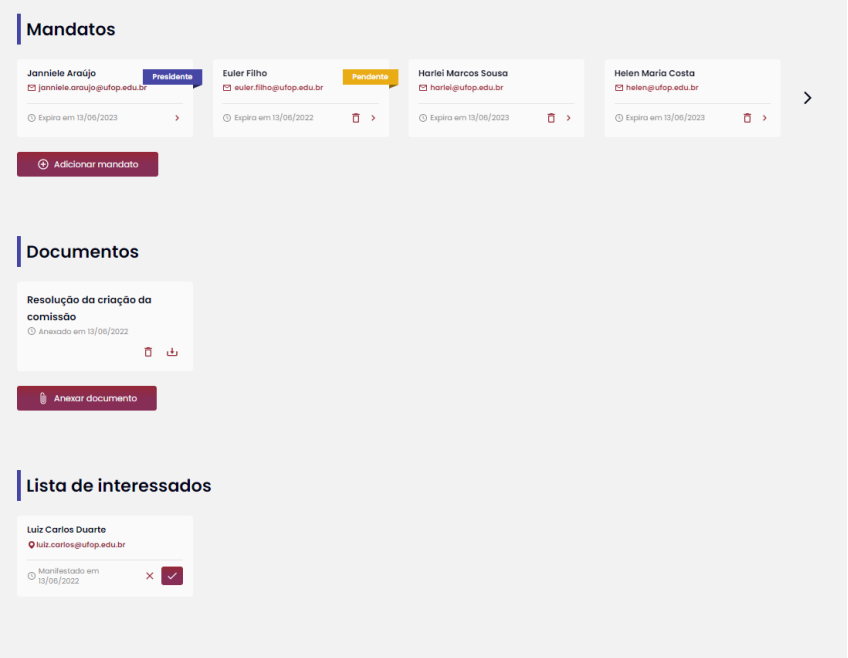


Figura 55 – Comissão - Parte 2

Fonte: Produzido pelo autor

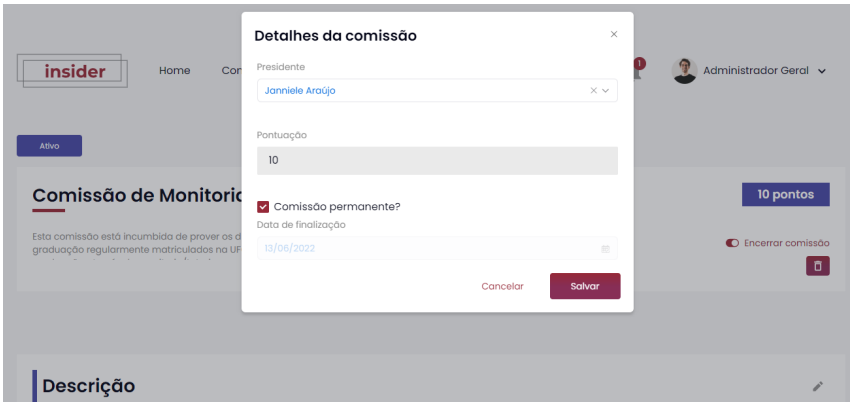


Figura 56 – Edição de detalhes

Fonte: Produzido pelo autor

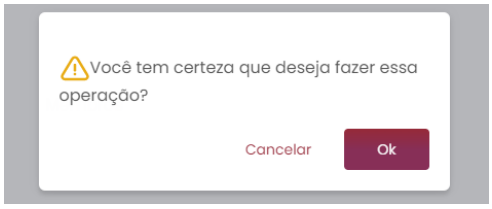


Figura 57 – Confirmação de deleção

Fonte: Produzido pelo autor

4.2.11 Listagem dos Membros

A tela de listagem dos membros é mostrada na Figura 58. Estão disponíveis os filtros de departamento e nome.

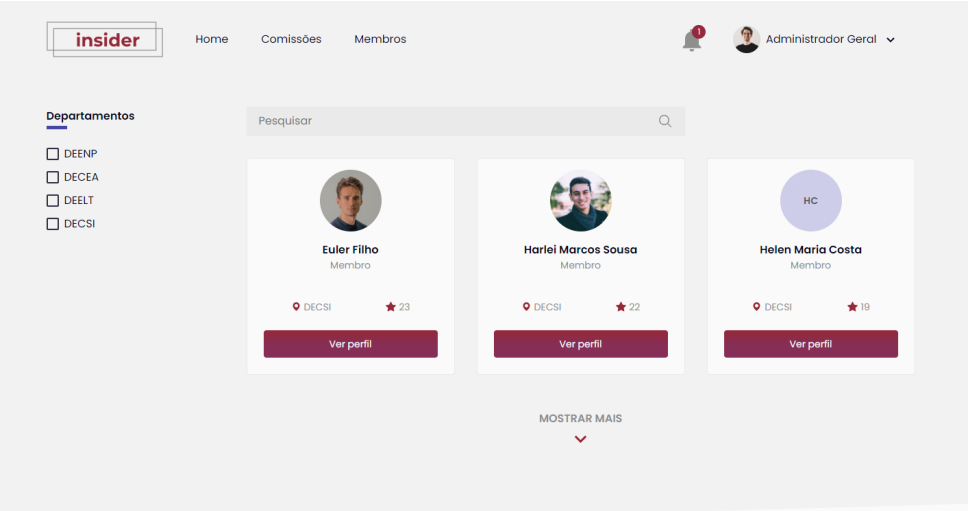


Figura 58 – Listagem dos membros

Fonte: Produzido pelo autor

4.2.12 Perfil do Usuário

A tela 59 exibe a tela de perfil do usuário. O controle de acessos também é feito nesta tela, obedecendo seguintes regras:

1. Administradores: os administradores são capazes de alterar o acesso de cada usuário livremente;
2. Secretários: os secretários são capazes de alterar o acesso de cada usuário pertencente ao seu departamento somente para os papéis de membro ou secretário;
3. Membros: os membros não estão habilitados para o controle de acessos.

O componente de controle de acessos pode ser visto na Figura 60.

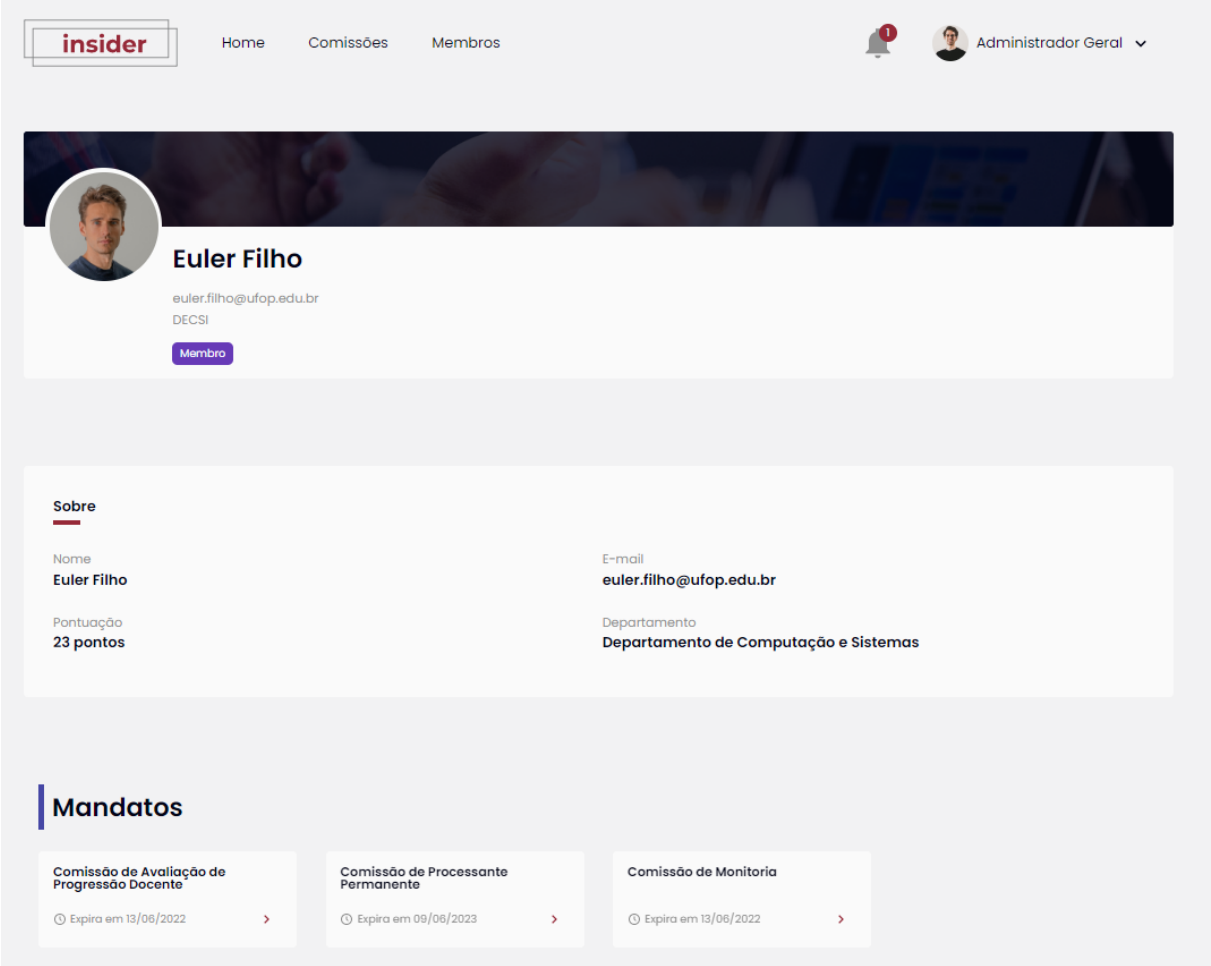


Figura 59 – Perfil do usuário

Fonte: Produzido pelo autor

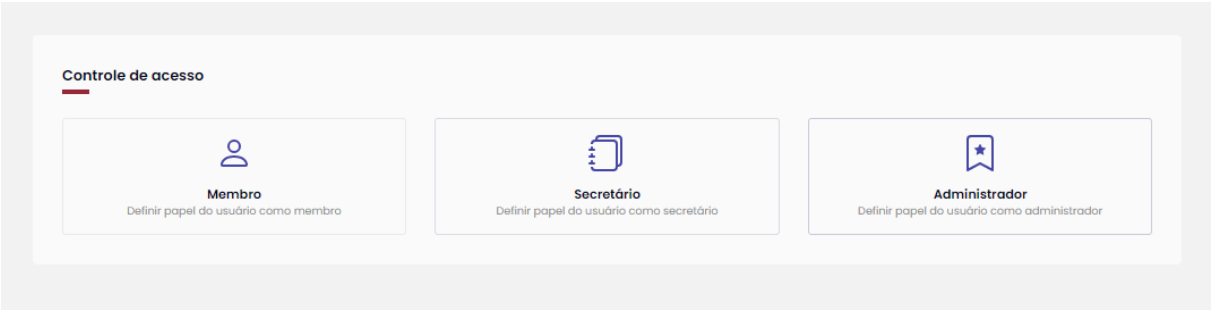


Figura 60 – Controle de acessos

Fonte: Produzido pelo autor

4.2.13 Comprovante de participação

Ao navegar para os seus mandatos - Figura 61 - e clicar no ícone da impressora, o utilizador obtém o certificado de participação em uma comissão. A Figura 62 exemplifica este documento.

Figura 61 – Mandatos

Fonte: Produzido pelo autor

Figura 62 – Certificado

Fonte: Produzido pelo autor

4.3 Considerações finais

As imagens expostas neste capítulo exibem os resultados da implementação da Insider. No capítulo seguinte, será apresentada a conclusão deste trabalho, bem como os trabalhos futuros.

5 Conclusão

Este trabalho emergiu da dificuldade do gerenciamento das comissões do DECSI por meio de planilhas eletrônicas. A despeito da comum utilização desse modelo em outros departamentos da UFOP, as planilhas eletrônicas não oferecem níveis satisfatórios de usabilidade e segurança necessárias para os seus usuários. Neste contexto, a partir do levantamento pormenorizado dos requisitos, foi possível construir uma plataforma *web* para esta administração.

O *software* desenvolvido mostra-se como uma potencial ferramenta para sanar as problemáticas atuais. A redução da complexidade no cadastro e atualização das comissões, assim como uma maior transparência no acompanhamento de suas informações, são fatores impulsionadores para a adesão do projeto.

Espera-se que a solução formulada neste trabalho possa contribuir para o aumento na qualidade da gestão das comissões. Além disso, almeja-se que, com as devidas adequações, o emprego da plataforma seja expandido para toda a universidade.

5.1 Trabalhos Futuros

Ao longo das etapas de desenvolvimento, foram mapeadas novas ideias para a continuação do projeto da plataforma Insider. São elas:

- Integração com os sistemas autenticadores da UFOP;
- Envio de notificações por *e-mail*;
- Esquema de validação dos documentos gerados pela plataforma via *Quick Response (QR) Code*;
- Possibilitar que o membro de uma comissão solicite a renovação do seu mandato;
- Módulo para o gerenciamento da pontuação administrativa dos docentes;
- Aplicação de técnicas de Ciência de Dados para extração de informações relevantes;
- Vinculação ao MinhaUfop[1].

[1] Disponível em: <https://www.minhaufop.ufop.br/>

Referências

ANTUNES, M. *API Restful: conceito, princípios e como criar*. [S.l.], 2019. Disponível em: <https://www.hostgator.com.br/blog/api-restful/>. Acesso em: 17 abr. 2022. Citado na página 26.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. Addison-Wesley, 2003. (SEI series in software engineering). ISBN 9780321154958. Disponível em: <https://books.google.fi/books?id=mdiIu8Kk1WMC>. Citado 2 vezes nas páginas 23 e 24.

BECK, K.; ANDRES, C. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Boston: Addison-Wesley Professional, 2004. ISBN 0321278658. Disponível em: <http://portal.acm.org/citation.cfm?id=1076267>. Citado na página 33.

BULEY, L. *The User Experience Team of One: A Research and Design Survival Guide*. Rosenfeld Media, 2013. ISBN 9781933820187. Disponível em: <https://books.google.com.br/books?id=vQ7cnAEACAAJ>. Citado na página 29.

CLEMENTE, R. G. *Uma arquitetura para processamento de eventos de log em tempo real*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2008. Citado na página 42.

DAVENPORT, T. *Ecologia da Informação*. [S.l.]: Editora Futura, 2000. ISBN 9788586082726. Citado na página 21.

ELMASRI, R.; NAVATHE, S. *Sistemas de Bancos de Dados-Fundamentos e Aplicações, 5 edição*. [S.l.]: LTC, 2010. Citado na página 47.

FEATHERS, M. *Working Effectively with Legacy Code*. Prentice Hall PTR, 2004. (Martin, Robert C). ISBN 9780131177055. Disponível em: <https://books.google.com.br/books?id=vlo_nWophSYC>. Citado na página 31.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Tese (Publication) — University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado 2 vezes nas páginas 25 e 26.

IRAWAN, S.; FOSTER, S.; TANNER, K. The mandated adoption and implementation of an academic information system: Empirical evidence from an indonesian university. *Australasian Journal of Information Systems*, v. 22, 12 2018. Citado na página 21.

JONES, M.; BRADLEY, J.; SAKIMURA, N. *JSON Web Token (JWT)*. RFC Editor, 2015. RFC 7519. (Request for Comments, 7519). Disponível em: <https://www.rfc-editor.org/info/rfc7519>. Citado na página 51.

LAUDON, K.; LAUDON, J. *Sistemas de informação gerenciais*. Pearson Prentice Hall, 2011. ISBN 9788576059233. Disponível em: <https://books.google.com.br/books?id=e4ARYAAACAAJ>. Citado na página 21.

LEMOS, O. Arquitetura limpa na prática. 2022. Citado 2 vezes nas páginas 26 e 29.

MARTIN, R. C.; GRENNING, J.; BROWN, S. *Clean architecture: a craftsman's guide to software structure and design*. [S.l.]: Prentice Hall, 2018. Citado 3 vezes nas páginas 27, 28 e 43.

MORVILLE, P. *User Experience Design*. [S.l.], 2004. Disponível em: <http://semanticstudios.com/user_experience_design/>. Acesso em: 05 abr. 2022. Citado na página 29.

MÜLLER, E. J. *Argos: Solução centralizada para logging de aplicações*. [S.l.], 2013. Disponível em: <https://repositorio.ufsm.br/bitstream/handle/1/156/Muller_Ezequiel_Juliano.pdf?sequence=1>. Acesso em: 14 mai. 2022. Citado na página 42.

NIELSEN, J. *Usability engineering*. San Francisco, Calif.: Morgan Kaufmann Publishers, 1994. ISBN 0125184069 9780125184069. Disponível em: <http://www.amazon.de/gp/product/0125184069/ref=oh_details_o02_s00_i00>. Citado na página 35.

OFOEDA, J.; BOATENG, R.; EFFAH, J. Application programming interface (api) research: A review of the past to inform the future. *International Journal of Enterprise Information Systems (IJEIS)*, IGI Global, v. 15, n. 3, p. 76–95, 2019. Citado na página 43.

PINA, D. de J. *Dívida Técnica: Identificando, Medindo e Monitorando*. [S.l.], 2013. Disponível em: <https://bcc.ime.usp.br/tccs/2013/diogo/files/monografia.pdf>. Acesso em: 14 mai. 2022. Citado na página 39.

PINEDA, D. J. R.; MEDINA, M. C. S. *Desarrollo de un sistema de gestión académica para las unidades educativas particulares de la ciudad de Loja*. [S.l.], 2014. Disponível em: <https://dspace.unl.edu.ec/jspui/handle/123456789/13985>. Acesso em: 12 abr. 2022. Citado 2 vezes nas páginas 21 e 22.

PRADO, E.; SOUZA, C. de. *Fundamentos de Sistemas de Informação*. Elsevier Brasil, 2014. ISBN 9788535274363. Disponível em: <https://books.google.com.br/books?id=JJaoBQAAQBAJ>. Citado na página 17.

PRESSMAN, R.; MAXIM, B. *Engenharia de Software - 8ª Edição*. [s.n.], 2016. ISBN 9788580555349. Disponível em: <https://books.google.com.br/books?id=wexzCwAAQBAJ>. Citado 4 vezes nas páginas 29, 31, 33 e 38.

RICHARDS, M.; FORD, N. *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, Incorporated, 2020. ISBN 9781492043454. Disponível em: <https://books.google.com.br/books?id=_pNdwgEACAAJ>. Citado na página 23.

ROCHA NETO, A.; LIMA, G. Turma virtual do sigaa como ferramenta de apoio ao ensino. UFRN, 2009. Citado na página 21.

SOMMERVILLE, I. *Engenharia de software*. Pearson Prentice Hall, 2011. ISBN 9788579361081. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>. Citado 2 vezes nas páginas 33 e 36.

SOUZA, M.; MONTEIRO, A. *O uso da tecnologia da informação e comunicação na educação superior: o caso do portal docente do sistema de gestão acadêmica da UFC*. [S.l.], 2015. Disponível em: <http://www.repositorio.ufc.br/handle/riufc/36709>. Acesso em: 20 mar. 2022. Citado na página 17.

STYLOS, J.; MYERS, B. Mapping the space of api design decisions. In: IEEE. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)*. [S.l.], 2007. p. 50–60. Citado na página 43.

TUCKER, A.; NOONAN, R. *Linguagens de programação: princípios e paradigmas*. McGraw-Hill, 2009. ISBN 9788577260447. Disponível em: <https://books.google.com.br/books?id=v3OrPgAACAAJ>. Citado na página 49.

VALENTE, M. T. Engenharia de software moderna: Princípios e práticas para desenvolvimento de software com produtividade. 2020. Citado 8 vezes nas páginas 24, 25, 30, 31, 32, 33, 34 e 44.

VARELLA, W. A. *Arquitetura de solução de computação em nuvem*. [S.l.]: Editora Senac São Paulo, 2019. Citado 2 vezes nas páginas 40 e 41.

VERAS, M. Computação em nuvem: Nova arquitetura de ti. *Rio de Janeiro: Brasport*, 2015. Citado na página 41.