CENTRO UNIVERSITÁRIO DO DISTRITO FEDERAL – UDF COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

EventsLink: Sistema de Gerenciamento de Eventos Acadêmicos

Artur Dantas Martins

Bernardo de Melo Matuchewski - 31957226

Paulo Abdiel Sardinha de Sousa Santos

BRASÍLIA

2025

UDF Centro Universitário

Trabalho de conclusão de curso apresentado à Coordenação de Sistemas de Informação, do Centro Universitário do Distrito Federal - UDF, como requisito parcial para obtenção do grau de bacharel em Ciência da Computação.

Orientador: Profa. Dra. Leticia Toledo Maia Zoby

BRASÍLIA

2025

OBS.: Esta folha deverá ser impressa no verso da folha de rosto (folha anterior)

Sobrenome, Nome.

Título : subtítulo / Nome Sobrenome. -- Brasília, 2021.

xx f. *(quantidade de folhas da monografia)*

Orientador: XXXXXXX.

Trabalho de conclusão de curso (Graduação – Direito) -- Centro Universitário do Distrito Federal – UDF. Coordenação de Direito, Brasília, DF, 2021.

1. Assunto. 2. Assunto. 3. Assunto. I. Título.

C

DU: (consultar na biblioteca)

Artur Dantas Martins

Bernardo de Melo Matuchewski

Paulo Abdiel Sardinha de Sousa Santos

Sistema de Gerenciamento de Eventos Acadêmicos

EventsLink

Trabalho de conclusão de curso apresentado à Coordenação de Sistemas de Informação, do Centro Universitário do Distrito Federal - UDF, como requisito parcial para obtenção do grau de bacharel em Ciência da Computação..

Orientador: Prof. Dra. Leticia Toledo Maia Zoby

Brasília, de _____de 2025.

Banca Examinadora

NOME DO EXAMINADOR

Titulação Instituição a qual é filiado

NOME DO EXAMINADOR

Titulação Instituição a qual é filiado

NOME DO EXAMINADOR

Titulação Instituição a qual é filiado

NOTA: _____

AGRADECIMENTOS

Gostaríamos de registrar aqui nossa sincera e profunda gratidão a Deus, fonte de inspiração e força ao longo de toda essa jornada. Em cada passo deste projeto, sentimos Sua presença nos guiando e concedendo coragem, paciência e discernimento para enfrentar os desafios que surgiram. A Ele, que nos amparou em momentos de incerteza e nos abençoou com momentos de vitória, dedicamos nossos sinceros agradecimentos.

Também dedicamos nossa gratidão às nossas famílias, que, com amor e paciência, foram nosso alicerce desde o início. Sem o apoio, a compreensão e as palavras de encorajamento de cada um deles, este trabalho não teria sido possível. Foram eles que compreenderam nossos sacrifícios, dividiram nossas alegrias e, nos momentos de dúvida, nos lembraram de nossa própria força.

RESUMO

Este Trabalho de Conclusão de Curso apresenta o *EventsLink*, um sistema de gestão de eventos acadêmicos desenvolvido para o Centro Universitário do Distrito Federal (UDF). A plataforma tem como objetivo centralizar a divulgação, inscrição e acompanhamento de atividades como palestras e workshops, facilitando a comunicação entre alunos e instituição. Com funcionalidades como consulta de horários, emissão de certificados e registro de presença, o sistema também contribui para a validação de horas complementares. O aplicativo mobile foi desenvolvido em Flutter, garantindo interfaces responsivas, enquanto o backend utiliza Spring Boot, com integração a um banco de dados relacional PostgreSQL. A arquitetura do sistema prioriza escalabilidade, modularidade e segurança. O *EventsLink* oferece uma experiência acessível e organizada tanto para estudantes quanto para administradores, promovendo maior engajamento e eficiência na gestão de eventos acadêmicos. Assim, torna-se uma ferramenta essencial para aprimorar a comunicação institucional e fortalecer a participação da comunidade universitária.

Palavras-chave: Eventos, Aplicativo, EventsLink, Comunicação, Universitária.

ABSTRACT

This thesis presents EventsLink, an academic event management system developed for the Centro Universitário do Distrito Federal (UDF). The system aims to centralize the dissemination, registration, and participation tracking of activities such as lectures and workshops, addressing the challenges posed by informal communication in universities. EventsLink offers features including event registration, schedule and location access, and issuance of participation certificates. Its architecture ensures scalability, modularity, and security through the use of

modern technologies. The mobile application, built with Flutter, provides a responsive and user-friendly interface across platforms, while the backend, developed with Spring Boot, manages data and system functionalities. Integrated with a PostgreSQL relational database, the system efficiently organizes user, event, and attendance data. Additionally, it facilitates the recording and validation of complementary hours, a key academic requirement. With an organized and accessible interface for students and administrators alike, it promotes student engagement through a complete academic event management solution.

Keywords: Events, Application, EventsLink, Academic, System.

UDF Centro Universitário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UDF - Centro Universitário do Distrito Federal

TCC - Trabalho de Conclusão de Curso

MVVM - Model-View-ViewModel

MVC - Model-View-Controller

SOLID - Princípios de design de software

BD - Banco de Dados

API - Interface de Programação de Aplicações

UI - Interface do Usuário

PK - Primary Key (Chave Primária)

FK - Foreign Key (Chave Estrangeira)

UUID - Identificador Universalmente Único

CEP - Código de Endereçamento Postal

UDF Centro Universitário

SUMÁRIO

1. INTRODUÇÃO

O ensino superior no Brasil tem experimentado uma expansão significativa nas últimas décadas. De acordo com o Instituto Nacional De Estudos e Pesquisas Educacionais Anísio Teixeira (INEP, 2023), o país contava com mais de 8,7 milhões de matrículas em cursos de graduação em 2022, distribuídas entre instituições públicas e privadas. Esse crescimento é acompanhado por desafios em termos de qualidade, infraestrutura e organização acadêmica, exigindo soluções tecnológicas que possam melhorar a gestão das atividades universitárias. Dentre essas atividades, destaca-se a importância dos eventos extracurriculares, que complementam a formação dos estudantes e proporcionam experiências práticas e de interação com o meio profissional.

Um dos principais instrumentos regulatórios da formação acadêmica no Brasil é a exigência de horas complementares, previstas nas Diretrizes Curriculares Nacionais (DCNs). De acordo com o Ministério da Educação (MEC, 2019), essas horas devem ser integralizadas por meio de atividades extracurriculares, como participação em eventos, projetos de extensão, iniciação científica, entre outros. A obrigatoriedade dessas atividades visa ampliar a formação do aluno além do conteúdo programático, promovendo o desenvolvimento de habilidades e competências essenciais para o mercado de trabalho. Nesse contexto, torna-se necessário um controle eficiente das participações dos alunos, tanto por parte das instituições quanto dos próprios discentes.

No Centro Universitário do Distrito Federal (UDF), onde este trabalho foi desenvolvido, são registrados mais de 20.000 estudantes matriculados nos diversos cursos de graduação e pós-graduação. A instituição, referência na região, promove frequentemente eventos acadêmicos, como palestras, congressos e workshops, os quais se tornam oportunidades valiosas para o cumprimento das exigências do MEC e para o enriquecimento da vivência universitária. No entanto, a organização e divulgação desses eventos ainda ocorrem, em muitos casos, por meios informais, como grupos de mensagens ou redes sociais, o que dificulta o controle de presença, emissão de certificados e registro das atividades para fins acadêmicos. Essa dificuldade é

percebida de acordo com o Apêndice N, após um usuário apresentar dúvidas em relação a palestras na universidade, que valem horas complementares.

Diante dessa realidade, surge o desenvolvimento do EventsLink, uma plataforma tecnológica voltada à gestão de eventos acadêmicos, com foco na centralização das informações, controle de participação, emissão automatizada de certificados e integração com as diretrizes institucionais. A ferramenta foi idealizada para atender tanto aos estudantes, com uma interface mobile acessível, quanto aos organizadores e coordenadores, com um painel web completo para gerenciar os eventos. A adoção do sistema visa transformar a maneira como as atividades extracurriculares são organizadas e acessadas na UDF, promovendo maior eficiência, engajamento e rastreabilidade no cumprimento das horas complementares exigidas.

1.1 Objetivo

Este Trabalho de Conclusão de Curso tem como objetivo desenvolver o EventsLink, uma aplicação multiplataforma para gestão de eventos acadêmicos. A proposta é facilitar a divulgação, inscrição e controle de participação em atividades extracurriculares.

1.1.2 Objetivos específicos

1.2. Trabalhos Correlatos

Um Sistema de Gerenciamento de Eventos é um software online projetado para simplificar a organização de eventos como congressos, simpósios, workshops e outros tipos de eventos esportivos, sociais, culturais ou científicos. Ele é utilizado em todas as fases do processo, desde a coleta de inscrições e submissões de trabalhos até as etapas finais, incluindo a emissão de certificados, a apresentação de estatísticas e a geração de relatórios.

O Sympla (2024) é uma plataforma brasileira que se destaca no mercado de gerenciamento e venda de ingressos para eventos. Fundada em 2012, a plataforma oferece uma solução completa para a criação e administração de eventos, desde a emissão de ingressos até o controle de acesso. O Sympla se diferencia por sua interface amigável e suas funcionalidades diversificadas, que incluem personalização de eventos, integração com redes sociais e ferramentas analíticas. A plataforma se destaca pela sua capacidade de integrar soluções de pagamento e proporcionar uma experiência fluida tanto para organizadores quanto para participantes.

O Even3 (2024) é uma outra solução popular no Brasil para a gestão de eventos, com foco em eventos acadêmicos e corporativos. Lançada em 2013, a plataforma oferece funcionalidades voltadas para a organização de conferências, seminários e workshops, com ênfase em aspectos como submissão de trabalhos, gerenciamento de inscrições e emissão de certificados.

O Eventim (2024) é uma conhecida principalmente pela venda de ingressos para grandes eventos, como shows e esportes. Fundada na Alemanha em 1989 e expandida para diversos países, a Eventim oferece um sistema robusto para a gestão de ingressos e o controle de acesso, com uma forte presença em eventos de grande porte e popularidade.

. SIQUEIRA e SILVA (2018) descrevem o desenvolvimento de uma aplicação voltada para o gerenciamento de eventos específicos da universidade UniEVANGÉLICA. A aplicação é focada no próprio evento e inclui funcionalidades como sorteio de participantes e geração de relatórios para o acompanhamento do evento. No entanto, o sistema não inclui a funcionalidade de geração automatizada de certificados.

ALMEIDA (2020), apresenta o desenvolvimento de uma aplicação web voltada para a gestão de um evento anual específico realizado na Universidade Federal Fluminense (UFF). A aplicação inclui funcionalidades para a administração do evento, a geração e envio de certificados, bem como o controle dos pagamentos de inscrições. Contudo, o sistema foi

projetado para gerenciar apenas um único grupo de eventos e não foi planejado para ser adaptado ou disponibilizado para outras instituições.

LEMOS, ZANATTA e ZAINA (2020) descrevem resumidamente a proposta para o desenvolvimento de um sistema de gerenciamento de eventos destinado às atividades do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP). No entanto, não são fornecidos muitos detalhes sobre as funcionalidades planejadas para o sistema.

(Mais um paragrafo relatando as coletas mais importantes para a solução proposta. Um diferencial em comparação aos outros)

1.3 Solução proposta

Em muitas universidades, inclusive no Centro Universitário do Distrito Federal (UDF), a divulgação de eventos acadêmicos ainda é feita de forma descentralizada e informal, muitas vezes por redes sociais, murais ou grupos de mensagens. Esse tipo de comunicação, embora comum, acaba dificultando o acesso dos alunos a informações importantes, como datas, horários, locais e, principalmente, detalhes sobre o aproveitamento acadêmico desses eventos – como as horas complementares.

Essa dificuldade foi observada de forma recorrente pelos estudantes e até mesmo pelos professores da instituição de ensino. Eventos relevantes deixavam de ser frequentados por falta de divulgação clara, prazos eram perdidos e a gestão da participação em atividades extracurriculares acabava sendo desorganizada e pouco eficiente. Essas constatações serviram como ponto de partida para a motivação do projeto.

Diante desse cenário, surgiu a proposta do desenvolvimento do EventsLink, um sistema de gerenciamento de eventos acadêmicos que tem como objetivo centralizar a divulgação, inscrição e controle de participação em palestras, workshops e demais atividades universitárias. A plataforma foi idealizada para atender tanto os estudantes, por meio de um aplicativo mobile

com interface intuitiva e acessível, quanto os gestores e organizadores da instituição, por meio de um sistema web com funcionalidades administrativas.

A proposta visa não apenas facilitar o acesso dos alunos às atividades extracurriculares, mas também tornar mais eficiente a divulgação dos eventos e o processo de gestão desses eventos, contribuindo para uma melhor organização institucional. Com o EventsLink, pretende-se promover uma comunicação mais clara, uma participação mais ativa dos alunos e uma administração mais prática e integrada dos eventos acadêmicos, impactando positivamente a experiência universitária como um todo.

1.4 Conteúdo e Organização

No **Capítulo 1**, são apresentados a introdução do problema e os objetivos do projeto. No **Capítulo 2**, é realizada a revisão bibliográfica, proporcionando a base teórica necessária para o entendimento do tema e das tecnologias utilizadas. O **Capítulo 3** contém o desenvolvimento do sistema, incluindo as etapas de modelagem, escolha das arquiteturas e implementação das principais funcionalidades. O **Capítulo 4** contém os resultados parciais, como pesquisas ou design do sistema. O **Capítulo 5** contém a síntese do projeto.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos teóricos e técnicos que fundamentam o desenvolvimento do sistema EventsLink. São abordadas as metodologias de engenharia de software, arquiteturas modernas, linguagens de programação, banco de dados, ferramentas de desenvolvimento e modelagem UML. A fundamentação tem como objetivo justificar, com base em estudos e autores consagrados, as decisões técnicas adotadas no projeto e demonstrar a adequação da solução proposta ao contexto acadêmico.

(Falar sobre o MEC e extensão universitária)

2.1. Engenharia de software

SOMMERVILLE (2011) apresenta uma definição abrangente de software, destacando sua natureza intangível e abstrata. O autor argumenta que, diferentemente do hardware, o software não está limitado pelas propriedades físicas dos materiais e não segue as mesmas leis da física. Essa característica confere ao software uma flexibilidade única, permitindo que ele seja adaptado e modificado de forma mais ágil.

Além disso, SOMMERVILLE (2011) expande o conceito de software para além do simples programa executável. Ele inclui nesse conjunto a documentação técnica, que descreve a estrutura interna do sistema e os processos de desenvolvimento; a documentação do usuário, que orienta o usuário final na utilização do software; dados de configuração, que personalizam o sistema para diferentes ambientes; e, é claro, o próprio código fonte. Essa visão holística do software enfatiza a importância de todos esses elementos para a criação de um sistema de software completo e funcional.

Segundo PRESSMAN (2011), a Engenharia de Software é uma tecnologia em camadas que envolve um processo, além de um conjunto de métodos e práticas que possibilitam o desenvolvimento de software com foco na melhoria contínua. À medida que os requisitos de tecnologia da informação se tornam mais complexos a cada ano, a Engenharia de Software foi

criada para transformar o desenvolvimento de software em um processo sistemático, disciplinado, quantificável e adaptável.

Essa abordagem visa garantir que o desenvolvimento de software seja eficiente e capaz de lidar com as crescentes demandas, ao mesmo tempo que mantém a qualidade e a capacidade de evolução dos sistemas.

Figura 1 - Camadas da Engenharia de Software

Fonte: PRESSMAN (2011)

Compreendida a importância da engenharia de software para a estruturação de sistemas robustos e escaláveis, é possível explorar agora as metodologias ágeis que direcionaram o processo de desenvolvimento do EventsLink

2.1.1 Metodologia Ágil

A metodologia ágil é um conjunto de práticas e princípios que busca otimizar o desenvolvimento de projetos, especialmente no campo de software, através da flexibilidade e da colaboração. Segundo SUTHERLAND (2014), um dos criadores da Metodologia Ágil e do Scrum, afirma que essa estrutura tem como objetivo aproveitar a forma de como as equipes trabalham, oferecendo para as equipes ferramentas de auto-organização e, o mais importante, melhorar a velocidade e a qualidade do trabalho.

Sua origem remonta ao Manifesto Ágil, publicado em 2001, que propõe quatro valores fundamentais:

- **Indivíduos e interações acima de processos e ferramentas**: A comunicação eficaz entre os membros da equipe é priorizada em relação ao uso de ferramentas ou a aderência estrita a processos.
- **Software funcionando acima de documentação abrangente**: A entrega de um produto funcional é considerada mais importante do que a produção de uma documentação extensa. Embora a documentação seja importante, o foco deve ser na criação de um valor real.
- **Colaboração com o cliente acima da negociação de contratos**: O envolvimento ativo do cliente durante o processo de desenvolvimento é essencial. O feedback contínuo permite ajustes e melhorias, assegurando que o produto final atenda às necessidades do usuário.
- **Responder a mudanças acima de seguir um plano**: As equipes ágeis são encorajadas a ser adaptáveis e a reagir rapidamente a novas informações ou mudanças de requisitos, em vez de se prender a um plano fixo.

A adoção da metodologia ágil traz diversos benefícios para o desenvolvimento de software. De acordo com BECK (2001), autor do Manifesto Ágil, a flexibilidade é um dos pilares fundamentais, permitindo que as equipes respondam rapidamente a mudanças de requisitos e adaptem o produto de acordo com novas necessidades.

A colaboração contínua com o cliente é também enfatizada, sendo apontada por HIGHSMITH (2002) como um fator essencial para o sucesso do projeto, pois o envolvimento ativo do cliente permite ajustes frequentes e aumenta a aderência do produto às expectativas dos usuários. A satisfação do cliente, conforme destacado por SUTHERLAND (2014), é maximizada no desenvolvimento ágil devido à entrega contínua de valor, com funcionalidades incrementais sendo disponibilizadas rapidamente para avaliação. Além disso, segundo PRESSMAN (2011), a qualidade do produto é aprimorada através do feedback constante e das entregas iterativas,

possibilitando a identificação e correção precoce de defeitos, o que reduz os custos de retrabalho e aumenta a confiabilidade do sistema final.

A Metodologia Ágil possui também algumas abordagens, cada uma com suas características que influenciam no seu desenvolvimento, sendo elas:

- **Scrum**: Uma das abordagens mais populares, que divide o trabalho em sprints (ciclos curtos de desenvolvimento). Envolve papéis específicos, como o Scrum Master e o Product Owner, e eventos como reuniões diárias e revisões de sprint.
- **Kanban**: Foca na visualização do trabalho em andamento e na limitação do trabalho em progresso. Utiliza um quadro Kanban para gerenciar o fluxo de trabalho, permitindo que as equipes visualizem suas tarefas e priorizem o que deve ser feito a seguir.
- **Extreme Programming (XP)**: Enfatiza práticas técnicas como programação em par, desenvolvimento orientado a testes (TDD) e integração contínua. O objetivo é melhorar a qualidade do software e responder rapidamente a mudanças.
- **Crystal**: Um conjunto de métodos que enfatiza a adaptabilidade e a comunicação entre os membros da equipe, permitindo que as práticas sejam moldadas com base nas necessidades específicas do projeto.
- **Feature-Driven Development (FDD)**: Uma abordagem centrada em funções, onde o desenvolvimento é organizado em torno da entrega de recursos específicos. O FDD é orientado a um planejamento e desenvolvimento mais predefinidos.

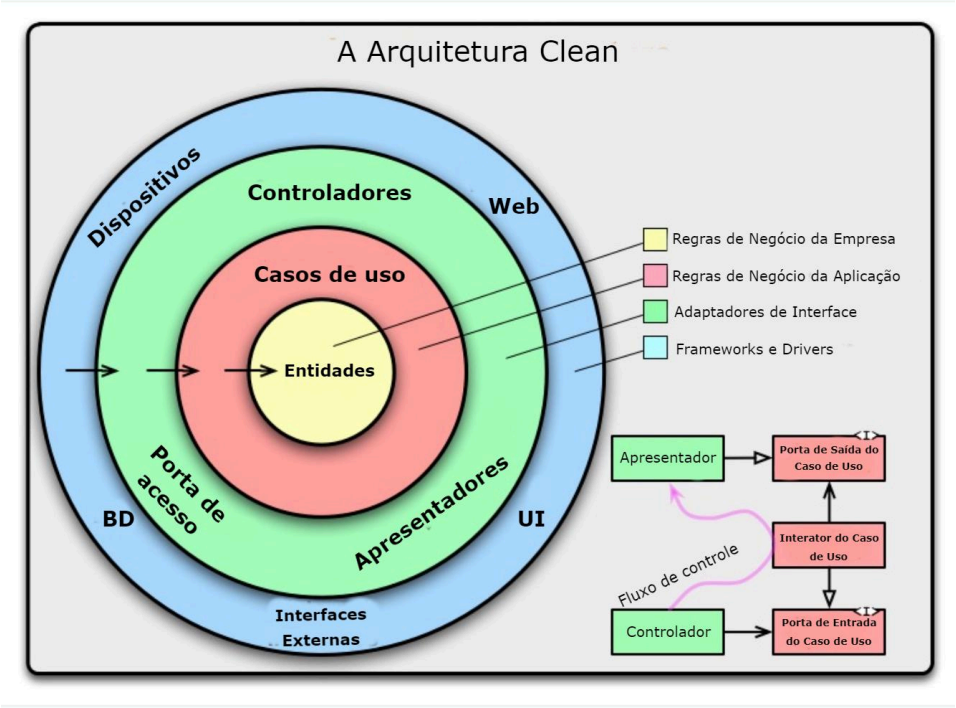
Dentre as abordagens modernas de desenvolvimento de software, destaca-se a metodologia ágil, que orientou o planejamento do sistema aqui proposto.

2.2. Arquiteturas de software

2.2.1 Clean Architecture

Segundo o site PROGRAMMING PULSE (2023), a Clean Architecture, popularizada por Robert C. Martin, divide o software em camadas concêntricas, protegendo as regras de negócio de influências externas. Essa abordagem garante flexibilidade, estabilidade e independência do núcleo do sistema em relação a frameworks e tecnologias externas. As principais características da Clean Architecture incluem:

Figura 2 - Arquitetura Clean



Ator: Robert C. Martin

- **Separação de preocupações**: Cada camada tem uma responsabilidade distinta, como a lógica de negócios, a interface do usuário e a comunicação externa.

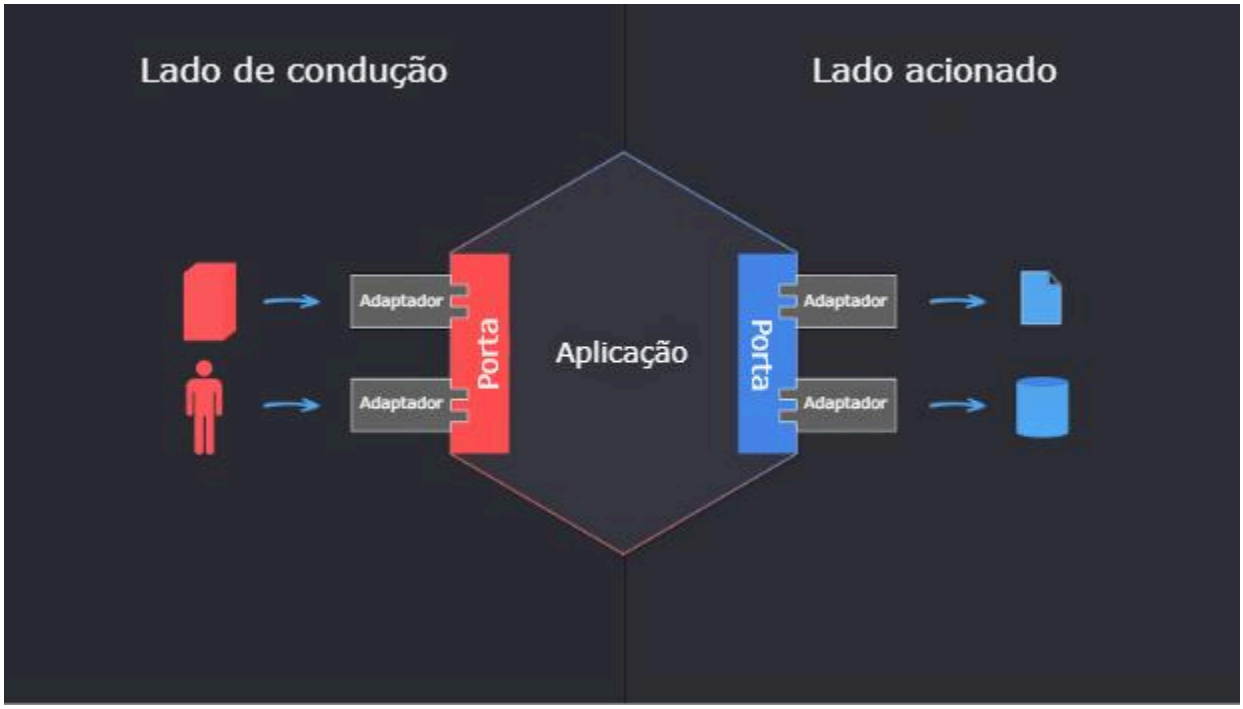
- **Independência do framework**: O núcleo do sistema não depende de frameworks externos, permitindo mudanças sem impactar a lógica de negócios.
- **Reutilização e manutenção**: Facilita a reutilização de componentes e a manutenção, especialmente em sistemas de grande escala.

Essa arquitetura é amplamente usada para sistemas complexos que exigem uma base sólida e sustentável a longo prazo. Dessa forma, a Clean Architecture se mostra ideal para sistemas como o EventsLink, que exigem separação clara de responsabilidades e alta manutenibilidade.

2.2.2 Arquitetura Hexagonal

De acordo com o PROGRAMMING PULSE (2023), a Arquitetura Hexagonal, também conhecida como Ports and Adapters, propõe uma separação clara entre o núcleo da aplicação e suas interações com o mundo externo (como bancos de dados e APIs). Suas principais características incluem:

Figura 3 - Arquitetura Hexagonal



Ator: Pablo Martinez

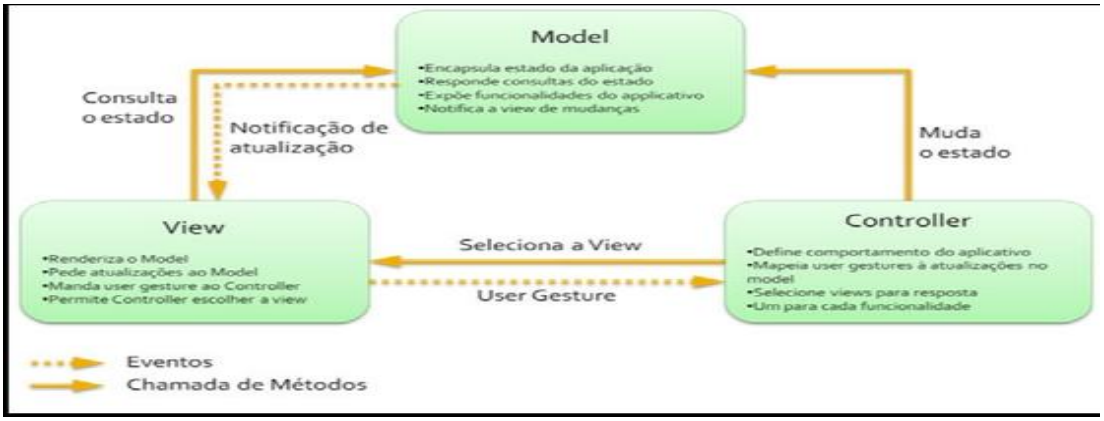
- **Desacoplamento**: A aplicação é desacoplada de detalhes de implementação externos, como interfaces de usuário ou sistemas de persistência.
- **Testabilidade**: Facilita a criação de testes ao permitir que interações externas sejam simuladas ou substituídas.
- **Flexibilidade**: Adapta-se a mudanças externas, permitindo a integração com novos sistemas sem modificar o núcleo da aplicação.

Essa arquitetura é eficaz em projetos que precisam de flexibilidade na comunicação com sistemas externos.

2.2.3 MVC (Model-View-Controller)

Ainda segundo o PROGRAMMING PULSE (2023), o padrão MVC é amplamente utilizado em aplicações web e divide o código em três componentes principais.

Figura 4 - Arquitetura MVC



Ator: Zemel

- **Model**: Responsável pela lógica de negócios e pelos dados.
- **View**: Cuida da interface de usuário e da apresentação visual.
- **Controller**: Atua como intermediário entre o Model e o View, processando as entradas do usuário e atualizando os dados e a interface.

Embora seja ideal para aplicações simples, o MVC pode se tornar difícil de gerenciar em sistemas mais complexos devido ao acoplamento entre seus componentes.

2.2.4 Princípios SOLID

Os princípios SOLID, de acordo com o PROGRAMMING PULSE (2023), orientam o design de software orientado a objetos, melhorando a modularidade e a manutenção do código. Esses princípios incluem:

Figura 5 - Arquitetura SOLID

Ator: Anncode

- **Princípio da Responsabilidade Única (SRP)**: Cada classe deve ter uma única responsabilidade.
- **Princípio da Inversão de Dependência (DIP)**: Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.

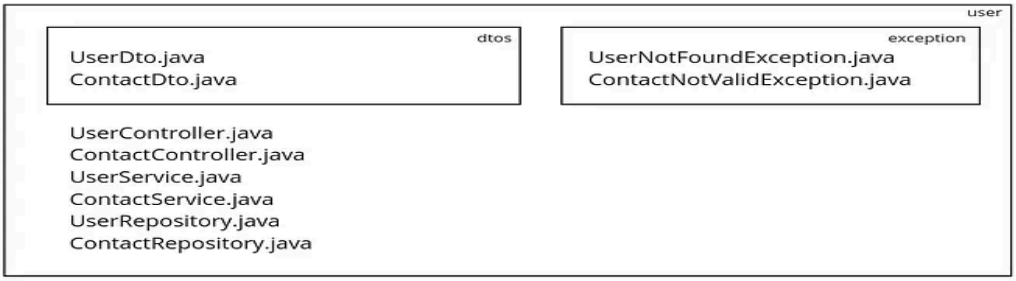
Os princípios SOLID são frequentemente aplicados em conjunto com arquiteturas como Clean Architecture e Hexagonal Architecture, promovendo código mais organizado e de fácil manutenção.

2.2.5 Package by Feature

O padrão da arquitetura Package by Feature é comumente conhecido pela sua abordagem popular ao empacotamento, como descrito por PHAUER (2020). Assim, cada módulo possui suas respectivas classes, organizadas com base no contexto funcional ao qual pertencem, ela é inspirada em princípios de Hexagonal Architecture, Clean Architecture, MVC e SOLID. Por exemplo, ao utilizar o contexto de "user" em um sistema, todas as classes relacionadas à

funcionalidade de usuários – como controladores, serviços, repositórios e entidades – estariam agrupadas em um único pacote ou módulo específico para "user".

Figura 6 - Arquitetura Feature



Fonte: Tabnews

Essa abordagem evita a fragmentação do código entre várias camadas técnicas, como acontece no empacotamento por camadas (ex.: separar controladores, serviços e repositórios em pastas diferentes), o que facilita o entendimento e a manutenção do sistema. No contexto de "user", haveria um pacote dedicado que conteria todas as classes necessárias para gerir as operações relacionadas aos usuários, como criação, autenticação, atualização de dados e permissões. Dessa forma, os desenvolvedores podem concentrar-se em um único espaço para modificar ou adicionar funcionalidades relacionadas a usuários, o que aumenta a coesão e reduz o acoplamento entre as diferentes partes do sistema.

Além disso, essa abordagem modular melhora a escalabilidade e a testabilidade do sistema, já que cada módulo de feature é tratado como uma unidade autônoma. No exemplo de "user", testes unitários e de integração podem ser realizados apenas nesse módulo sem afetar ou depender de outros módulos. Com isso, a manutenção e a introdução de novas funcionalidades se tornam mais rápidas e seguras, visto que as mudanças ficam isoladas a um único contexto funcional.

Em projetos de grande escala, como sistemas de gerenciamento de eventos, essa organização permite que equipes diferentes trabalhem em paralelo em módulos distintos, como "user", "event", "payment", entre outros, sem interferências ou conflitos. Isso otimiza o processo de desenvolvimento colaborativo e reduz os riscos de erros ao adicionar ou modificar funcionalidades.

2.2.6 MVVM (Model-View-ViewModel)

O padrão de arquitetura MVVM (Model-View-ViewModel) se assemelha ao padrão MVC, e é mais comumente usado no desenvolvimento de aplicativos móveis, separando o código também em três camadas distintas:

Figura 7 - Arquitetura MVVM

Ator: Gabriel Sanzone (Medium)

- **Model**: Responsável pela lógica de negócios e pelos dados.
- **View**: Cuida da interface de usuário e da apresentação visual.
- **ViewModel**: Atua como intermediário entre o Model e o View, processando as entradas do usuário e atualizando os dados e a interface.

A abordagem MVVM oferece várias vantagens no desenvolvimento de aplicativos móveis. Ao dividir claramente as responsabilidades, facilita a realização de mudanças e a adição de novos recursos, já que cada parte do sistema tem um papel específico. Além disso, a

modularidade do MVVM promove a reutilização de componentes em diferentes seções do aplicativo, o que otimiza tempo e reduz o esforço durante o desenvolvimento.

2.3. Banco de dados

Segundo Carvalho (2017), no livro *PostgreSQL: Banco de Dados para Aplicações Web Modernas*, os sistemas de banco de dados desempenham um papel crucial no suporte às operações e à tomada de decisões em diferentes níveis de uma empresa, desde a operação até a gerência. Tanto grandes corporações quanto pequenos empreendedores dependem da persistência de dados para manter o funcionamento eficaz de suas atividades. Com a evolução tecnológica, os bancos de dados passaram a ser hospedados na nuvem, o que reduziu significativamente os custos de implementação, além de permitir uma melhor escalabilidade e, em alguns casos, menores tempos de resposta (latência), dependendo da infraestrutura oferecida pelos provedores de serviços de nuvem.

Essa transição para a nuvem não apenas otimizou os custos, mas também atendeu às crescentes demandas do mundo moderno, onde a quantidade de dados gerados e processados aumentou exponencialmente. De acordo com o conceito dos 3Vs do Big Data, descrito por Taution (2013) no livro *Big Data*, os desafios relacionados ao volume, velocidade e variedade dos dados tornam a escolha de um sistema de banco de dados e a sua implementação em nuvem uma decisão estratégica crítica para os departamentos de TI e para os CIOs (Chief Information Officers). A capacidade de processar grandes quantidades de dados em tempo real, de forma eficiente e flexível, é essencial para lidar com as demandas atuais de dados massivos.

Dentre as diversas soluções de banco de dados disponíveis, duas grandes categorias se destacam: os bancos de dados relacionais e não relacionais. O modelo relacional, proposto originalmente por Edgar F. Codd na década de 1970, organiza dados em tabelas (ou relações) interligadas por chaves primárias e estrangeiras. Esse modelo, utilizado em sistemas como o PostgreSQL, oferece robustez na integridade dos dados, garantida por meio de transações ACID

(Atomicidade, Consistência, Isolamento e Durabilidade). Além disso, bancos de dados relacionais utilizam a linguagem SQL, que permite consultas estruturadas, flexíveis e poderosas, adequadas para a maioria das aplicações empresariais e sistemas críticos.

Por outro lado, com o surgimento do Big Data e a necessidade de lidar com dados não estruturados, bancos de dados não relacionais (ou NoSQL) começaram a ganhar destaque. Esses sistemas oferecem maior flexibilidade na modelagem de dados, permitindo o armazenamento de documentos, grafos e dados chave-valor de maneira mais eficiente para determinados tipos de aplicações. Exemplos de bancos de dados não relacionais incluem MongoDB, Cassandra e Redis, que são especialmente úteis em aplicações que demandam alta escalabilidade e velocidade de resposta.

2.3.1 PostgreSQL

De acordo com Carvalho (2017), no livro *PostgreSQL: Banco de Dados para Aplicações Web Modernas*, o PostgreSQL é um poderoso sistema de gerenciamento de banco de dados objeto-relacional de código aberto, lançado originalmente em 1996. Embora tenha começado de maneira modesta, foi gradualmente ganhando espaço nas empresas e se consolidou como uma das soluções mais robustas e seguras do mercado. Em 2023, foi considerado o banco de dados mais popular, superando outras opções como MySQL, SQLite e MongoDB.

Dentre suas principais características, o PostgreSQL se destaca por:

- **Conformidade com padrões SQL**: Totalmente aderente ao padrão ANSI SQL, o que garante interoperabilidade com diversas outras tecnologias e facilita a migração de outros bancos de dados.
- **Extensibilidade**: Permite que os usuários criem seus próprios tipos de dados, funções e operadores, tornando-o altamente adaptável a diferentes necessidades.
- **Segurança e Controle de Acesso**: Oferece um conjunto abrangente de ferramentas de controle de acesso e autenticação de usuários, garantindo segurança em sistemas complexos.

- **Transações ACID**: Garantindo que operações em banco de dados sejam atômicas, consistentes, isoladas e duráveis, o PostgreSQL oferece robustez para manipulação de dados críticos.
- **Alta Disponibilidade e Escalabilidade**: Possui mecanismos nativos de replicação e alta disponibilidade, além de suportar grande quantidade de dados sem perda de performance.

Algumas das suas limitações, ainda de acordo com *PostgreSQL: Banco de Dados para Aplicações Web Modernas:*

- Tamanho máximo do banco de dados: ilimitado;
- Tamanho máximo de uma tabela: 32TB;
- Tamanho máximo de uma linha: 1.6TB;
- Tamanho máximo de um campo: 1GB;
- Máximo de linhas de um campo: 1GB;
- Máximo de linhas por tabela: ilimitado;
- Máximo de colunas por tabela: 250-1600, dependendo do tipo de coluna;
- Máximo de índices por tabela;

2.4. UML

De acordo com Booch (2006), no livro *UML: Guia do Usuário*, a UML (Linguagem Unificada de Modelagem) é uma linguagem gráfica para a visualização, especificação, construção e documentação de artefatos de sistemas de software complexos. A UML é amplamente ensinada nas universidades e utilizada por empresas para modelagem de sistemas. Entre os diagramas mais utilizados estão o diagrama de casos de uso e o diagrama de classes.

A UML foi desenvolvida pela Object Management Group (OMG) em meados da década de 1990, com o objetivo de padronizar as notações usadas na análise e projeto de sistemas orientados a objetos. Antes de sua criação, havia diversas metodologias diferentes, o que dificultava a comunicação entre equipes de desenvolvimento. A UML resolveu esse problema ao

unificar essas abordagens em uma linguagem comum, que é amplamente reconhecida e utilizada globalmente.

A UML é composta por diversos tipos de diagramas, cada um focado em diferentes aspectos do sistema. Os mais utilizados são:

- Diagrama de Casos de Uso: Focado em descrever as interações entre os usuários (atores) e o sistema, identificando as funcionalidades (casos de uso) que o sistema deve oferecer. Ele é especialmente útil nas fases iniciais de levantamento de requisitos, pois ajuda a definir o escopo do projeto.
- Diagrama de Classes: Essencial para o projeto orientado a objetos, este diagrama descreve a estrutura do sistema, especificando as classes, seus atributos, métodos e os relacionamentos entre elas (associações, heranças, dependências). O diagrama de classes é amplamente utilizado durante o design do sistema e serve como base para a implementação.

Outros diagramas importantes incluem:

- Diagrama de Sequência: Representa a interação entre objetos no decorrer do tempo, mostrando a ordem em que as mensagens são trocadas entre eles. É muito útil para entender o fluxo de comunicação dentro do sistema.
- Diagrama de Atividades: Descreve o fluxo de trabalho ou o comportamento de um sistema em termos de atividades. Ele é útil para modelar processos de negócios e lógicas complexas.
- Diagrama de Componentes: Usado para representar a estrutura física do sistema, mostrando como os diferentes módulos ou componentes de software se integram para formar o sistema completo.

Apesar de suas inúmeras vantagens, a UML também apresenta alguns desafios. Um dos principais é que, em projetos grandes e complexos, os diagramas podem se tornar muito extensos

e difíceis de gerenciar. Além disso, a UML, em si, não prescreve como deve ser implementada, o que pode gerar variações significativas dependendo da equipe e do contexto do projeto.

Outro desafio é que, em equipes que adotam metodologias ágeis, o uso extensivo da UML pode ser visto como "excesso de documentação", o que contraria o princípio ágil de foco no desenvolvimento rápido e iterativo.

Ainda de acordo com Booch (2006), no livro *UML: Guia do Usuário*, a UML (Linguagem Unificada de Modelagem), a UML se destina principalmente a sistemas complexos de software, como:

- Sistemas de informações corporativos
- Serviços bancários e financeiros
- Telecomunicações
- Transportes
- Serviços distribuídos baseados em WEB
- Vendas de Varejo

A utilização de diagramas UML foi essencial para documentar e validar o comportamento do sistema durante o desenvolvimento, servindo como ponte entre os requisitos e a implementação.

2.5. Ferramentas de desenvolvimento

As principais ferramentas utilizadas no desenvolvimento.

A. Draw.io: É uma ferramenta online gratuita usada para a criação de diagramas, fluxogramas, e outros gráficos visuais. Muito utilizada para representar visualmente casos de uso, diagramas de classes, e outras documentações relacionadas ao desenvolvimento de software.

B. Visual Studio Code: É uma IDE (Integrated Development Environment) leve, mas poderosa, que suporta desenvolvimento web, de desktop e mobile. Suporta várias linguagens de

programação e vem com recursos de depuração, controle de versão integrado (Git) e suporte a extensões que aumentam suas funcionalidades.

C. Docker: É uma plataforma de software que permite criar, gerenciar e executar aplicações em contêineres. Os contêineres isolam as aplicações e suas dependências, tornando mais fácil desenvolver, testar e distribuir software de maneira consistente em diferentes ambientes.

D.Github: É uma plataforma de hospedagem de código-fonte e colaboração baseada em Git. Permite que desenvolvedores gerenciem o versionamento do código, colaborem com outros desenvolvedores, revisem código e gerenciem projetos de desenvolvimento de software.

3. DESENVOLVIMENTO DO TRABALHO

Neste capítulo são apresentados os elementos para o desenvolvimento do Sistema de Gerenciamento de Eventos Acadêmicos (EventsLink) no Centro Universitário do Distrito Federal.

3.1. Metodologia

No desenvolvimento de software, há uma sequência de etapas essenciais para garantir um sistema bem estruturado. Essas etapas incluem a modelagem do banco de dados, a criação de diagramas UML, o design da interface do usuário (UI), a definição da arquitetura e, por fim, a codificação. Cada uma dessas fases contribui para o sucesso do projeto, mas a escolha da arquitetura é um dos pontos mais críticos, tanto para o backend quanto para o frontend. Uma má escolha pode resultar em um projeto simples que se torna complexo e demorado para se desenvolver.

Segundo o site PROGRAMMING PULSE (2023), atualmente, existem várias arquiteturas pré-definidas, desenhadas para maximizar a produção e a eficiência no desenvolvimento de software. Entre elas, destacam-se a Clean Architecture, a Arquitetura

Hexagonal, o MVC (Model-View-Controller), MVVM (Model-View-ViewModel) e os princípios SOLID. Essas arquiteturas são amplamente utilizadas devido à sua capacidade de organizar o código, facilitar a manutenção e garantir a escalabilidade dos sistemas.

Ainda de acordo com o site PROGRAMMING PULSE (2023), a importância de escolher a arquitetura certa é um papel vital no desenvolvimento, onde cada arquitetura separa as funções que cada componente é exercida, como se fosse um empresa, onde é dividido em funções, setores, cargos, para que uma pessoa não fosse responsável por tudo, mas sim que cada pessoa têm sua função específica, onde aumenta a performance da empresa. No desenvolvimento de um software não é diferente, usando uma arquitetura certa, para um respectivo projeto, aumenta a performance do código, reutilização do mesmo, fácil de ter manutenção e capacidade de testar o código, já que vai está separado em funções, será até fácil para achar bugs no sistema.

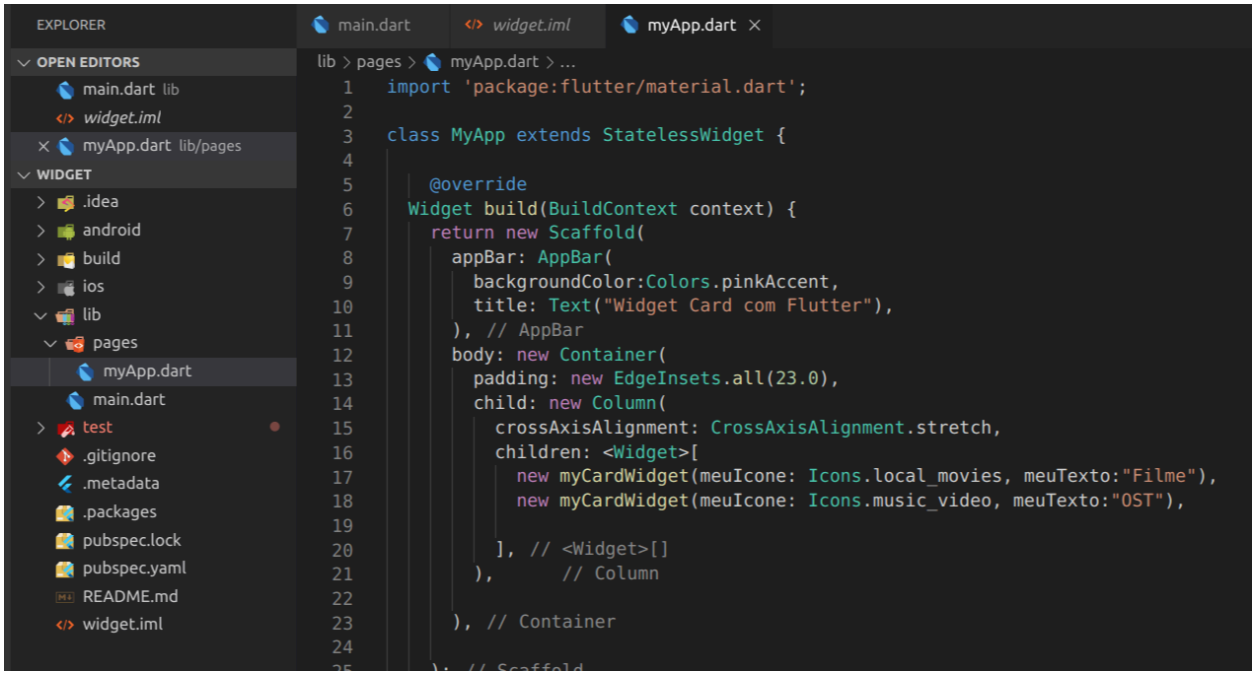
3.2. Ferramentas e Tecnologias Utilizadas

(Neste tópico serão apresentados….)

3.2.1 Desenvolvimento Mobile (Flutter + Dart)

(Começar com um texto e explicar depois a imagens. E uma breve conclusão após as imagens)

Figura 8 - Flutter



Fonte: Medium

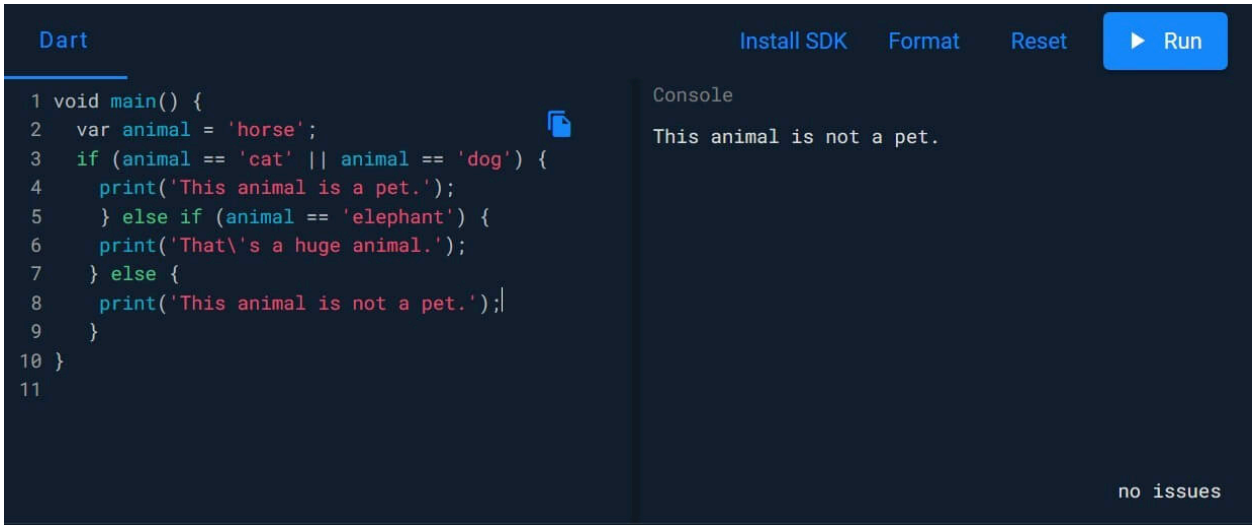
Lançado em 2018, o Flutter é um framework de código aberto criado e mantido pelo Google. Desenvolvedores front-end e full-stack utilizam o Flutter para construir interfaces de usuário (UI) em várias plataformas a partir de uma única base de código. Quando foi lançado, era focado principalmente no desenvolvimento de aplicativos móveis. Hoje, ele suporta seis plataformas: iOS, Android, Web, Windows, macOS e Linux.

Na prática, o Flutter permite que você escreva o código uma única vez e o mesmo aplicativo funcione em diferentes plataformas, como Android e iOS. Seus principais pontos fortes são a versatilidade, uma curva de aprendizado mais suave e maior agilidade no desenvolvimento. Essa ferramenta traz vantagens para as empresas, pois um único desenvolvedor pode criar um aplicativo que funciona em múltiplos sistemas operacionais. Diferente do desenvolvimento nativo, que exige dois desenvolvedores — um para cada plataforma.

Essas são algumas das formas pelas quais o Flutter se destaca como um framework de desenvolvimento multiplataforma:

- **Desempenho próximo ao nativo**: O Flutter utiliza a linguagem Dart, que é compilada diretamente em código de máquina. Isso significa que os dispositivos entendem o código diretamente, resultando em um desempenho rápido e eficiente.
- **Renderização rápida e personalizável**: Ao invés de depender de motores de renderização específicos para cada plataforma, o Flutter utiliza a biblioteca gráfica Skia, também de código aberto e desenvolvida pelo Google. Isso garante uma experiência visual consistente, independentemente da plataforma onde o aplicativo é executado.
- **Ferramentas fáceis para desenvolvedores**: O Flutter foi projetado com foco na simplicidade e na produtividade do desenvolvedor. Ferramentas como a "hot reload" permitem que os desenvolvedores vejam rapidamente as mudanças no código sem precisar reiniciar o aplicativo. Além disso, o inspetor de widgets facilita a análise e a correção de problemas nos layouts da interface de usuário.

Figura 9 - Dart



Fonte: Ionos

Dart é uma linguagem de programação fortemente tipada, criada pela Google em 2011. Inicialmente, a intenção era substituir o JavaScript no desenvolvimento de scripts para páginas web. No entanto, com o tempo e a evolução da linguagem, Dart se tornou uma linguagem multi-paradigma, por mais que tenha fortes características de linguagens orientadas a objetos.

Apesar de não ter alcançado sucesso em sua meta inicial de substituir o JavaScript nos navegadores, o Dart ganhou uma grande relevância com o desenvolvimento e sucesso do Flutter, que se baseia nessa linguagem. Esse avanço atraiu a atenção de muitos desenvolvedores.

O Dart oferece diferentes opções de execução. Ele pode rodar em uma máquina virtual chamada DartVM, parte de um conjunto de ferramentas conhecido como Dart Native. Essa máquina virtual opera em dois modos: JIT (Just-in-Time Compiler) e AOT (Ahead-of-Time Compiler). No modo JIT, a compilação ocorre em tempo de execução, convertendo o código Dart em código de máquina à medida que ele é executado. Já no modo AOT, essa conversão é feita antes da execução.

Outra forma de executar código Dart é através da transpilação para JavaScript, usando a ferramenta dart2js, que faz parte do Dart SDK. Essas várias formas de execução tornam o Dart uma linguagem altamente flexível, capaz de rodar tanto em ambientes nativos, como em aplicativos móveis e de desktop, quanto em ambientes web, como em aplicações que utilizam o Angular, por exemplo.

3.2.2 Desenvolvimento Web (JavaScript + TypeScript + ReactJS)

Figura 10 - JavaScript

Fonte: Medium

O desenvolvimento do sistema de gerenciamento de eventos acadêmicos, utilizou uma série de tecnologias moderna e amplamente utilizadas no mercado. No frontend web foi utilizada TypeScript e Js, no frontend mobile foi utilizada a linguagem Dart e no Backend foi utilizada a linguagem Java, juntamente com os frameworks React js, Flutter e Spring Boot. Que desempenharam papéis fundamentais na construção do sistema como um todo.

JavaScript (JS) é uma das linguagens mais populares hoje, especialmente para construir aplicações web. Criada por Brendan Eich em 1995, sua grande força está na flexibilidade e na compatibilidade com todos osp navegadores modernos. Isso significa que ele permite criar experiências de usuário dinâmicas e interativas diretamente nas páginas web, tornando o desenvolvimento muito mais ágil e acessível. Uma das principais maneiras de utilizá-lo é manipulando o DOM (Document Object Model), o que possibilita modificar o conteúdo da página em tempo real, criando uma experiência mais fluida para o usuário.

De acordo com Flanagan (2020), uma das grandes vantagens do JavaScript é sua capacidade de operar tanto no front-end (no navegador do usuário) quanto no back-end, graças a plataformas como o Node.js. Isso cria uma continuidade no desenvolvimento, permitindo que o mesmo código base funcione em diferentes partes do sistema. Além disso, o JavaScript passou

por grandes atualizações, como o ECMAScript 6 (ES6) e o ECMAScript 8 (ES8), que trouxeram melhorias significativas na sintaxe e no gerenciamento de tarefas assíncronas, tornando o código mais limpo e eficiente.

Frameworks como React.js e Vue.js têm ajudado a fortalecer ainda mais o ecossistema JavaScript. O React.js, criado pelo Facebook, é uma ferramenta poderosa que facilita o desenvolvimento de interfaces de usuário ao usar componentes reutilizáveis. Ele tem sido muito escolhido por desenvolvedores devido à sua simplicidade e ao fato de não impor uma estrutura rígida para o desenvolvimento, o que oferece liberdade na organização do código. Conforme Edwards (2021), isso faz com que o React se destaque em relação ao Vue.js e ao Angular, tornando-o uma escolha popular para a criação de aplicações modernas e escaláveis.

Figura 11 - TypeScript

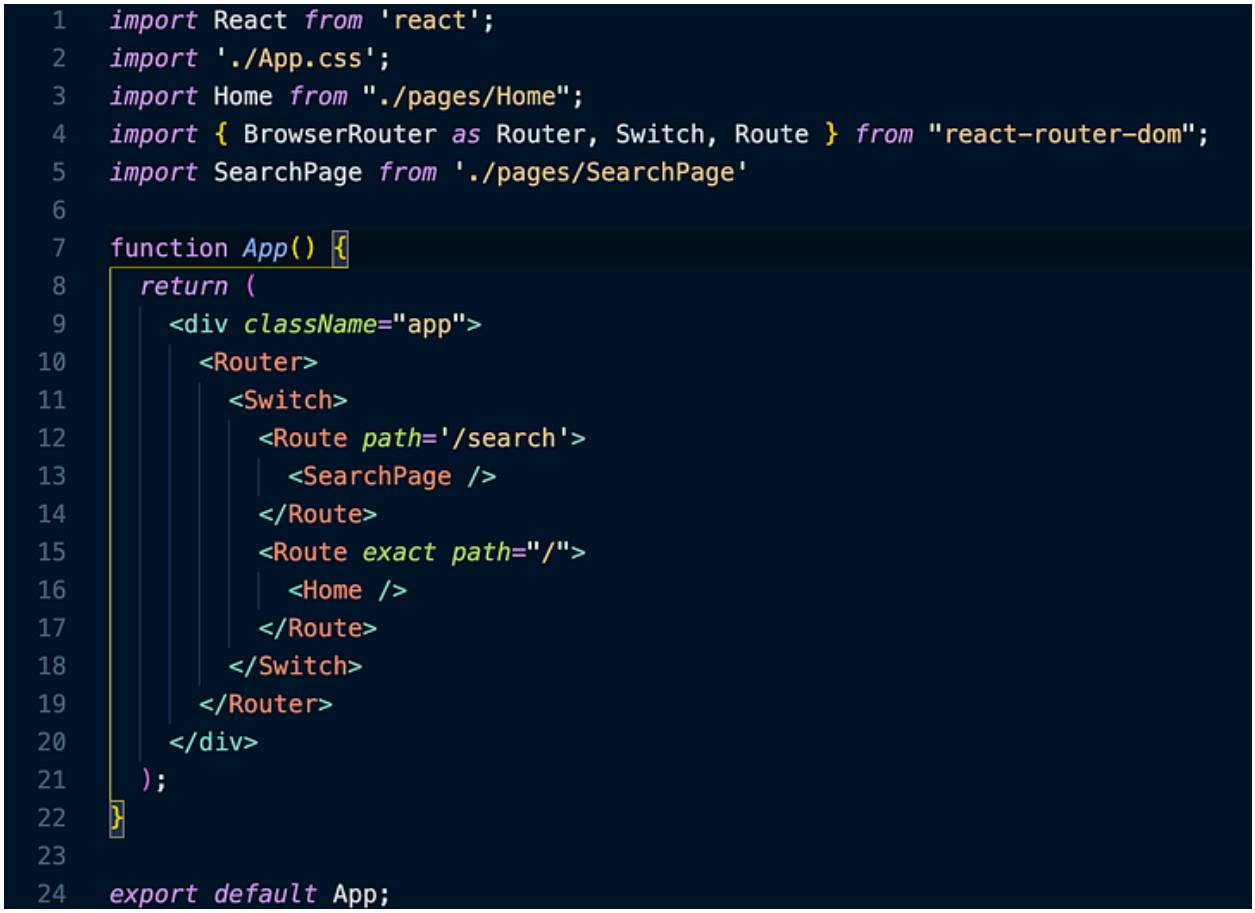
Fonte: Autoria Própria

O TypeScript foi introduzido para melhorar o desenvolvimento em JavaScript ao trazer a tipagem estática para a linguagem, adicionando mais robustez e segurança ao código. Desenvolvido pela Microsoft e lançado em 2012, o TypeScript foi criado por Anders Hejlsberg, o mesmo criador do C#. Ele surgiu como uma solução para alguns dos desafios do JavaScript, especialmente em projetos mais complexos e de grande escala, onde a falta de tipagem pode causar problemas difíceis de detectar. A tipagem estática do TypeScript permite que os erros sejam encontrados durante o desenvolvimento, antes mesmo da execução do código, o que torna o processo de desenvolvimento mais seguro e previsível.

Além disso, o TypeScript é 100% compatível com o JavaScript, o que significa que os desenvolvedores podem começar a adotá-lo de maneira incremental, sem a necessidade de reescrever todo o código existente. Como mencionado por Hejlsberg et al. (2018), essa abordagem é especialmente útil em projetos de médio e grande porte, onde a tipagem estática ajuda a evitar bugs comuns que poderiam passar despercebidos em um código JavaScript puro.

Essa combinação de JavaScript e TypeScript oferece o melhor dos dois mundos: a flexibilidade do JavaScript, que é ideal para prototipagem rápida, junto com a segurança e previsibilidade do TypeScript, que se destaca em projetos maiores e mais complexos. O uso do TypeScript também melhora a produtividade dos desenvolvedores, permitindo que eles se concentrem em criar funcionalidades, enquanto a linguagem os ajuda a evitar erros comuns e a manter o código mais organizado e fácil de manter.

Figura 12 - React Js



```jsx
import React from 'react';
import './App.css';
import Home from "./pages/Home";
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import SearchPage from './pages/SearchPage'

function App() {
  return (
    <div className="app">
      <Router>
        <Switch>
          <Route path='/search'>
            <SearchPage />
          </Route>
          <Route exact path="/">
            <Home />
          </Route>
        </Switch>
      </Router>
    </div>
  );
}

export default App;
```

Fonte: Ichi.pro

O React.js é uma biblioteca JavaScript criada pelo Facebook em 2011, amplamente utilizada para a construção de interfaces de usuário. Ela permite o desenvolvimento de views declarativas e é baseada no conceito de componentes reutilizáveis, o que facilita a criação de aplicações dinâmicas e escaláveis. Como uma ferramenta open source, o React possui uma vasta comunidade de desenvolvedores, o que contribui para seu constante aprimoramento e oferece um ecossistema de bibliotecas e ferramentas que suportam o desenvolvimento eficiente de aplicações.

O React é frequentemente preferido em comparação com outras bibliotecas e frameworks como Vue.js e Angular, principalmente por sua simplicidade e flexibilidade. De acordo com Edwards (2021), o React é amplamente escolhido por desenvolvedores devido à sua curva de aprendizado relativamente rápida e ao fato de ser "unopinionated", ou seja, ele não impõe uma estrutura rígida para o desenvolvimento da aplicação, permitindo que os desenvolvedores tenham mais liberdade para escolher como organizar e estruturar seu código. Além disso, o React se destaca pelo seu suporte robusto a aplicações single-page (SPAs) e pela eficiência de renderização através do uso do Virtual DOM, o que melhora o desempenho, especialmente em aplicações de grande porte.

O sucesso do React também é impulsionado por sua integração fácil com outras bibliotecas e frameworks, bem como pelo suporte à React Native, que permite a criação de aplicações móveis utilizando o mesmo conhecimento em React. Enquanto Vue.js e Angular oferecem boas alternativas, o React continua sendo uma das principais escolhas devido à sua vasta comunidade, suporte corporativo e flexibilidade na construção de interfaces de usuário modernas.

Essas ferramentas e linguagens foram escolhidas para este projeto devido à sua eficiência, flexibilidade e capacidade de se integrar a diferentes partes do desenvolvimento web, resultando em uma aplicação mais rápida, confiável e fácil de manter. Também a velocidade do desenvolvimento de um sistema web usando-as é muito grande.

3.2.3 Desenvolvimento Backend (Java + Spring Boot)

Figura 13 - Java

```java
import java.io.*;
public class Vectores {

    public static void main(String[] args) throws IOException {
        BufferedReader br=new BufferedReader (new InputStreamReader (System.in));
        int sum=0;
        double prom=0;
        int[] Num=new int [6];
        for(int i=1;i<(Num.length);i++){
            System.out.println("ingrese la nota ["+i+"]:");
            Num[i]=Integer.parseInt(br.readLine());
            sum=sum + Num[i];
            prom=sum/5;
        }
        System.out.println("el promedio es:"+prom);



    }
```

Fonte: Twohackerone

Java é uma linguagem de programação criada pela empresa Sun Microsystems em 1995, com o objetivo de permitir que programas possam ser executados em uma ampla variedade de sistemas e dispositivos. A linguagem ganhou o slogan “Write once, run anywhere” (Escreva uma vez, execute em qualquer lugar), porque o código é compilado para um formato intermediário chamado “bytecode”, que pode ser executado em qualquer ambiente que possua a Java Virtual Machine (JVM) instalada.

De acordo com Deitel, “A contribuição mais importante até agora da revolução dos microprocessadores é que ela permitiu o desenvolvimento de computadores pessoais. Os microprocessadores estão tendo um impacto profundo em dispositivos eletrônicos inteligentes de consumo popular. Reconhecendo isso, a Sun Microsystems, em 1991, financiou um projeto de pesquisa corporativa interna chefiado por James Gosling, que resultou em uma linguagem de

programação orientada a objetos chamada Java" (DEITEL, Paul; DEITEL, Harvey. *Java: Como Programar*, 10ª ed. São Paulo: Pearson, 2016, p. 13).

A decisão de usar Java no backend do sistema de gerenciamento de eventos se deu pela linguagem se tratar de uma linguagem robusta, madura, eficiente e segura o que dá confiança e a garantia de um alto nível de escalabilidade e performance, essenciais para lidar com grandes volumes de dados e usuários simultâneos no sistema de eventos.

Figura 14 - SpringBoot

```java
package com.trybe.controller;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/")
public class SaudacaoController {

    @GetMapping
    public ResponseEntity<?> olaMundo() {
        return ResponseEntity.ok().body("Olá Mundo! Estou pronto para aprender Spring Boot!!");
    }

}
```

Fonte: betrybe

O Spring é um framework de desenvolvimento de software para a plataforma Java que visa tornar a programação de Java mais rápida, fácil e segura. Por se tratar de um framework flexível e possuir diversas bibliotecas externas o spring se tornou o mais usado dentre a comunidade de Java.

De acordo com o site da Snyk, "com 4 em cada 10 desenvolvedores usando o Spring Boot em suas aplicações, é interessante observar que ele superou o framework Spring MVC pela primeira vez" (SNYK, 2018) e isso foi um dos fatores principais para a escolha do framework para a aplicação além de sua facilidade de configuração e desenvolvimento rápido, permitindo que os desenvolvedores se concentrem mais na lógica de negócio do que em detalhes técnicos.

3.3. A escolha da arquitetura

A escolha da arquitetura correta se torna cada vez mais importante à medida que a complexidade dos sistemas aumenta, especialmente com os desafios de segurança e a necessidade de seguir padrões que tornam o código mais robusto e padronizado.

3.3.1 Backend

A escolha da arquitetura Package by Feature, que segue os princípios de Hexagonal Architecture, Clean Architecture, MVC e SOLID, para o desenvolvimento do backend do sistema EventsLink, foi motivada pelo desejo de maior modularidade, manutenibilidade e independência entre as camadas. A arquitetura apresenta uma separação clara por features ou módulos, o que aprimora a organização e facilita o progresso contínuo.

De acordo com PHANER (2020), o agrupamento de funcionalidades específicas em módulos independentes possibilita que o código seja mais intuitivo e condizente com as funcionalidades reais do sistema, em vez de ser dividido em camadas técnicas. Isso facilita não somente a escalabilidade, mas também a manutenção, uma vez que cada feature se torna uma unidade autônoma e facilmente gerenciável.

Devido a essa modularização por features, a escalabilidade do projeto se torna mais fácil, uma vez que novas funcionalidades podem ser adicionadas de forma isolada, sem prejudicar outras partes do sistema. A implementação se torna mais ágil e menos propensa a equívocos, uma vez que cada feature ou model incorpora seus próprios detalhes. Dessa forma, o sistema

pode crescer de forma rápida e consistente, atendendo às novas demandas sem comprometer a estabilidade das partes já implementadas.

A modularidade e a independência entre as camadas também aumentam a testabilidade do sistema, permitindo que cada unidade funcional seja testada individualmente, garantindo que o sistema continue robusto e confiável. Isso é particularmente significativo em um sistema como o EventsLink, onde atualizações frequentes e a inclusão de novas funcionalidades são comuns. Além disso, a arquitetura feature-based facilita o desenvolvimento colaborativo, pois diferentes equipes podem trabalhar em features distintas sem interferências, o que favorece o desenvolvimento paralelo e a escalabilidade do projeto.

Em resumo, a arquitetura feature-based foi escolhida como a mais adequada para o projeto **EventsLink** devido à sua flexibilidade e capacidade de lidar com a complexidade crescente do sistema, garantindo manutenibilidade, modularidade e testabilidade ao longo do tempo. Essa abordagem proporciona uma base sólida para o desenvolvimento ágil e sustentável de um sistema que precisa evoluir continuamente sem comprometer a qualidade ou a integridade das funcionalidades já existentes.

3.3.2 Front-end Web

Embora, à primeira vista, seja incomum pensar em uma arquitetura robusta para o frontend, a escolha de uma estrutura de uma arquitetura bem definida é essencial para garantir a escalabilidade, manutenção e organização do projeto. A arquitetura adotada é inspirada nos princípios da Clean Architecture ou Arquitetura em Camadas, tradicionalmente associadas ao backend, mas perfeitamente aplicáveis ao desenvolvimento frontend moderno.

A separação de responsabilidades em diferentes camadas ou diretórios oferece diversos benefícios:

1. Manutenção: A organização clara e modular facilita a identificação e modificação de partes específicas do código, sem afetar outras áreas da aplicação. Isso é crucial em projetos maiores, onde a complexidade do código pode crescer rapidamente.
2. Escalabilidade: A arquitetura modular permite que o projeto seja facilmente escalável. Novas funcionalidades podem ser adicionadas em camadas específicas sem a necessidade de reestruturar todo o sistema.
3. Reutilização de Código: Componentes bem definidos e separados em camadas permitem a reutilização de código em diferentes partes da aplicação, reduzindo a duplicação e melhorando a consistência.
4. Testabilidade: Ao isolar a lógica de negócios, os componentes e as integrações com o backend em diferentes camadas, é mais fácil escrever testes unitários e de integração, garantindo a qualidade e confiabilidade do software.

3.3.3 Estrutura do Projeto

De acordo com o FREECODECAMP (2022) a estrutura do projeto é dividida na respectivas camadas:

- Adapters: Responsáveis por adaptar a comunicação entre a camada de apresentação e as fontes de dados, como APIs externas, garantindo que a aplicação esteja desacoplada de detalhes específicos de implementação.
- Contexts: Utilizados para gerenciar estados globais da aplicação, como autenticação e configurações do usuário, proporcionando uma maneira eficiente de compartilhar dados entre diferentes componentes sem a necessidade de passar props manualmente.
- Components: Contém os componentes reutilizáveis da interface, focando na criação de elementos modulares e independentes. Isso permite que a aplicação seja construída de maneira mais coesa e previsível, promovendo a reutilização de código.
- Styles: Armazena os estilos globais e específicos de componentes, garantindo uma abordagem centralizada e consistente para a estilização da aplicação, facilitando a manutenção de uma identidade visual unificada.

- Pages: Organiza os componentes responsáveis por renderizar as páginas da aplicação, geralmente associados às rotas. Cada página é composta por diversos componentes menores, agrupando a lógica de exibição específica de cada seção da aplicação.

3.3.4 Mobile

A escolha da arquitetura MVVM (Model-View-ViewModel) para o desenvolvimento do projeto mobile foi motivada pela sua capacidade de promover uma separação entre a interface do usuário e a lógica de negócios, o que favorece a manutenção do código. De acordo com diversos estudos e práticas de mercado, o MVVM estrutura a aplicação em três componentes principais: o Model, o View e o ViewModel, cada um com responsabilidades bem definidas.

O **Model** é responsável pela gestão dos dados e das regras de negócio da aplicação, sendo a camada que lida diretamente com o armazenamento e manipulação de informações. O **View** trata exclusivamente da interface do usuário, sendo a responsável por exibir os dados de forma acessível e visualmente adequada. Já o **ViewModel** faz a mediação entre o Model e o View, processando os dados e eventos da interface, garantindo que o View possa reagir às mudanças de forma eficiente e desacoplada da lógica de negócios.

Essa separação de responsabilidades contribui diretamente para a modularidade do projeto, com diferentes equipes ou desenvolvedores trabalhando simultaneamente em camadas distintas, sem interferências. Além disso, o MVVM facilita a testabilidade do código, já que a lógica de negócios e a interface estão bem separadas, tornando mais simples a criação de testes unitários e de integração.

Outro fator relevante na escolha do MVVM foi sua compatibilidade com frameworks modernos, como o Flutter, amplamente utilizado no desenvolvimento mobile. O Flutter, em conjunto com o MVVM, oferece ferramentas e suporte robustos para a implementação dessa arquitetura, o que torna o processo de desenvolvimento mais ágil e eficiente.

Por fim, a arquitetura MVVM foi considerada a mais adequada para o projeto, uma vez que ela equilibra a simplicidade na implementação para um sistema escalável e de fácil manutenção, que são características essenciais para garantir a evolução do projeto ao longo do tempo sem comprometer a qualidade do software.

3.4. Banco de dados

A escolha do PostgreSQL para o desenvolvimento do sistema de gerenciamento de eventos **EventsLink** se baseia em diversos fatores técnicos. Primeiramente, por ser uma plataforma de código aberto, oferece um excelente custo-benefício, eliminando a necessidade de licenças comerciais, o que é particularmente vantajoso em ambientes acadêmicos com restrições orçamentárias.

Além disso, o **EventsLink** precisa lidar com um grande volume de dados, como informações sobre palestrantes, participantes, horários de sessões, locais e feedbacks. A estrutura relacional do PostgreSQL, com suporte a relações complexas entre tabelas (como 1:1, 1:N e N:N), torna-o ideal para este tipo de aplicação. Essas relações garantem a integridade referencial dos dados, facilitando a organização e o acesso eficiente às informações.

Além das características mencionadas, o PostgreSQL facilita a normalização dos dados, um processo fundamental em bancos de dados relacionais. A normalização organiza as tabelas de maneira eficiente, minimizando redundâncias e garantindo a integridade dos dados. Este processo é particularmente importante no sistema **EventsLink**, onde a relação entre várias entidades, como palestrantes, participantes e eventos, precisa ser mantida de forma consistente e clara. A normalização não apenas melhora o desempenho do banco de dados, mas também simplifica a manutenção e a escalabilidade do sistema ao longo do tempo.

Outro ponto crucial é a capacidade de extensibilidade do PostgreSQL, permitindo personalizações e adequações específicas ao contexto acadêmico. Isso possibilita a criação de

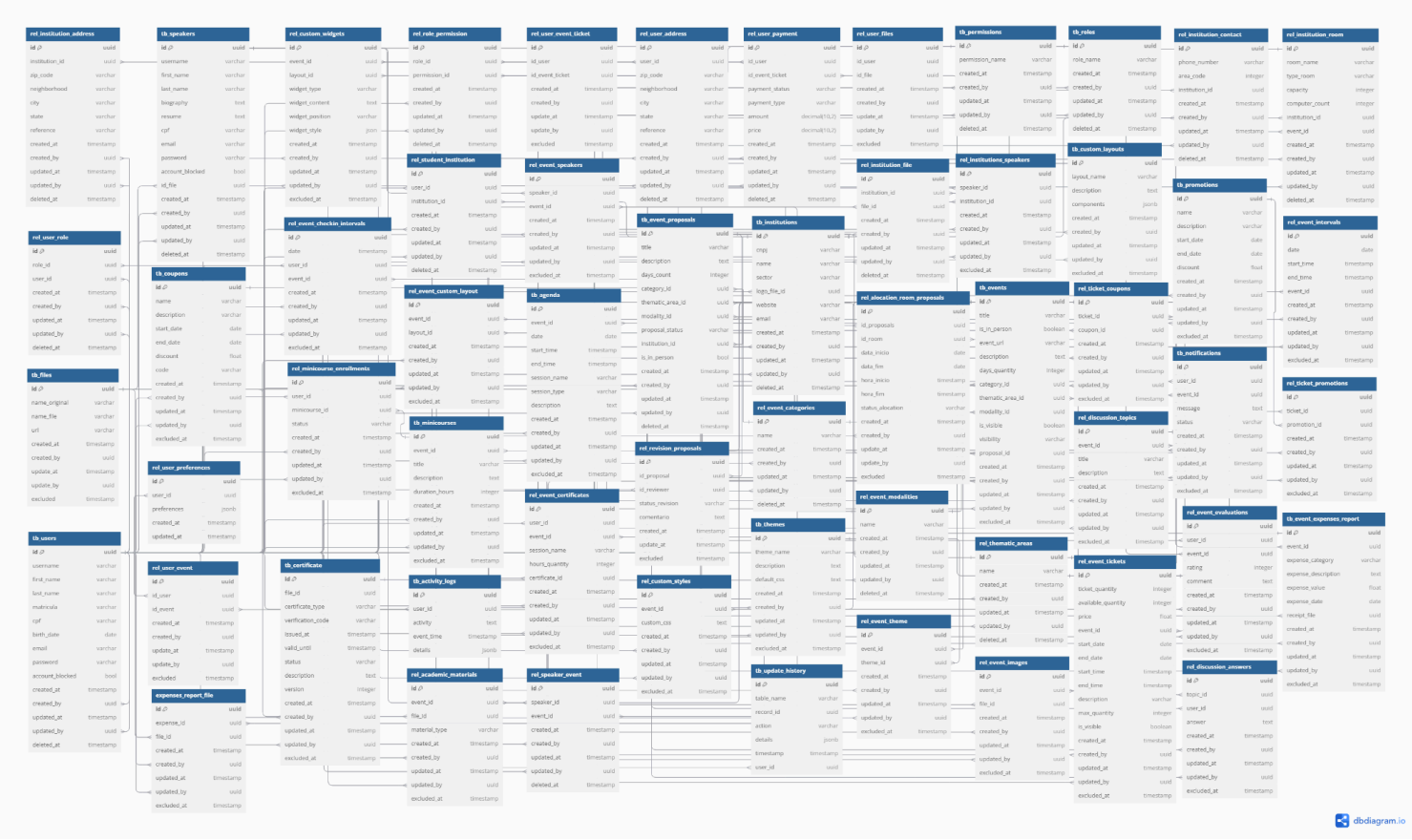
tipos de dados personalizados para armazenar informações específicas, como notas de avaliação de palestras ou categorias de eventos. Além disso, o PostgreSQL suporta consultas complexas, o que é fundamental para gerar relatórios detalhados e análises precisas.

3.5. Modelo de banco de dados

O modelo de Banco de Dados (BD) utilizado foi o modelo relacional no qual é composto por entidade(tabelas) e relações, em que cada linha é composta por campos e cada um é identificado por um nome (HEUSER, 2009)

UDF
Centro
Universitário

Figura 15 - Modelagem do banco



Fonte: Autoria Própria

Dicionário de dados referente a Figura 15

Tabela 1 - Dados Genéricos

Nome Coluna	Tipo	Propriedades	Descrição
id	uuid	PK	Identificador da entidade
created_at	timestamp	not null	Data e hora da entidade.
created_by	uuid	FK, null	FK do usuário que a criou.
updated_at	timestamp	null	Dia e hora da atualização da entidade.
updated_by	uuid	FK, null	FK do usuário que a atualizou a entidade.
deleted_at	timestamp	null	Dia e hora da remoção da entidade

A tabela 1, é um modelo genérico, diante os mesmo atributos apresentados nesta tabela, estão presentes nas tabelas posteriormente.

Tabela 2 - tb_users

Nome Coluna	Tipo	Propriedades	Descrição
username	varchar	not null	Username do usuário, entrada opcional
first_name	varchar	not null	Primeiro nome do usuário
last_name	varchar	not null	Último nome do usuário

matricula	varchar	not null	Matrícula do usuário, que pode ser o RGM ou a própria matrícula do professor ou etc.
cpf	varchar	not null	CPF do usuário
birth_date	date	not null	Data de nascimento
email	varchar	not null	Email do usuário, entrada principal
password	varchar	not null	Senha do usuário
account_blocked	boolean	null	Status da conta, para verificar se está bloqueada.

Como apresentado na Tabela 2, armazena informações sobre os usuários do sistema, que podem ser participantes, palestrantes ou administradores. Cada usuário tem um papel associado (administrador, palestrante, participante) que define seus direitos e permissões no sistema.

Tabela 3 - tb_files

Nome Coluna	Tipo	Propriedades	Descrição
name_original	varchar	not null	Nome original do arquivo
name_file	varchar	not null	Nome atual do arquivo
url	varchar	not null	URL onde se encontra o arquivo

Como apresentado na Tabela 3, armazena os dados dos arquivos submetidos na plataforma, com os atributos presentes na tabela.

Tabela 4 - rel_user_event_ticket

Nome Coluna	Tipo	Propriedades	Descrição
id_user	uuid	FK, not null	ID do usuário
id_event_ticket	uuid	FK, not null	Id do ingresso do evento

Como apresentado na Tabela 4, é armazenado os dados dos usuários e os ingressos, que o usuário adquiriu, comprando ou de graça.

Tabela 5 - rel_user_payment

Nome Coluna	Tipo	Propriedades	Descrição
id_user	uuid	FK, not null	Id do usuário
id_event_ticket	uuid	FK, not null	Id dos ingressos do usuário
payment_status	varchar	not null	Status do pagamento (pago, pendente, cancelado)
payment_type	varchar	not null	O tipo de pagamento (PIX, Boleto ou Cartão de crédito)
amount	float	not null	
price	float	not null	O preço do pagamento

Como apresentado na Tabela 5, é armazenado os dados do pagamento do usuário, juntamente com o retorno do pagamento, atualizando o status do pagamento, com as possibilidades de tipos de pagamento inserido na plataforma, como PIX, Boleto ou Cartão de Crédito.

Tabela 6 - rel_user_files

Nome Coluna	Tipo	Propriedades	Descrição
id_user	uuid	FK, not null	Id do usuário
id_file	uuid	FK, not null	Id do arquivo

Como apresentado na Tabela 6, é armazenado quais arquivos pertencem ao usuário.

Tabela 7 - tb_permissions

Nome Coluna	Tipo	Propriedades	Descrição
permission_name	varchar	FK, not null	Nome da permissão

Como apresentado na Tabela 7, é apresentado uma tabela, que irá armazenar os nomes das permissões, que será atribuído a respectivos usuários.

Tabela 8 - tb_roles

Nome Coluna	Tipo	Propriedades	Descrição
role_name	Nome da role	not null	Nome da regra

Como apresentado na Tabela 8, é apresentado uma tabela, que irá armazenar os nomes das regras, que serão atribuídos a respectivos usuários.

Tabela 9 - rel_user_role

Nome Coluna	Tipo	Propriedades	Descrição
role_id	uuid	FK, not null	Id da regra
user_id	uuid	FK, not null	Id do usuário atribuído

Como apresentado na Tabela 9, será a integração das regras aos respectivos usuários.

Tabela 10 - rel_role_permission

Nome Coluna	Tipo	Propriedades	Descrição
role_id	uuid	FK, not null	Id da regra
permission_id	uuid	FK, not null	Id da permissão

Como apresentado na Tabela 10, será a integração das permissões aos respectivos usuários.

Tabela 11 - tb_user_address

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário
zip_code	varchar	not null	Cep do usuário
neighborhood	varchar	not null	Bairro do usuário
city	varchar	not null	Cidade do usuário
state	varchar	not null	Estado do usuário
reference	varchar	null	Referência do usuário

Como apresentado na Tabela 11, será o armazenamento dos endereços dos usuários.

Tabela 12 - tb_instituitions

Nome Coluna	Tipo	Propriedades	Descrição
cnpj	varchar	not null	Cnpj da instituição
name	varchar	not null	Nome da instituição
sector	varchar	not null	Setor da instituição

logo_file_id	uuid	FK, not null	Id do arquivo de logo da instituição
website	varchar	null	Endereço online da instituição
email	varchar	not null	Email da instituição

Como apresentado na Tabela 12, representa as instituições que os alunos serão associados à instituição, onde está matriculado.

Tabela 13 - rel_institution_file

Nome Coluna	Tipo	Propriedades	Descrição
institution_id	uuid	FK, not null	Id da instituição
file_id	uuid	FK, not null	Id do arquivo

Como apresentado na Tabela 13, relaciona arquivos com instituições, como logotipos ou documentos.

Tabela 14 - tb_institution_room

Nome Coluna	Tipo	Propriedades	Descrição
room_name	varchar	not null	O nome da sala, por exemplo, sala 101
type_room	varchar	not null	Tipo da sala para identificação, se for sala ou auditório
capacity	integer	not null	Capacidade máxima suportada pelo local
computer_count	integer	not null	Número de

			computadores
institution_id	uuid	FK, not null	Id da instituição
event_id	uuid	FK, null	Id do Evento

Como apresentado na Tabela 14, registra salas dentro de instituições, associadas a eventos, incluindo capacidade e número de computadores.

Tabela 15 - tb_institution_contact

Nome Coluna	Tipo	Propriedades	Descrição
phone_number	varchar	not null	Número do telefone
area_code	varchar	not null	Código de área
institution_id	uuid	FK, not null	Id da instituição

Como apresentado na Tabela 15, armazena informações de contato das instituições, como números de telefone e DDD.

Tabela 16 - tb_institution_address

Nome Coluna	Tipo	Propriedades	Descrição
intitution_id	uuid	FK, not null	Id do instituição
zip_code	varchar	not null	Cep do instituição
neighborhood	varchar	not null	Bairro do instituição
city	varchar	not null	Cidade do instituição
state	varchar	not null	Estado do instituição
reference	varchar	not null	Referência do instituição

Como apresentado na Tabela 16, armazena endereços das instituições, incluindo CEP, bairro, cidade e estado.

Tabela 17 - rel_student_institution

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário
intitution_id	uuid	FK, not null	Id do instituição

Como apresentado na Tabela 17, relaciona usuários que são estudantes com instituições, mantendo o vínculo entre eles.

Tabela 18 - tb_event_proposals

Nome Coluna	Tipo	Propriedades	Descrição
title	varchar	not null	Titulo do evento
description	varchar	not null	Descrição do evento
days_count	integer	not null	Quantidade de dias do evento
category_id	uuid	FK, not null	Categoria do evento
thematic_area_id	uuid	FK, not null	Área temática do evento
modality_id	uuid	FK, not null	Modalidade do evento
proposal_status	varchar	not null	Status da proposta (Aprovado, Rejeitado ou Pendente)
institution_id	uuid	FK, not null	Id da instituição
is_in_person	bool	not null	Se o evento é presencial ou não

Como apresentado na Tabela 18, armazena propostas de eventos, incluindo título, descrição e status da proposta (pendente, aprovado, rejeitado).

Tabela 19 - tb_revision_proposals

Nome Coluna	Tipo	Propriedades	Descrição
id_proposal	uuid	FK, not null	Id da proposta para uma criação de um evento
id_reviewer	uuid	FK, not null	Id do avaliador da proposta
status_revision	varchar	not null	Status da revisão da proposta (Aprovado ou Rejeitado)
comentário	text	not null	Comentário sobre o porquê da revisão

Como apresentado na Tabela 19, registra as revisões de propostas de eventos feitas por usuários, incluindo comentários e status da revisão.

Tabela 20 - tb_alocation_room_proposals

Nome Coluna	Tipo	Propriedades	Descrição
id_proposals	uuid	FK, not null	Id da proposta da criação do evento
id_room	uuid	FK, not null	Id do local que deseja reservar (Sala ou Auditório)
data_inicio	Date	not null	O dia inicial que deseja reservar a sala
data_fim	Date	not null	O dia final da reserva da sala

hora_inicio	Date	not null	A hora inicial da alocação da sala
hora_fim	Date	not null	A hora final da alocação da sala
status_alocation	varchar	not null	Status da alocação (Reservada ou Aprovada)

Como apresentado na Tabela 20, relaciona propostas de eventos com alocações de salas, registrando o horário e status da alocação.

Tabela 21 - rel_user_event

Nome Coluna	Tipo	Propriedades	Descrição
id_user	uuid	FK, not null	Id do usuário
id_event	uuid	FK, not null	Id do Evento

Como apresentado na Tabela 21, relaciona usuários com eventos, mantendo informações de inscrição e participação.

Tabela 22 - rel_institutions_speakers

Nome Coluna	Tipo	Propriedades	Descrição
speaker_id	uuid	FK, not null	Id do palestrante
institution_id	uuid	FK, not null	Id da instituição

Como apresentado na Tabela 22, relaciona palestrantes com instituições, mantendo o vínculo entre eles.

Tabela 23 - tb_speakers

Nome Coluna	Tipo	Propriedades	Descrição
username	varchar	not null	Username do apresentador, entrada opcional
first_name	varchar	not null	Primeiro nome do apresentador
last_name	varchar	not null	Último nome do apresentador
biography	varchar	null	Biografia do apresentador
resume	varchar	null	Resume do usuário
cpf	varchar	not null	CPF do apresentador
email	varchar	not null	Email do apresentador, entrada principal
password	varchar	not null	Senha do apresentador
account_blocked	boolean	null	Status da conta, para verificar se está bloqueada.
id_file	uuid	FK, not null	Id do arquivo do apresentador

Como apresentado na Tabela 23, contém informações sobre palestrantes, incluindo biografia, resumo e CPF.

Tabela 24 - rel_speaker_event

Nome Coluna	Tipo	Propriedades	Descrição
speaker_id	uuid	FK, not null	Id do apresentador
event_id	uuid	FK, not null	Id do evento

Como apresentado na Tabela 24, relaciona palestrantes com eventos em que eles participam.

Tabela 25 - tb_event_certificates

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário
event_id	uuid	FK, not null	Id do evento
session_name	varchar	null	Nome da sessão
hours_quantify	integer	null	Quantidade de horas
certificate_id	uuid	FK, not null	Id do certificado

Como apresentado na Tabela 25, armazena certificados emitidos para usuários que participam de eventos, incluindo informações sobre as sessões e horas.

Tabela 26 - tb_certificate

Nome Coluna	Tipo	Propriedades	Descrição
file_id	uuid	FK, not null	Id do arquivo do certificado
certificate_type	varchar	not null	Tipo do certificado (Participação, Conclusão)

verification_code	varchar	not null	Código de verificação
issued_at	timestamp	not null	Data de emissão
valid_until	timestamp	null	Validade do certificado
status	varchar	not null	Status do certificado (Cancelado, Concluído)
description	text	not null	Descrição do certificado
version	integer	not null	Versão do certificado

Como apresentado na Tabela 26, armazena certificados, incluindo código de verificação e datas de emissão e validade.

Tabela 27 - tb_event_categories

Nome Coluna	Tipo	Propriedades	Descrição
name	varchar	not null	Nome da categoria

Como apresentado na Tabela 27, armazena categorias de eventos, como workshops, conferências, etc.

Tabela 28 - tb_event_modalities

Nome Coluna	Tipo	Propriedades	Descrição
name	varchar	not null	Nome da modalidade

Como apresentado na Tabela 28, armazena modalidades de eventos, como presencial ou online.

Tabela 29 - tb_thematic_areas

Nome Coluna	Tipo	Propriedades	Descrição
name	varchar	not null	Nome da área temática

Como apresentado na Tabela 29, armazena áreas temáticas relacionadas a eventos, como tecnologia, saúde, etc.

Tabela 30 - tb_custom_layouts

Nome Coluna	Tipo	Propriedades	Descrição
layout_name	varchar	not null	Nome do layout
description	text	not null	Descrição do layout
components	jsonb	not null	Components do layout em json

Como apresentado na Tabela 30, armazena layouts personalizados de eventos, incluindo nome, descrição e componentes visuais.

Tabela 31 - tb_custom_widgets

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
layout_id	uuid	FK, not null	Id do layout
widget_type	varchar	not null	O tipo de widget
widget_content	text	not null	O conteúdo do widget
widget_position	varchar	not null	A posição do widget

widget_style	json	not null	O stilo do widget

Como apresentado na Tabela 31, relaciona widgets personalizados com eventos e layouts, definindo tipo, conteúdo e estilo do widget.

Tabela 32 - rel_event_custom_layout

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do Evento
layout_id	uuid	FK, not null	Id do layout

Como apresentado na Tabela 32, relaciona layouts personalizados com eventos, aplicando um layout específico a um evento.

Tabela 33 - tb_custom_styles

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
custom_css	text	not null	CSS customizado

Como apresentado na Tabela 33, armazena estilos personalizados de eventos, incluindo CSS customizado.

Tabela 34 - tb_themes

Nome Coluna	Tipo	Propriedades	Descrição
theme_name	varchar	not null	Nome do thema
description	text	not null	Descrição do tema
default_css	text	not null	CSS default

Como apresentado na Tabela 34, armazena temas aplicáveis aos eventos, incluindo descrições e CSS padrão.

Tabela 35 - rel_event_theme

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
theme_id	uuid	FK, not null	Id do tema

Como apresentado na Tabela 35, relaciona temas com eventos, aplicando um tema específico a um evento.

Tabela 36 - tb_events

Nome Coluna	Tipo	Propriedades	Descrição
title	varchar	not null	Titulo do evento
is_in_person	bool	not null	Se o evento for presencialmente
event_url	varchar	null	A url do evento, se o evento for presencialmente
description	text	not null	A descrição do evento
days_quantity	integer	not null	Quantidade de dias do evento
category_id	uuid	FK, not null	Id da categoria do evento
thematic_area_id	uuid	FK, not null	Id da área temática do evento

modality_id	uuid	FK, not null	Id da modalidade do evento
is_visible	bool	not null	Se o evento foi liberado
visibility	varchar	not null	A visibilidade do evento, se é particular ou público
proposal_id	uuid	FK, not null	Id da proposta

Como apresentado na Tabela 36, armazena eventos, incluindo título, descrição, URL, modalidade, visibilidade e associação com propostas.

Tabela 37 - tb_event_intervals

Nome Coluna	Tipo	Propriedades	Descrição
date	date	not null	Data do evento
start_time	timestamp	not null	Hora que começa
end_time	timestamp	not null	Hora que termina
event_id	uuid	FK, not null	Id do evento

Como apresentado na Tabela 37, registra intervalos de tempo em que um evento ocorre, incluindo datas e horários de início e fim.

Tabela 38 - rel_event_checkin_intervals

Nome Coluna	Tipo	Propriedades	Descrição
date	timestamp	not null	Pegar o timestamp, dia, hora, segundo e minuto, do checkin

user_id	uuid	FK, not null	Id do usuário
event_id	uuid	FK, not null	Id do evento

Como apresentado na Tabela 38, registra check-ins de usuários em eventos, associando datas e horários de entrada.

Tabela 39 - rel_event_speakers

Nome Coluna	Tipo	Propriedades	Descrição
speaker_id	uuid	FK, not null	Id do palestrante
event_id	uuid	FK, not null	Id do evento

Como apresentado na Tabela 39, relaciona palestrantes com eventos.

Tabela 40 - rel_event_images

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
file_id	uuid	FK, not null	Id do arquivo

Como apresentado na Tabela 40, relaciona imagens com eventos, como fotos de divulgação ou documentos visuais.

Tabela 41 - tb_event_tickets

Nome Coluna	Tipo	Propriedades	Descrição
ticket_quantity	integer	not null	Quantidade total de

			ingressos
available_quantity	integer	not null	Quantidade restante de ingressos
price	float	not null	Preço do ingresso
event_id	uuid	FK, not null	Id do evento
start_date	date	not null	Data de inicio
end_date	date	not null	Data final
start_time	timestamp	not null	Hora inicial
end_time	timestamp	not null	Hora final
description	varchar	null	Descrição do ingresso
max_quantity	integer	null	Máximo de ingresso permitido por compra
is_visible	bool	not null	Se o ingresso já pode ser visível

Como apresentado na Tabela 41, armazena informações sobre ingressos de eventos, como quantidade disponível, preço e datas de início e término de venda.

Tabela 42 - rel_ticket_promotions

Nome Coluna	Tipo	Propriedades	Descrição
ticket_id	uuid	FK, not null	Id do ingresso
promotion_id	uuid	FK, not null	Id da promoção

Como apresentado na Tabela 42, relaciona ingressos de eventos com promoções aplicáveis.

Tabela 43 - rel_ticket_coupons

Nome Coluna	Tipo	Propriedades	Descrição
ticket_id	uuid	FK, not null	Id do ingresso
coupon_id	uuid	FK, not null	Id do cupom

Como apresentado na Tabela 43, relaciona ingressos de eventos com cupons de desconto aplicáveis.

Tabela 44 - tb_promotions

Nome Coluna	Tipo	Propriedades	Descrição
name	varchar	not null	Nome da promoção
description	varchar	null	Descrição da promoção
start_date	date	not null	Data inicial da promoção
end_date	date	not null	Data final da promoção
discount	float	not null	Desconto da promoção

Como apresentado na Tabela 44, armazena promoções, incluindo nome, descrição, datas e descontos.

Tabela 45 - tb_coupons

Nome Coluna	Tipo	Propriedades	Descrição
name	varchar	not null	Nome do cupom
description	varchar	null	Descrição do cupom

start_date	date	not null	Data inicial do cupom
end_date	date	not null	Data final do cupom
discount	float	not null	Disconto do cupom
code	varchar	not null	Código do cupom

Como apresentado na Tabela 45, armazena cupons de desconto, incluindo código, descrição e datas de validade.

Tabela 46 - tb_schedule

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
date	date	not null	Data da agenda
start_time	timestamp	not null	Hora inicial da agenda
end_time	timestamp	not null	Hora final da agenda
session_name	varchar	not null	Nome da sessão
session_type	varchar	not null	Tipo da sessão
description	text	not null	Descrição da agenda

Como apresentado na Tabela 46, armazena a programação de eventos, incluindo sessões, horários e descrições das atividades.

Tabela 47 - rel_academic_materials

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
file_id	uuid	FK, not null	Id do arquivo

material_type	varchar	not null	Tipo do material

Como apresentado na Tabela 47, relaciona eventos com materiais acadêmicos disponibilizados, como slides ou artigos.

Tabela 48 - rel_event_evaluations

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário
event_id	uuid	FK, not null	Id do evento
ratting	integer	not null	Quantidade de estrelas, máximo 5
comment	text	null	Comentário do evento

Como apresentado na Tabela 48, armazena avaliações de eventos feitas pelos usuários, incluindo notas e comentários.

Tabela 49 - tb_discussion_topics

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
title	varchar	not null	Título do tópico de discussão
description	text	null	Descrição do tópico da discussão

Como apresentado na Tabela 49, armazena tópicos de discussão em eventos, incluindo título e descrição.

Tabela 50 - rel_discussion_answers

Nome Coluna	Tipo	Propriedades	Descrição
topic_id	uuid	FK, not null	Id tópico da discussão
user_id	uuid	FK, not null	Id do usuário
answer	text	not null	Resposta da discussão

Como apresentado na Tabela 50, armazena respostas dos usuários a tópicos de discussão.

Tabela 51 - tb_notifications

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usúario
event_id	uuid	FK, not null	Id do evento
message	text	not null	Mensagem da notificação
status	varchar	not null	Status da notificação

Como apresentado na Tabela 51, armazena notificações enviadas aos usuários sobre eventos, incluindo mensagens e status.

Tabela 52 - rel_minicourse_enrollments

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário
minicourse_id	uuid	FK, not null	Id do mini curso
status	varchar	not null	Status do mini curso

Como apresentado na Tabela 52, registra a inscrição de usuários em minicursos de eventos, incluindo status de inscrição.

Tabela 53 - tb_minicourses

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Id do evento
title	varchar	not null	Título do mini curso
description	text	null	Descrição do mini curso
duration_hours	integer	not null	Duração do mini curso em horas

Como apresentado na Tabela 53, armazena minicursos relacionados a eventos, incluindo título, descrição e duração em horas.

Tabela 54 - tb_activity_logs

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário
activity	text	not null	atividade
event_time	timestamp	not null	Tempo de atividade
details	details	null	Detalhe da atividade

Como apresentado na Tabela 54, armazena as tarefas do usuário dentro do sistema.

Tabela 55 - tb_user_preferences

Nome Coluna	Tipo	Propriedades	Descrição
user_id	uuid	FK, not null	Id do usuário

preferences	jsonb	not null	Preferências do usuário

Como apresentado na Tabela 55, registra atualizações de preferências do usuário no sistema.

Tabela 56 - tb_event_expenses_report

Nome Coluna	Tipo	Propriedades	Descrição
event_id	uuid	FK, not null	Referência ao evento relacionado
expense_category	varchar	not null	Categoria da despesa
expense_description	text	not null	Descrição detalhada da despesas
expense_value	float	not null	Valor da despesas
expense_date	date	not null	Data em a despesa ocorreu
receipt_file	uuid	FK, not null	Id do arquivo de recibo

Como apresentado na Tabela 56, registrar o relatório de despesas relacionadas do evento em específico, como valor, descrição e o arquivo de recibo.

Tabela 57 - rel_expenses_report_file

Nome Coluna	Tipo	Propriedades	Descrição
expense_id	uuid	FK, not null	Referência a despesa
file_id	uuid	FK, not null	Referência ao evento arquivo do relatório

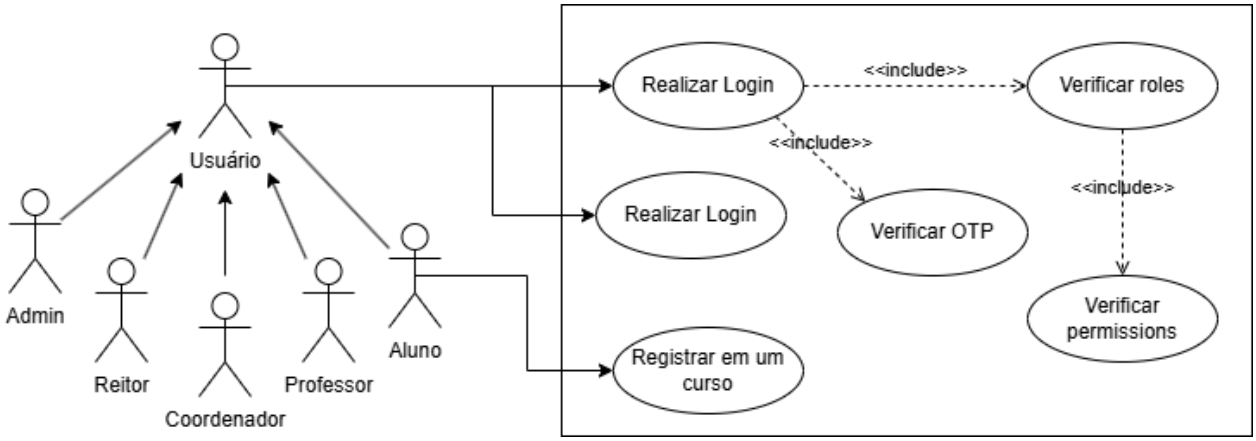
Como apresentado na Tabela 57, vincula relatórios de despesas a arquivos como recibos.

3.6. Diagramas

Para este projeto foram utilizados 3 diagramas: diagrama de caso de uso, diagrama de sequência e diagrama de classes.

3.6.1 Caso de Uso

Figura 16 - Diagrama de Caso de uso Login



Fonte: Autoria Própria

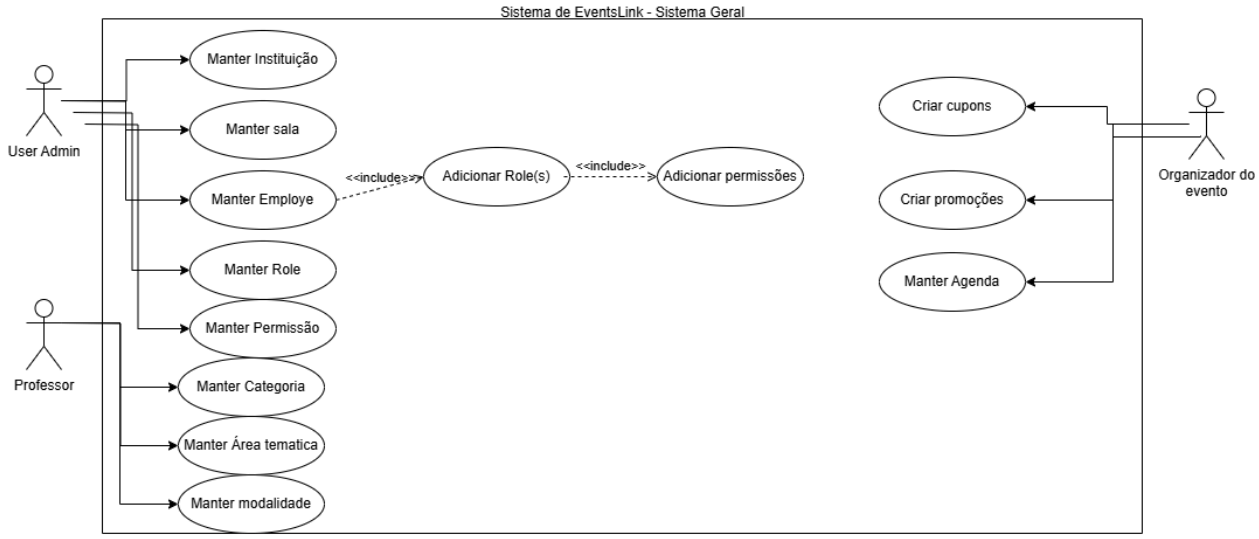
Tabela 58 - Descrição de caso de uso Efetuar login

Nome do caso de uso	UC01 - Efetuar login.
Descrição	Usuário acessa o sistema pelo navegador ou pelo app e informar o usuário que pode ser o username ou email e a senha.
Ator	Admin, Palestrante, Professor, Avaliador e Aluno
Pré-condições	O sistema deve estar hospedado no servidor web. O usuário deve estar cadastrado.
Fluxo de principal	Usuário preenche os campos solicitados na página/tela do login. Sistema valida os dados. Sistema valida as roles e permissions.

	Sistema envia o código OTP. Usuário preenche o campo OTP. O sistema verifica o OTP. Sistema direciona o usuário para a página inicial ou dashboard.
Fluxo alternativo	Usuário ou senha inválidos(s). Alerta com mensagem é mostrada. Usuário esqueceu a senha redefinir senha Usuário sem autorização.
Pós-condições	Usuário entra conectado ao sistema

Fonte: Autoria Própria

Figura 17 - Diagrama de Caso de uso Sistema



Fonte: Autoria Própria

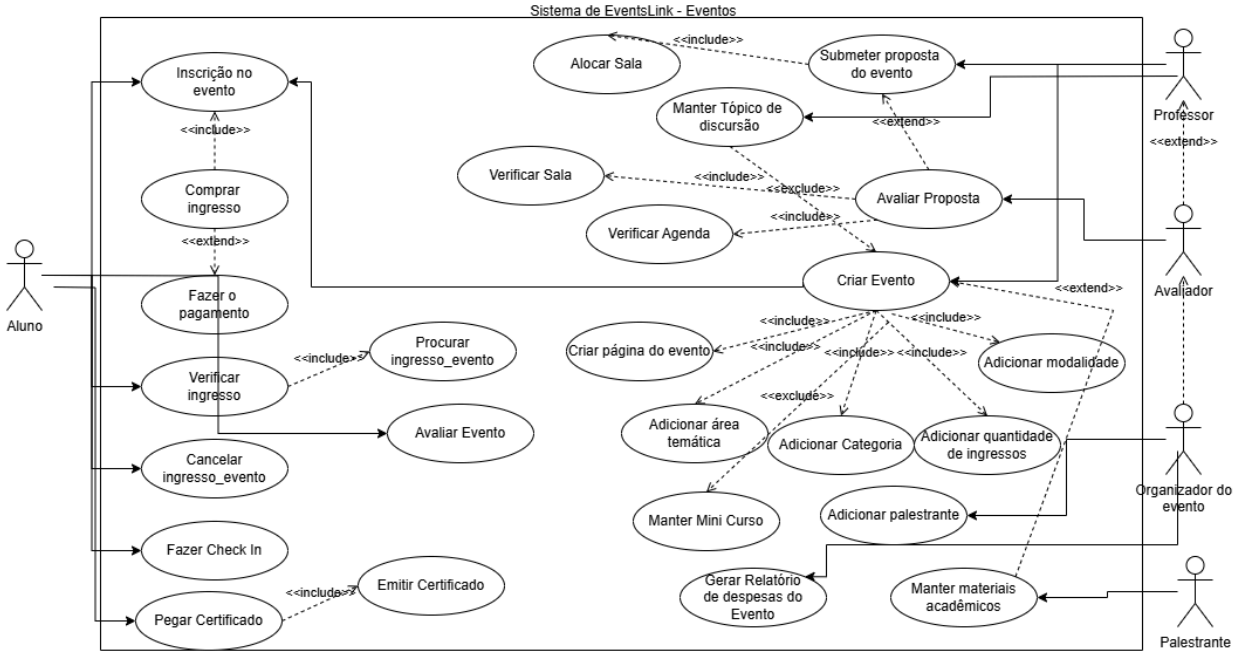
Tabela 59 - Descrição de caso de uso Efetuar cadastro

Nome do caso de uso	UC02 - Cadastro
Descrição	Admin adiciona a sala, adiciona o funcionário (Professor e

	Organizador do evento e Avaliador). Professor cria uma categoria, área temática e modalidade. Organizador do evento cria cupons e promoções do respectivo evento.
Ator	Admin, Professor, Avaliador
Pré-condições	Admin, Professor e Avaliador precisam estar logados no sistema.
Fluxo de principal	Admin informa a descrição da sala, após a análise se não estiver criada, é criada a sala. Adicionar uma instituição. Admin cria as roles e as permissões. Admin informa a descrição junto com suas respectivas roles e permissões do funcionário, após a análise se não estiver adicionado(a), é criado o funcionário. Professor cria categoria. Professor cria área temática. Professor cria a modalidade. O organizador do evento cria os cupons e as promoções.
Fluxo alternativo	Funcionário existente. Categoria existente. Instituição existente. Permission existente. Role existente. Área temática existente. Modalidade existente. Cupons existentes. Promoções existentes.
Pós-condições	Criado com sucesso.

Fonte: Autoria Própria

Figura 18 - Diagrama de Caso de uso Eventos



Fonte: Autoria Própria

Tabela 60 - Descrição de caso de uso Efetuar Inscrição

Nome do caso de uso	UC03 - Inscrição no(s) Evento(s)
Descrição	Este caso de uso descreve o processo de inscrição de um aluno em um ou mais eventos, incluindo o pagamento, verificação do ingresso, cancelamento e recebimento de certificado.
Ator	Aluno
Pré-condições	Aluno precisa estar logado no sistema.
Fluxo de principal	Aluno acessa a lista de eventos disponíveis no sistema. Aluno seleciona o evento para o qual deseja se inscrever. Sistema verifica se o evento é pago: Se for pago, o aluno é direcionado para a página de pagamento.

	Aluno realiza o pagamento do evento (se aplicável). Sistema confirma a inscrição do aluno e emite um ingresso digital. Aluno verifica o ingresso na sua conta. Aluno pode optar por cancelar a inscrição dentro do prazo determinado: Se o aluno cancelar, o sistema processa o estorno do valor (se houver). Aluno faz o check-in no evento no dia correspondente. Após o evento, o aluno pode acessar e baixar o certificado de participação.
Fluxo alternativo	Aluno não fez check In. Evento Cancelado.
Pós-condições	Evento concluído com sucesso. Aluno recebeu o ingresso e o certificado, se participou do evento.

Fonte: Autoria Própria

Tabela 61 - Descrição de caso de uso da criação do evento e proposta do mesmo.

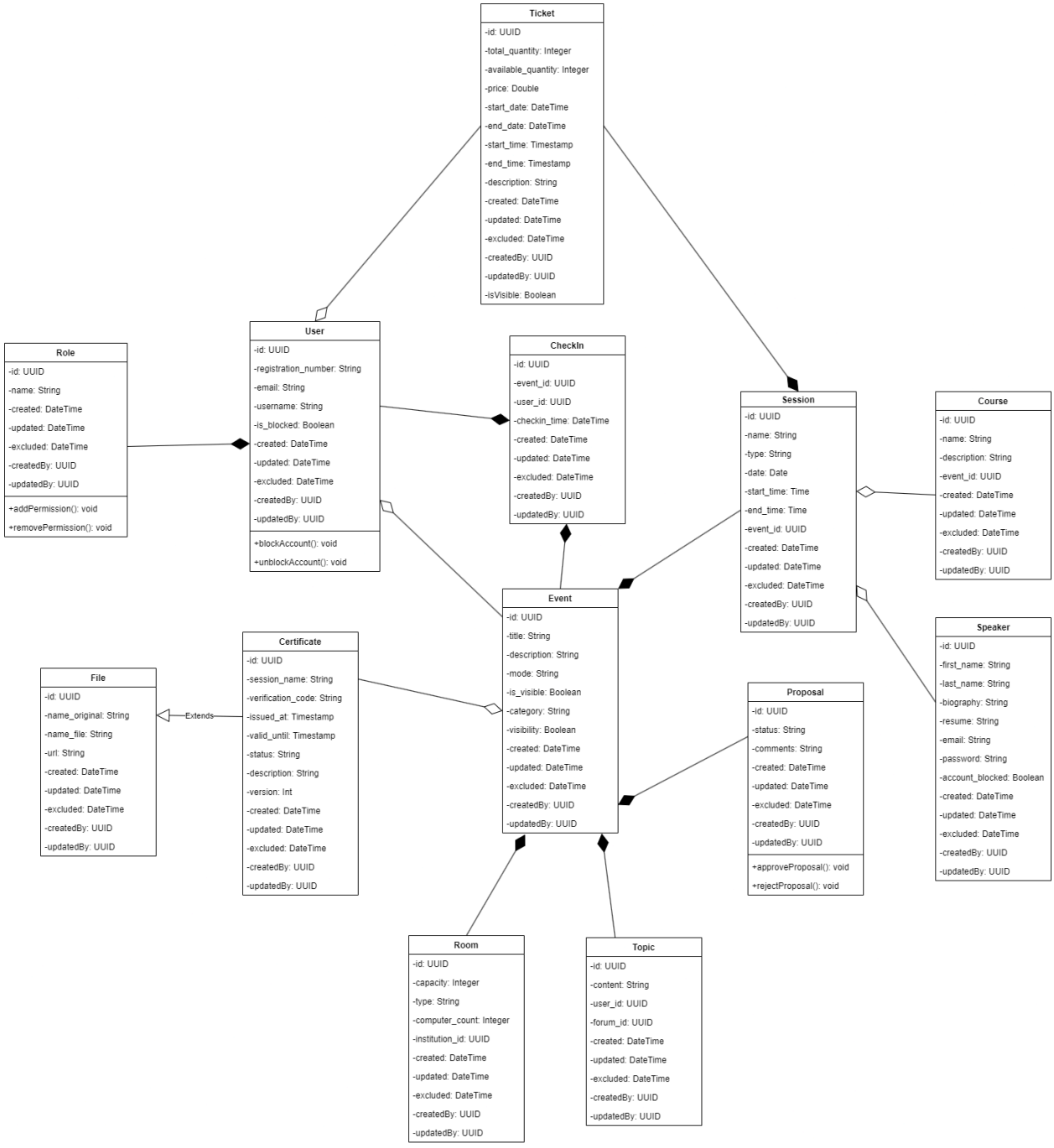
Nome do caso de uso	UC04 - Criação do evento e da proposta.
Descrição	Caso de uso, onde o professor fará uma proposta de evento, que incluirá a alocação da sala e após aprovada, a criação do evento será iniciada e o organizador do evento irá adicionar o palestrante para o evento.
Ator	Professor, Avaliador, Organizador do evento e Palestrante.
Pré-condições	Professor, Avaliador, Organizador do evento e Palestrante precisam estar logados no sistema.
Fluxo de principal	Professor acessa a funcionalidade de criação de proposta de evento no sistema. Professor preenche os detalhes da proposta, incluindo título, descrição, data, hora e alocação de sala. Professor submete a proposta para avaliação. Professor criar um tópico de discussão do evento. Sistema registra a proposta e notifica o Avaliador. Avaliador revisa a proposta. Avaliador aprova ou rejeita a proposta:

	Se aprovada, poderá fazer o cadastro do evento. Se rejeitada, o Professor é notificado e pode revisar a proposta. Sistema cria o evento baseado na proposta aprovada. Organizador do Evento acessa o evento recém-criado. Organizador do Evento adiciona palestrantes ao evento. Organizador do Evento gerar o Relatório de despesas do Evento. Sistema confirma a adição dos palestrantes e notifica os Palestrantes.
Fluxo alternativo	1: Aluno não fez check-in. Se um aluno não fez check-in, o sistema notifica e não permite que ele participe do evento. 2: Evento cancelado. Se o evento for cancelado, o sistema notifica todos os envolvidos e atualiza o status do evento.
Pós-condições	Evento concluído com sucesso. Proposta registrada e aprovada. Palestrantes adicionados ao evento.

Fonte: Autoria Própria

3.6.2 Diagrama de classe

Figura 19 - Diagrama de Classe de Entidades



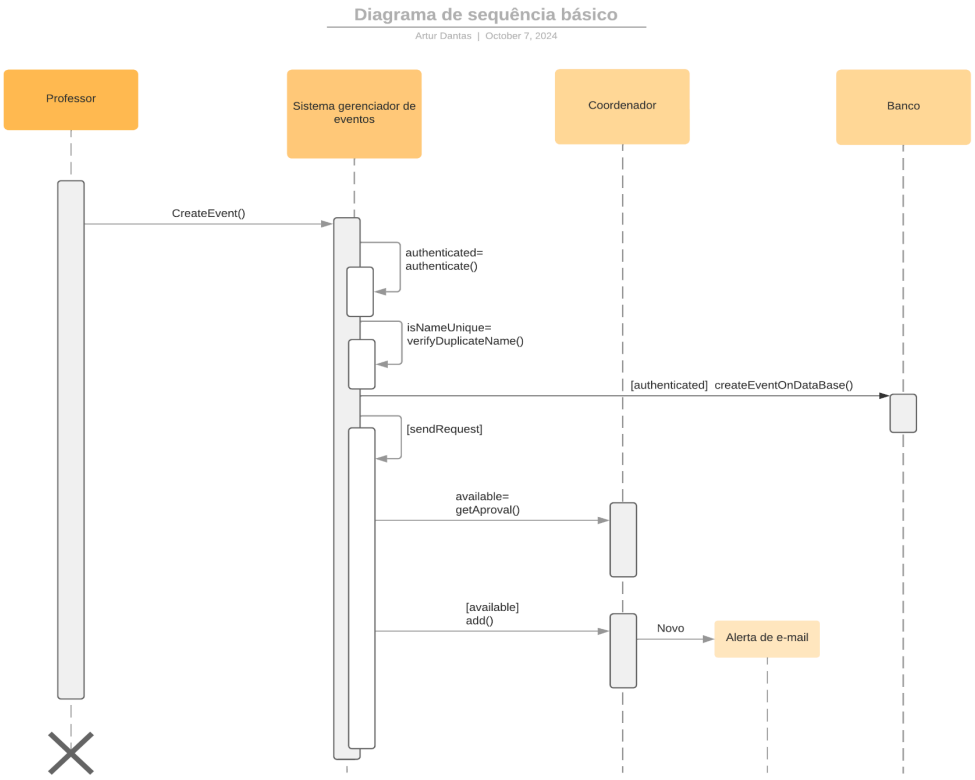
Event representa o evento principal. Se relaciona com várias outras entidades, como Session, Ticket, Checkin, Certificate, Proposal, e Room, mostrando que um evento pode envolver sessões, ingressos, check-ins, certificados, propostas e salas. Room é a UML das salas onde as sessões dos eventos ocorrem. Está associada a Event, indicando que um evento está associado a uma sala. Topic representa tópicos discutidos em fóruns relacionados ao evento. Está associado a User, sugerindo que o usuário pode criar tópicos em um fórum. Certificate é o UML dos certificados emitidos para os participantes dos eventos. Está associado a Even, indicando que o certificado é emitido sobre um evento específico. File representa arquivos que podem ser associados a diversas entidades.

Proposal representa uma proposta para um evento. Está associada a Event, indicando que um usuário pode submeter propostas para a instituição para apoio em eventos específicos. Session representa uma sessão específica dentro de um evento. Está diretamente associada a um Event, indicando que uma sessão pertence a um evento específico e um evento pode ter várias sessões. Course representa cursos que podem estar associados a uma sessão e são oferecidos pelo evento. Speaker é o UML de um palestrante do evento. Está associado a Session, sugerindo que cada palestrante pode estar vinculado a uma ou mais sessões de um evento.

Checkin representa o check-in do usuário em um evento. Está associado a User e Event, indicando que o check-in é feito por um usuário específico em um evento. Ticket representa os ingressos para participar de um evento. Está associado a uma Session e um User, indicando que os ingressos são emitidos para uma sessão específica e um usuário específico. User representa os usuários do sistema, como participantes ou organizadores de eventos. Essa classe se relaciona com Role para definir as permissões e papéis dos usuários, e está associado a entidades como Checkin, Certificate, e Ticket, sugerindo que um usuário pode fazer check-in em eventos, receber certificados e receber seus tickets. Role define as funções ou permissões dos usuários, como administrador, participante, etc.

3.6.3 Diagrama de Sequência

Figura 20 - Diagrama de Sequência



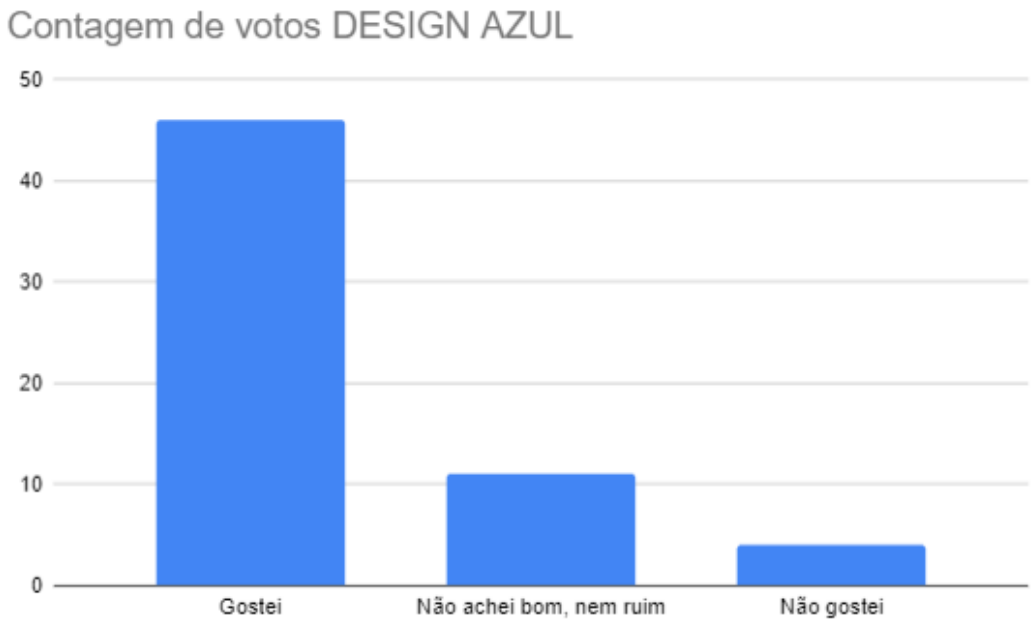
Fonte: Autoria própria

4. RESULTADOS PARCIAIS

Até o momento, o desenvolvimento do sistema de gerenciamento de eventos acadêmicos, o EventsLink, nos trouxe alguns resultados através de pesquisas que realizamos com a utilização do Google Forms, ferramenta para gerar formulários. Após algumas discussões a respeito de como organizar o design, como na questão de cores, decidimos por abrir espaço e receber uma opinião popular para qual seria a melhor cor para utilizar no projeto, e com isso recebemos 61 respostas no total.

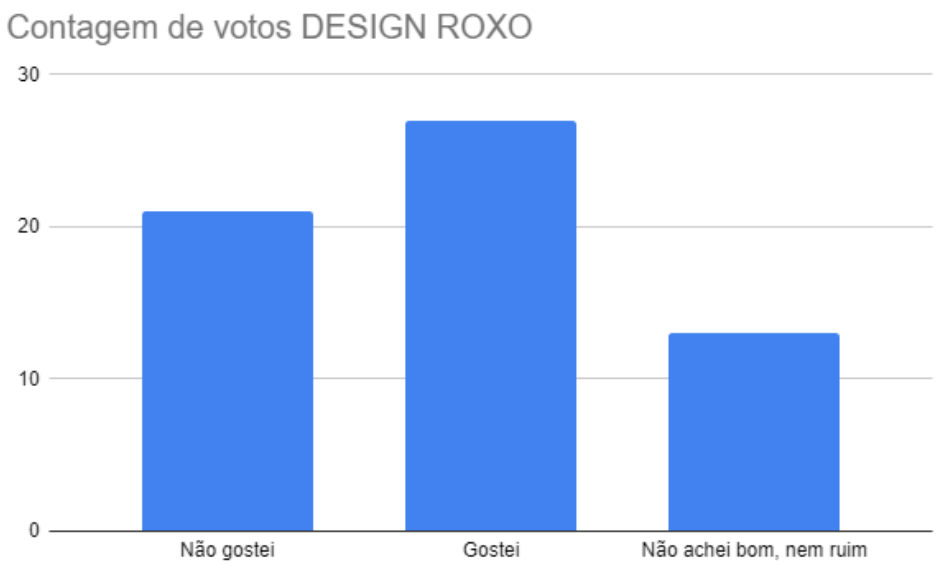
Com 3 opções de cor e 3 opções de resposta cada pergunta, sendo elas “Gostei”, “Não achei bom, nem ruim” e “Não gostei”, o design com cor azul ficou em primeiro lugar, com 46 votos em “Gostei”, representando 75.4% dos votos “Gostei” da cor azul. Em segundo lugar ficou a cor roxa, com 27 votos em “Gostei”, representando 44.3% de votos “Gostei” da cor roxa. E em terceiro lugar ficou a cor laranja, com 25 votos em “Gostei”, representando 25% dos votos “Gostei” da cor laranja.

Figura 21 - Gráfico de Contagem de votos Design Azul



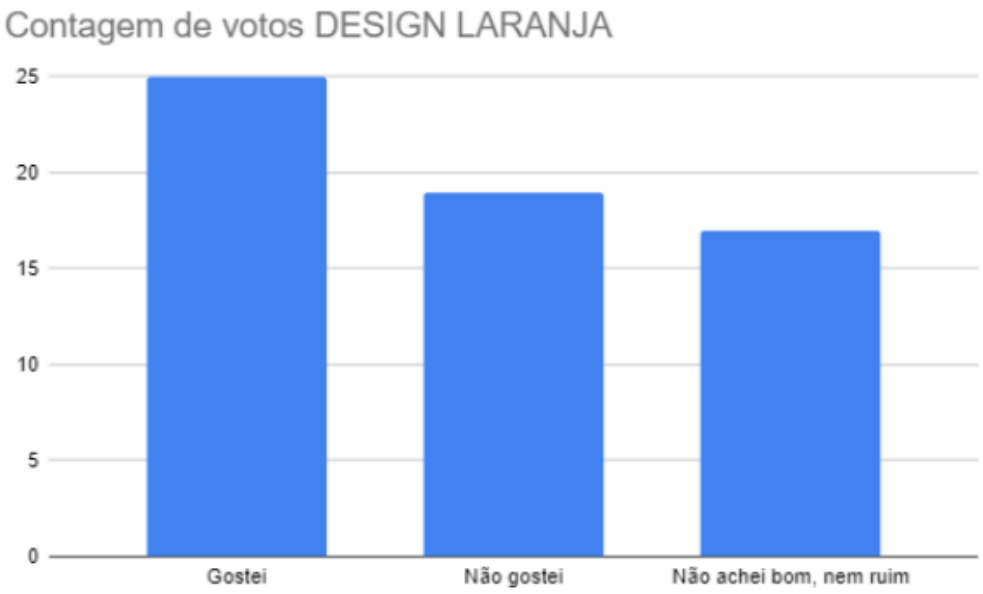
Fonte: Autoria Própria

Figura 22 - Gráfico de Contagem de votos Design Roxo



Fonte: Autoria Própria

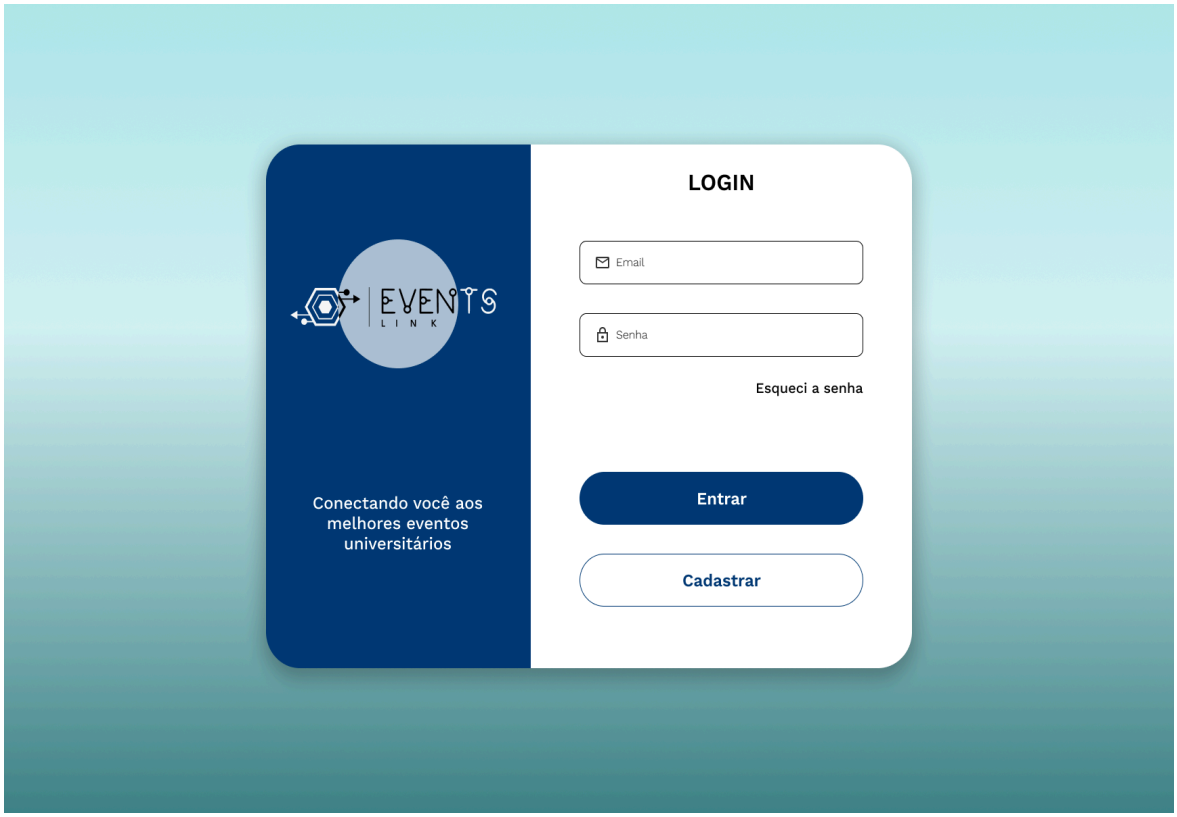
Figura 23 - Gráfico de Contagem de votos Design Laranja



Fonte: Autoria Própria

Com esses dados coletados, conseguimos uma base para começar a construir alguns protótipos de interface de usuário via Figma. Utilizamos como base a cor com maior quantidade de votos e, com ajuda de alguns modelos coletados via pesquisa web, construímos o primeiro protótipo para design web, que são as telas de login de usuário, e também a tela com a proposta do evento.

Figura 24 - Protótipo tela de login

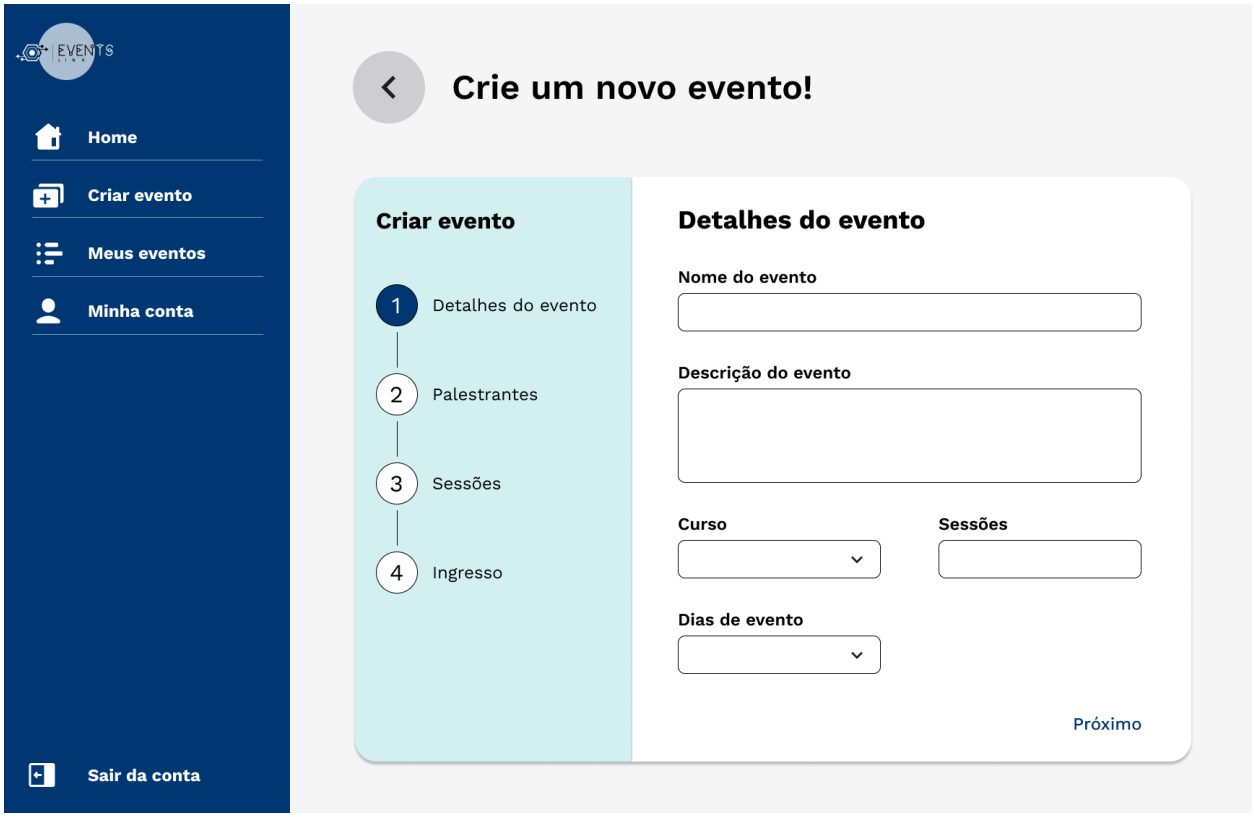


Fonte: Autoria Própria

Figura 25 - Protótipo tela de proposta de evento

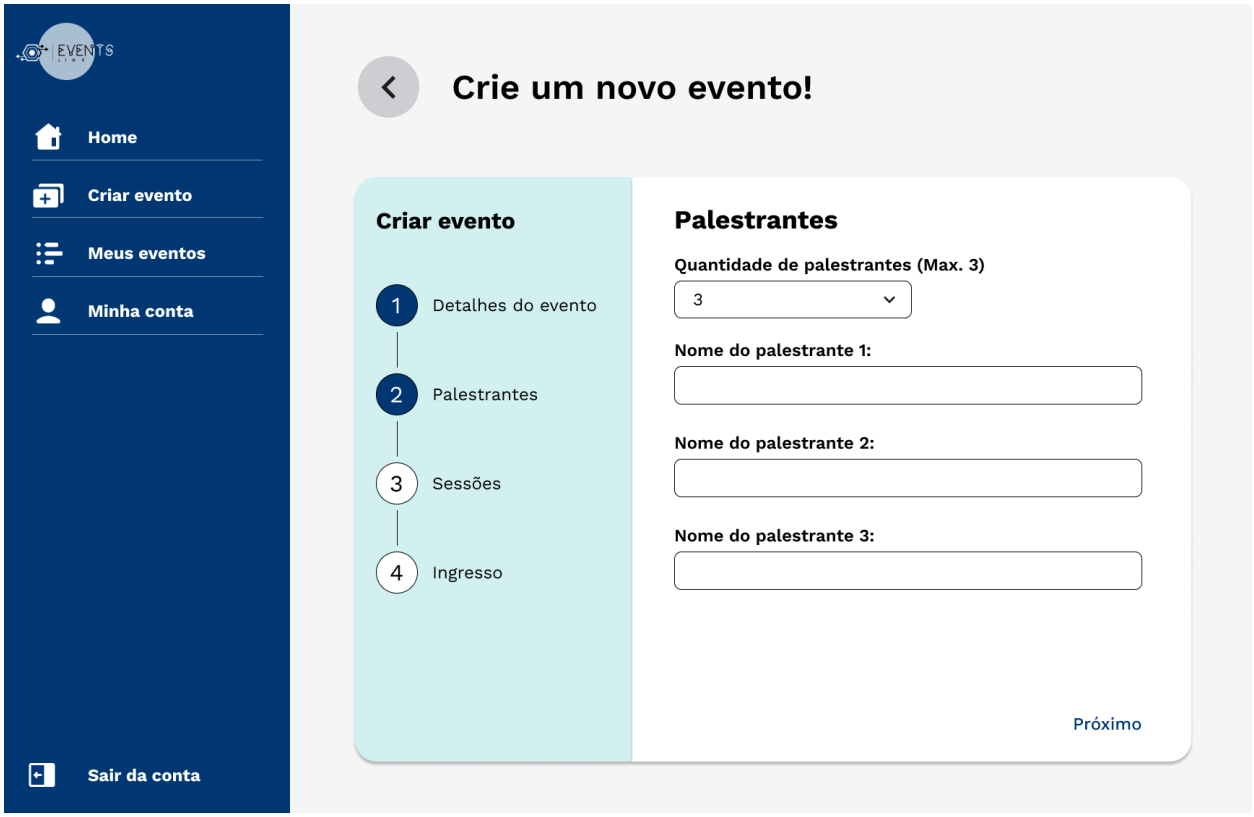
Fonte: Autoria Própria

Figura 26 - Protótipo tela de criação de evento detalhes



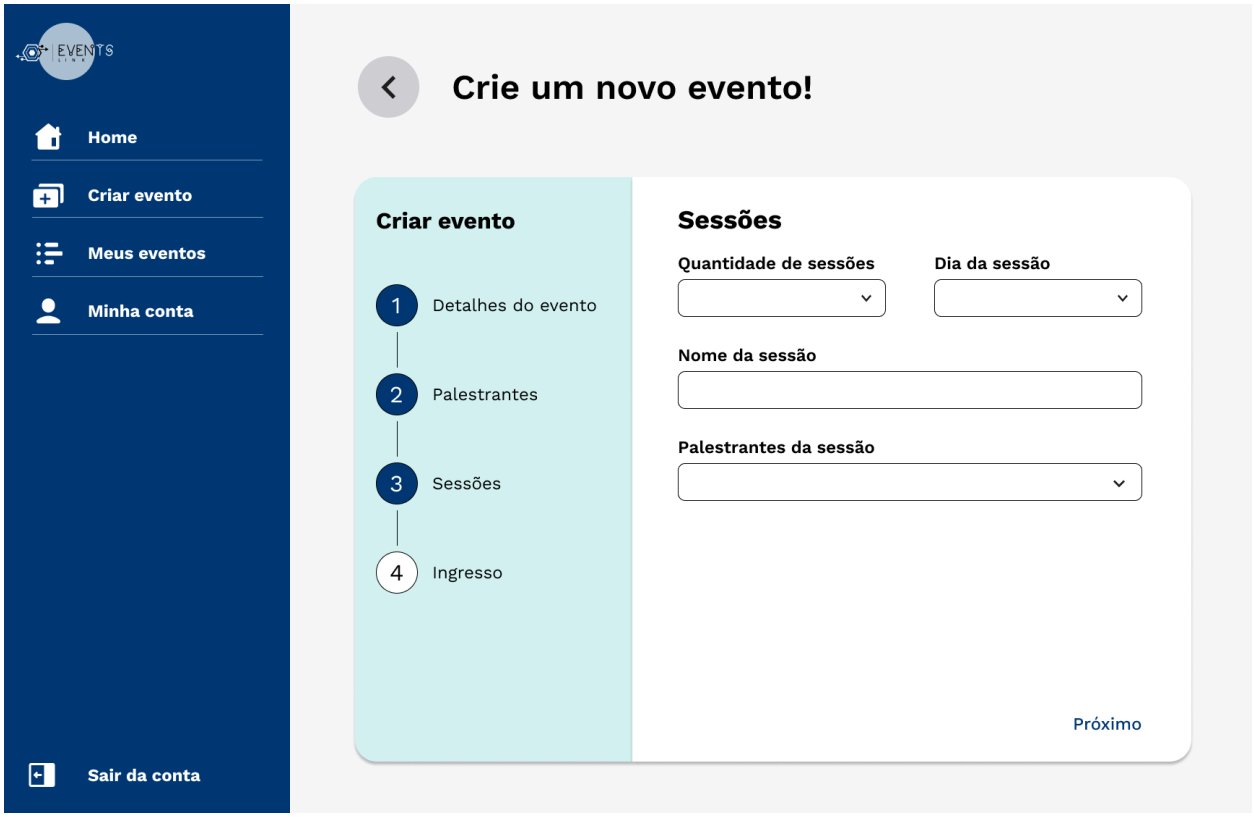
Fonte: Autoria Própria

Figura 27 - Protótipo tela de criação de evento palestrantes



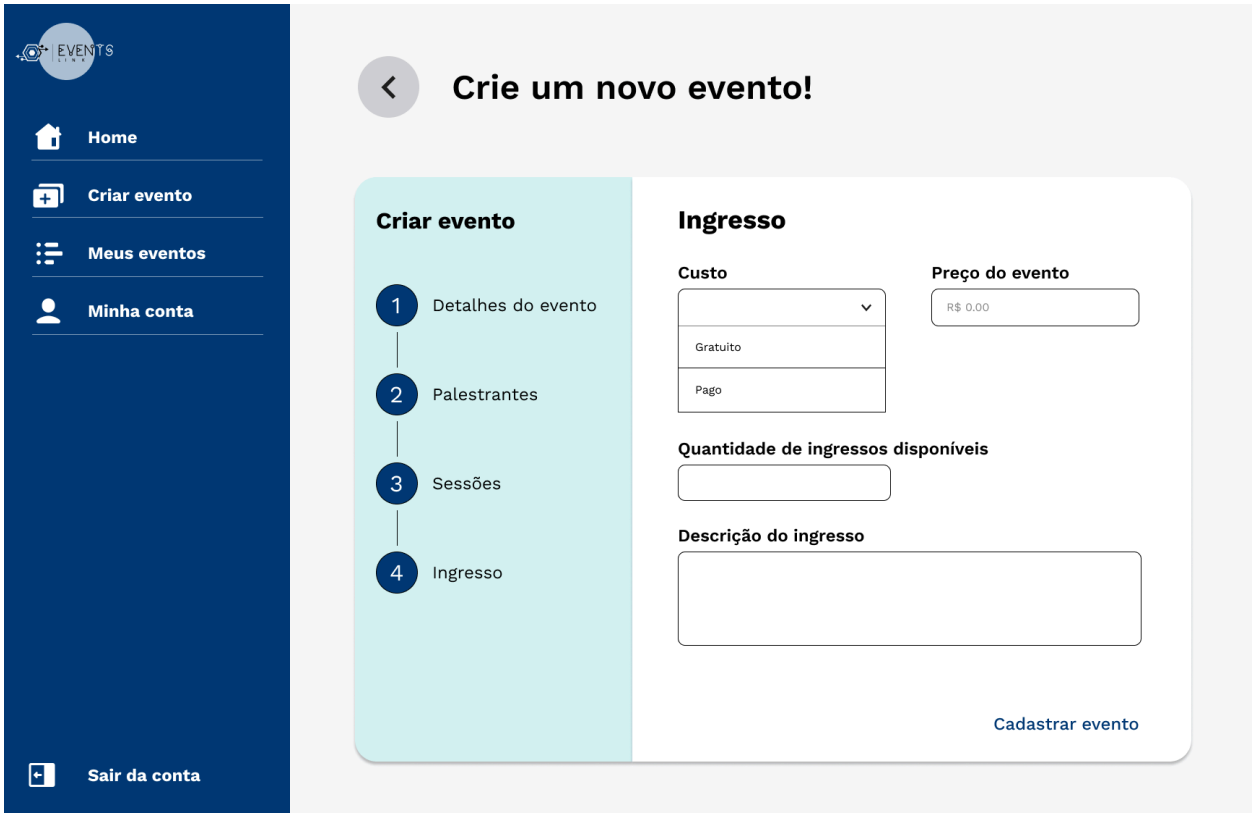
Fonte: Autoria própria

Figura 28 - Protótipo tela de criação de evento sessões



Fonte: Autoria própria

Figura 29 - Protótipo tela de criação de evento ingresso



Fonte: Autoria própria

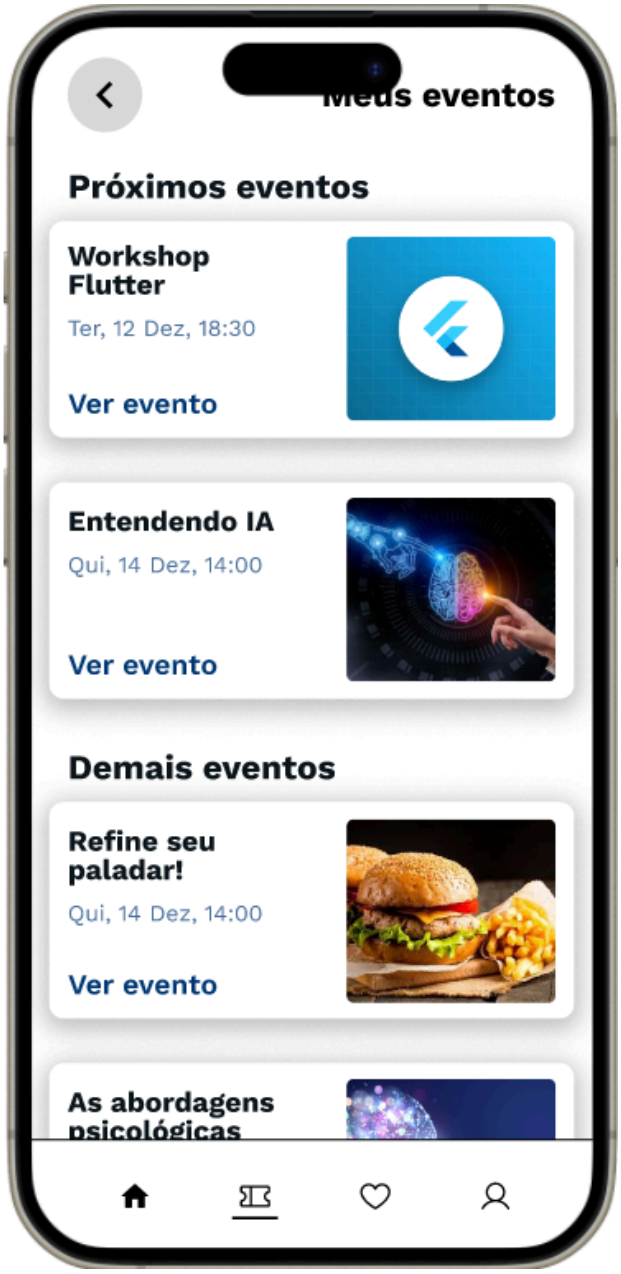
Figura 30 - Protótipo da tela inicial mobile

Fonte: Autoria Própria

Figura 31 - Protótipo da tela mobile de descrição do evento

Fonte: Autoria Própria

Figura 32 - Protótipo da tela mobile de eventos inscritos



Fonte: Autoria Própria

5. SÍNTESE

O desenvolvimento do sistema de gerenciamento de eventos acadêmicos, o EventsLink, representa um avanço significativo para a comunidade do Centro Universitário do Distrito Federal (UDF). Criado para atender às necessidades de alunos, docentes e administradores, o sistema centraliza todas as informações sobre eventos acadêmicos, facilitando o acesso dos alunos a palestras, workshops e demais atividades complementares que enriquecem sua formação**.**

Desde o início, o projeto foi guiado pela busca por uma solução que simplificasse a jornada dos estudantes na universidade. A escolha de tecnologias modernas foi essencial para garantir a funcionalidade e a experiência amigável do sistema. A utilização do Flutter para o desenvolvimento do aplicativo mobile, por exemplo, permite que alunos utilizem a plataforma em dispositivos Android e iOS, com interfaces intuitivas e responsivas. Aliado ao padrão arquitetural MVVM, que separa claramente as funções da interface, da lógica de negócios e dos dados, o Flutter viabilizou uma aplicação organizada, fácil de manter e pronta para evoluir conforme novas necessidades surgirem. No backend, o uso do Spring Boot trouxe segurança e escalabilidade, permitindo que o sistema suporta grandes volumes de usuários e dados com desempenho estável.

A arquitetura modular baseada no padrão Package by Feature contribui para a flexibilidade do EventsLink. A separação clara entre as diferentes funcionalidades não só facilita a manutenção, mas também torna mais simples a expansão do sistema, agregando novas funcionalidades conforme o crescimento do UDF e o surgimento de novas demandas. A integração com o banco de dados PostgreSQL reforça essa robustez e escalabilidade, garantindo que o sistema lide com grandes volumes de dados com integridade e segurança, essenciais para preservar a confidencialidade e a confiabilidade das informações dos usuários.

O impacto do EventsLink vai além da praticidade de suas funcionalidades; ele transforma a forma como alunos e docentes se comunicam e participam das atividades acadêmicas. Com um

canal centralizado para divulgar e organizar eventos, a universidade tem agora uma comunicação direta e eficaz com seus alunos, o que fortalece o engajamento deles com o meio acadêmico. Além disso, a possibilidade de emissão de certificados diretamente pelo sistema facilita o processo de registro de horas complementares, permitindo que os alunos aproveitem integralmente as oportunidades que enriquecem sua formação acadêmica sem enfrentarem burocracia. Em outras palavras, o EventsLink encurta o caminho entre a vontade de aprender e o acesso às atividades que promovem esse aprendizado.

A experiência de desenvolvimento do sistema também foi marcante para a equipe, que trabalhou em sintonia com os stakeholders e os usuários finais, garantindo que cada ajuste e decisão fossem orientados pela experiência prática dos alunos e pela visão da universidade. Esse contato próximo permitiu que o sistema refletisse as necessidades reais da comunidade acadêmica, proporcionando uma solução humanizada, que prioriza a simplicidade e a acessibilidade. Para que o sistema funcione de forma segura e responsável, adotamos práticas rigorosas de proteção de dados e controle de acesso, assegurando que as informações dos usuários sejam tratadas com cuidado e estejam em conformidade com as regulamentações de privacidade.

O EventsLink, portanto, não é apenas um sistema; ele é uma ferramenta que inspira uma nova experiência acadêmica, em que a organização e o engajamento com eventos tornam-se acessíveis e dinâmicos. Ele valoriza o tempo dos alunos e permite que se concentrem no que realmente importa: aprender, crescer e se conectar com o conhecimento e com a comunidade universitária. O sucesso do projeto é uma prova do poder da inovação tecnológica no ambiente acadêmico e da importância de soluções personalizadas e humanizadas para melhorar a experiência estudantil.

Com o EventsLink, o UDF estabelece um novo padrão para a gestão de eventos, facilitando a vida dos estudantes e aproximando a comunidade universitária, num esforço conjunto para garantir que cada aluno tenha à sua disposição uma jornada acadêmica mais rica e organizada.

5.1 Atividades Previstas

Na sessão de atividades previstas do nosso projeto, destacamos algumas funcionalidades que buscam enriquecer a experiência dos alunos, incentivando o engajamento, promovendo acessibilidade e ampliando o alcance dos eventos acadêmicos. Esses aprimoramentos refletem um compromisso em oferecer uma plataforma mais completa, interativa e inclusiva. Vamos aos detalhes:

- **Integração com Ferramentas Externas e Redes Sociais**

  Para tornar o EventsLink uma ferramenta de conexão entre alunos e a comunidade acadêmica, planejamos integrar a plataforma com redes sociais e ferramentas externas de comunicação. Essa funcionalidade permitirá que os estudantes compartilhem eventos de interesse em suas redes pessoais, ampliando a visibilidade e o impacto das atividades universitárias. Além disso, a integração com plataformas de videoconferência possibilitará uma experiência de fácil acesso a eventos online, permitindo que estudantes, professores e convidados participem de maneira remota e dinâmica.

- **Gamificação e Incentivo à Participação**

  Incentivar a participação ativa dos alunos é um dos pilares do nosso projeto. Por isso, estudamos a introdução de elementos de gamificação, como pontuação, conquistas e recompensas para quem participa dos eventos e completa desafios acadêmicos. Esse sistema cria uma jornada mais interativa e motivadora para o estudante, estimulando o envolvimento em atividades que vão além da sala de aula. A gamificação transforma o processo de aprendizado e engajamento em uma experiência divertida e recompensadora, incentivando os alunos a explorar mais o universo acadêmico.

- **Internacionalização do Sistema e Acessibilidade**

  Reconhecendo a diversidade e as diferentes necessidades dos nossos usuários, queremos tornar o EventsLink acessível e acolhedor para todos. A internacionalização do sistema, com suporte a múltiplos idiomas, permitirá que alunos estrangeiros sintam-se parte da comunidade acadêmica sem barreiras linguísticas. Além disso, estamos comprometidos em melhorar as funcionalidades de acessibilidade para garantir que estudantes com necessidades especiais possam navegar e utilizar a plataforma com facilidade. Essa adaptação visa a criar uma experiência digital inclusiva, onde todos têm igual acesso às oportunidades e eventos oferecidos pela universidade.

REFERÊNCIAS BIBLIOGRÁFICAS

HIGHSMITH, Jim. Agile Software Development Ecosystems. Addison-Wesley, 2002. https://archive.org/details/agilesoftwaredev0000high. Acesso em 25 abr. 2025

BECK, Kent. Manifesto for Agile Software Development, 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 28 abr. 2025

ALMEIDA, R. Desenvolvimento de uma aplicação web voltada para a gestão de um evento anual específico realizado na Universidade Federal Fluminense (UFF). 2020. Acesso em: 19 ago. 2024.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML: guia do usuário. 2. ed. São Paulo: Elsevier, 2006. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=ddWqxcDKGF8C&oi=fnd&pg=PR13&dq=UML&ots=fgxMpihGPM&sig=6-EWUzDdemFNH_TwGvnvGXtN8lw&redir_esc=y#v=onepage&q=OMG&f=false. Acesso em: 18 set. 2024.

CARVALHO, Vinícius. PostgreSQL: Banco de dados para aplicações web modernas. 1. ed. São Paulo: Novatec, 2017. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=KlFVDgAAQBAJ&oi=fnd&pg=PT3&dq=postgresql&ots=1_otk6TT3u&sig=DL1IsIMBo7dPbPIrT-J9ZECWuwU&redir_esc=y#v=onepage&q&f=false. Acesso em: 16 set. 2024.

DEITEL, Paul; DEITEL, Harvey. *Java: como programar*. 10. ed. São Paulo: Pearson, 2016. p. 13.

DIO. O que é o Spring Framework e para que ele é usado. Disponível em: https://www.dio.me/articles/o-que-e-o-spring-framework-e-para-que-ele-e-usado. Acesso em: 8 set. 2024.

EDWARDS, C. Why React is the Preferred Choice Over Angular and Vue. JavaScript Journal, 2021. Disponível em: https://javascriptjournal.com/react-vs-angular-vs-vue. Acesso em: 13 ago. 2024.

EVEN3. Even3 - Plataforma de Gestão de Eventos. Disponível em: https://www.even3.com.br/. Acesso em: 13 ago. 2024.

EVENTIM. Eventim - Ingressos para Shows e Eventos. Disponível em: https://www.eventim.com.br/. Acesso em: 13 ago. 2024.

FLANAGAN, D. JavaScript: The Definitive Guide. 7. ed. O'Reilly Media, 2020.

FREECODECAMP. Uma maneira melhor de estruturar projetos em React. Disponível em: https://www.freecodecamp.org/portuguese/news/uma-maneira-melhor-de-estruturar-projetos-em-react/. Acesso em: 04 set. 2024, às 14:48.

HEJLSBERG, A.; GAMMON, D.; STEVENS, W. TypeScript Language Specification. Microsoft, 2018. Disponível em: https://www.typescriptlang.org. Acesso em: 13 ago. 2024.

HEUSER, C. A. Projeto de Banco de Dados. 6ed, Ed. Artmed, 2009.

JAVA. O que é Java. Disponível em: https://www.java.com/en/download/help/whatis_java.html. Acesso em: 8 set. 2024.

KENT, R. Cost Management in Plastics Processing. Elsevier, 2018. Acesso em: 23 set. 2024.

LEMOS, P.; ZANATTA, F.; ZAINA, L. Proposta para o desenvolvimento de um sistema de gerenciamento de eventos destinado às atividades do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP). 2020. Acesso em: 20 ago. 2024.

PHUER, O. Package by feature, not layer – An old idea revisited. Disponível em: https://phauer.com/2020/package-by-feature/. Acesso em: 20 set. 2024.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: McGraw Hill, 2011.

PROGRAMMING PULSE. Hexagonal vs Clean vs Onion Architectures. Disponível em: https://programmingpulse.vercel.app/blog/hexagonal-vs-clean-vs-onion-architectures. Acesso em: 13 ago. 2024.

SANZONE, G. MVVM: Simplificando o desenvolvimento mobile para iniciantes. Disponível em: https://medium.com/orangejuicefc/mvvm-simplificando-o-desenvolvimento-mobile-para-iniciantes-419ade97bc42. Acesso em: 5 set. 2024.

SIQUEIRA, J.; SILVA, M. Desenvolvimento de uma aplicação voltada para o gerenciamento de eventos específicos da universidade UniEVANGÉLICA. 2018. Acesso em: 15 ago. 2024.

SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SNYK. JVM Ecosystem Report 2018: Platform & Application. Disponível em: https://snyk.io/pt-BR/blog/jvm-ecosystem-report-2018-platform-application/. Acesso em: 8 set. 2024.

SPRING.IO. Why Spring? Disponível em: https://spring.io/why-spring. Acesso em: 08 set. 2024.

SUTHERLAND, Jeff. Scrum - A arte de fazer o dobro pela metade do tempo. 1. ed. São Paulo: Leya, 2014. Disponível em:

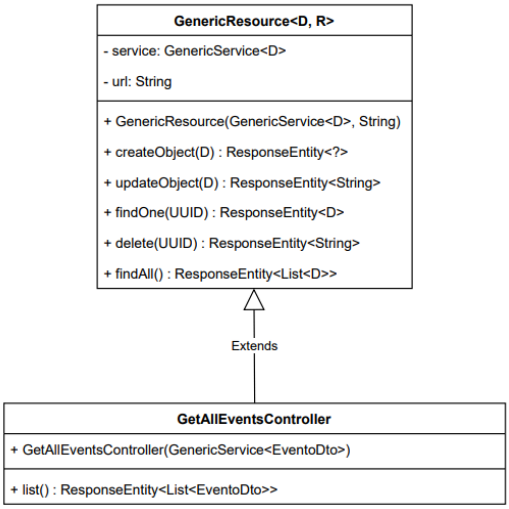
https://edisciplinas.usp.br/pluginfile.php/8000882/mod_resource/content/1/SCRUM%20-%20A%20Arte%20de%20Fazer%20o%20Dobro.pdf. Acesso em 18 set. 2024.

SYMPLA. Sympla - Ingressos para Eventos, Teatros, Shows, Cursos e mais. Disponível em: https://www.sympla.com.br/. Acesso em: 13 ago. 2024.

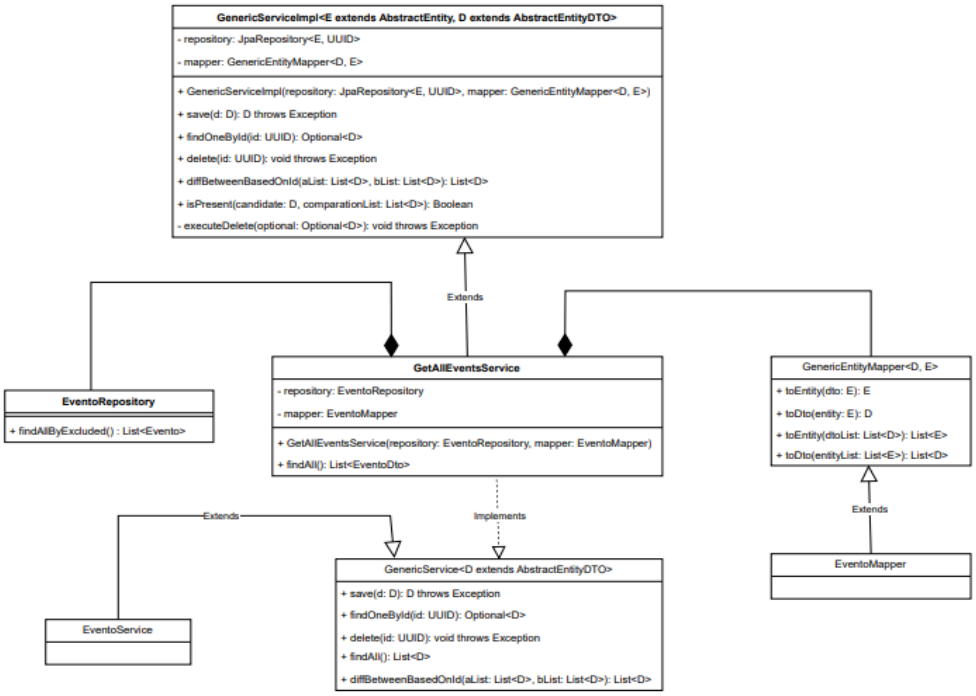
TAUTION, Cezar. Big Data: princípios e melhores práticas. Rio de Janeiro: Brasport, 2013. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=GAVLAgAAQBAJ&oi=fnd&pg=PT11&dq=Big+Data&ots=YTcrmXvgtH&sig=OvkY-deSBw9sZu-uUWSg409Vnms&redir_esc=y#v=onepage&q&f=false. Acesso em: 18 set. 2024.

APÊNDICES

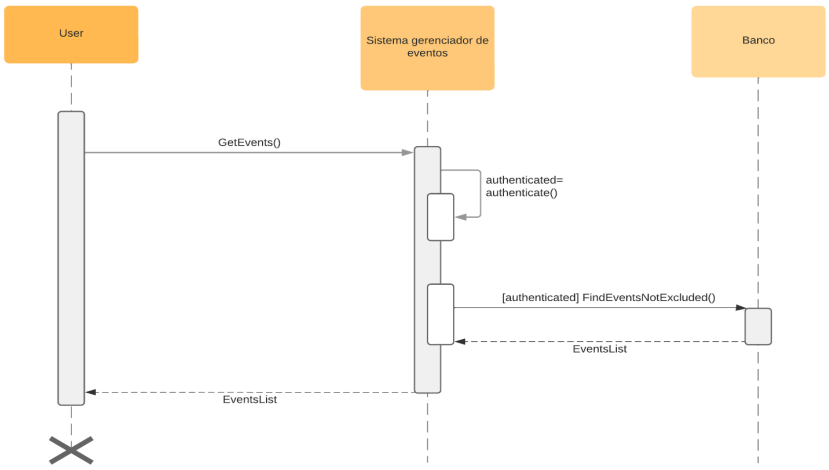
Apêndice A - Diagrama de classe controller de eventos



Apêndice B - Diagrama de Classe Services, Mappers e Repository

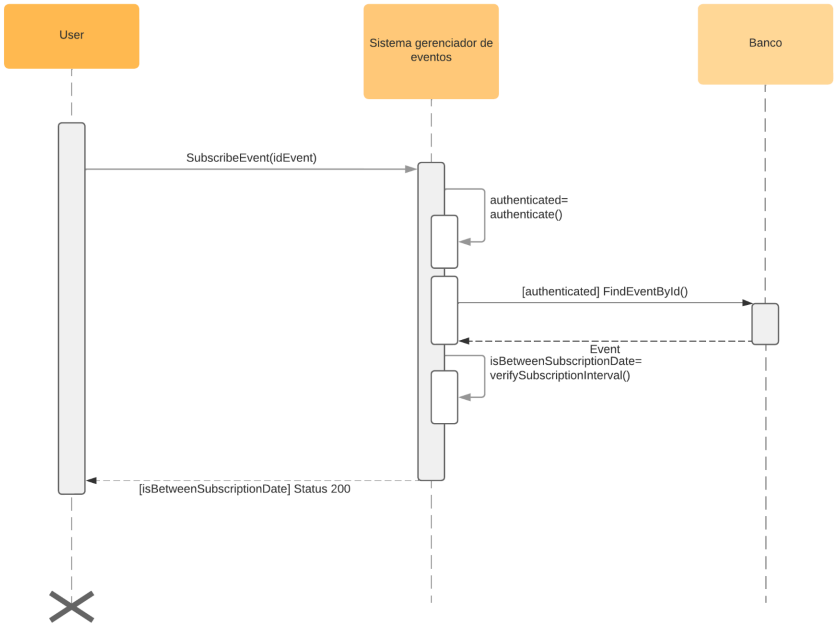


Apêndice C - diagrama de sequência listagem de eventos



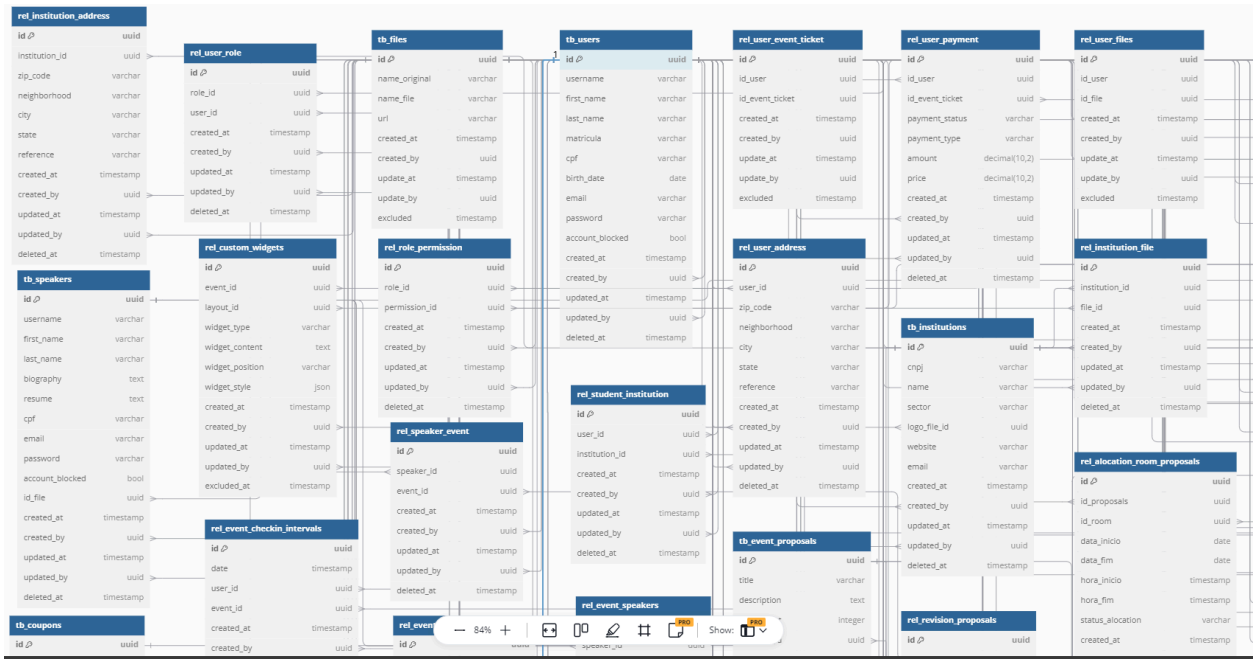
Fonte: Autoria própria

Apêndice D - Diagrama de sequência inscrição evento



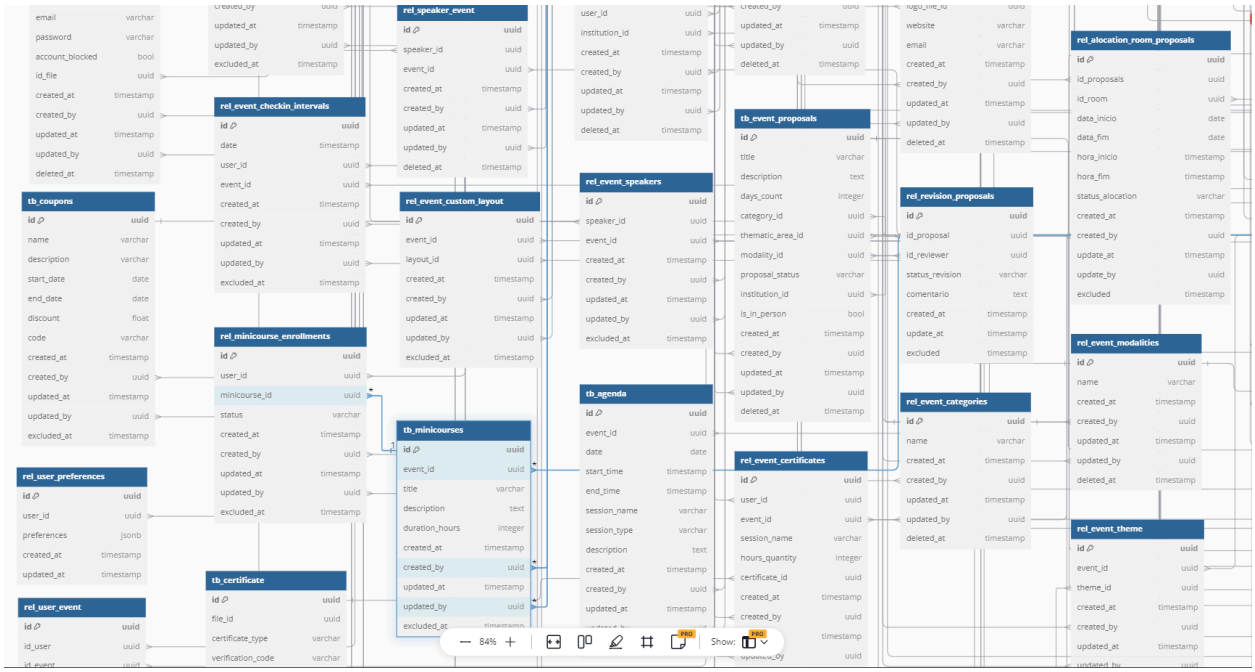
Fonte: Autoria própria

Apêndice E - Fragmento 1



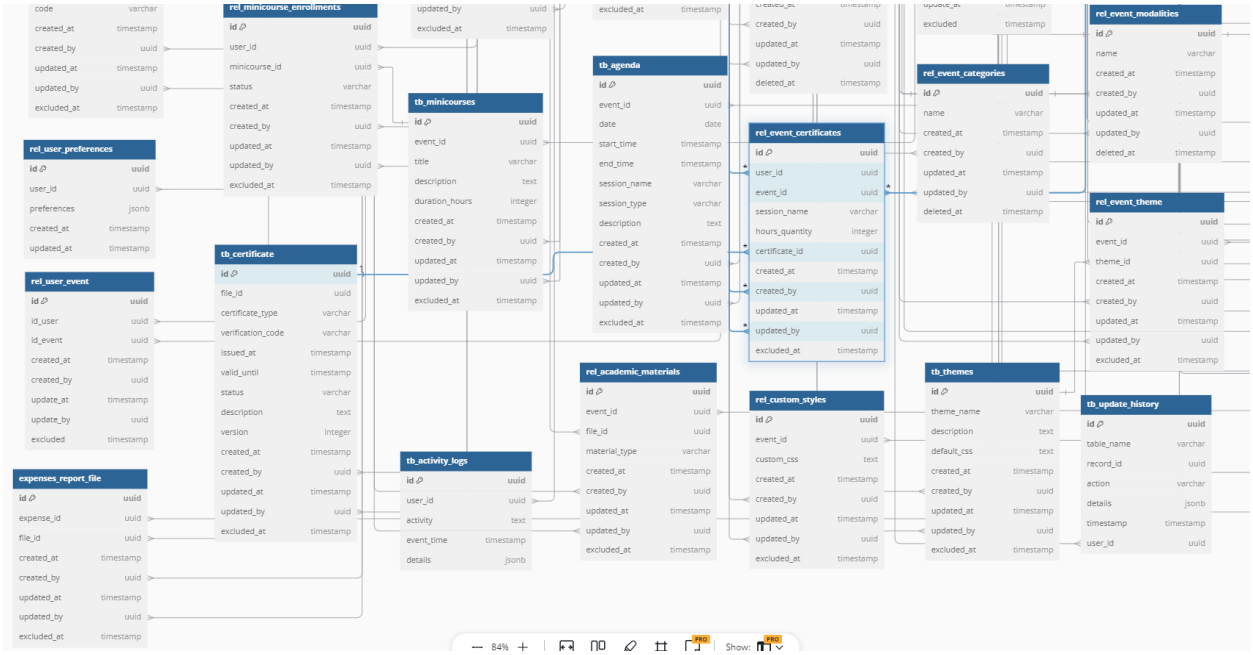
Fonte: Autoria Própria

Apêndice F - Fragmento 2



Fonte: Autoria Própria

Apêndice G - Fragmento 3

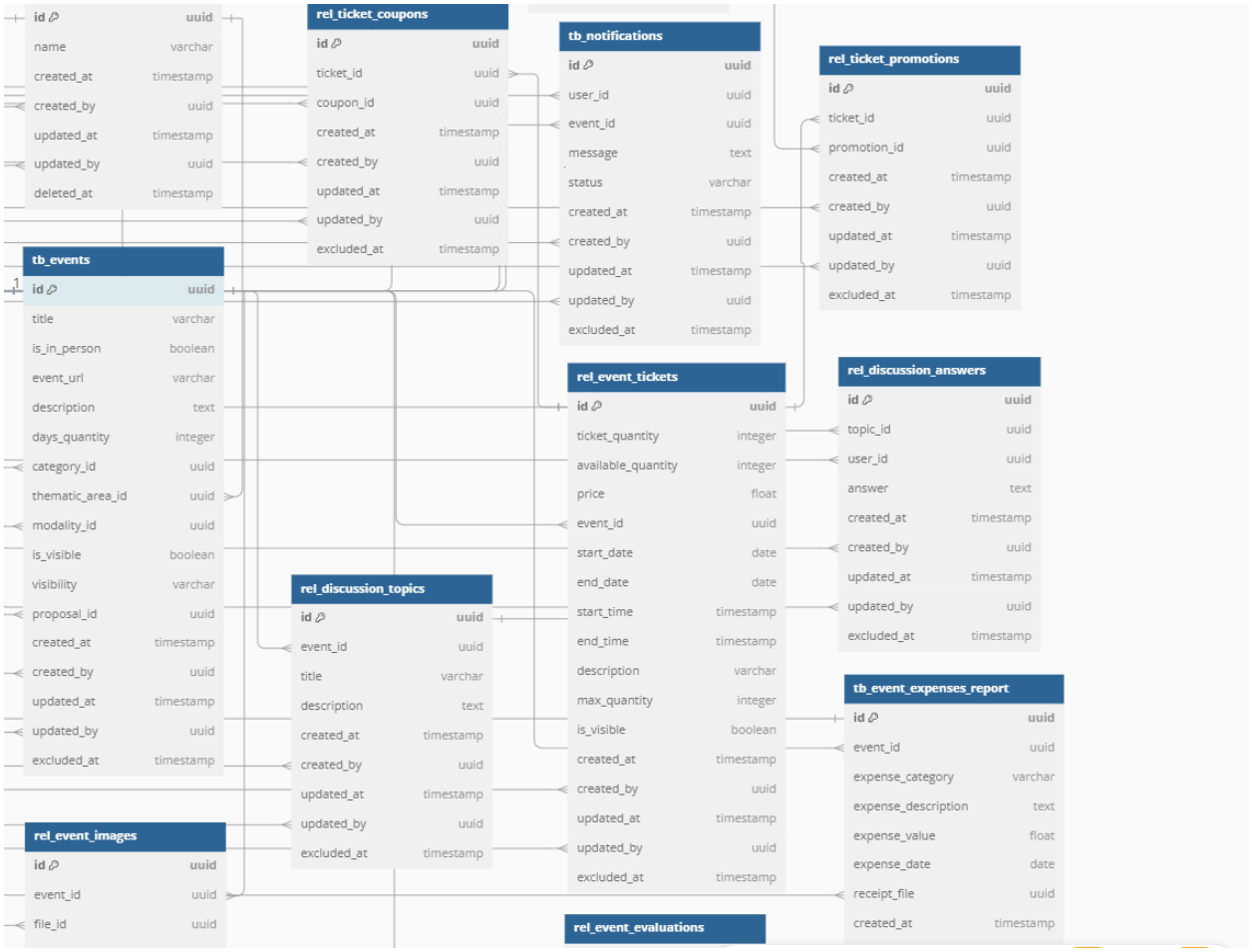


Fonte: Autoria Própria

Apêndice H - Fragmento 4

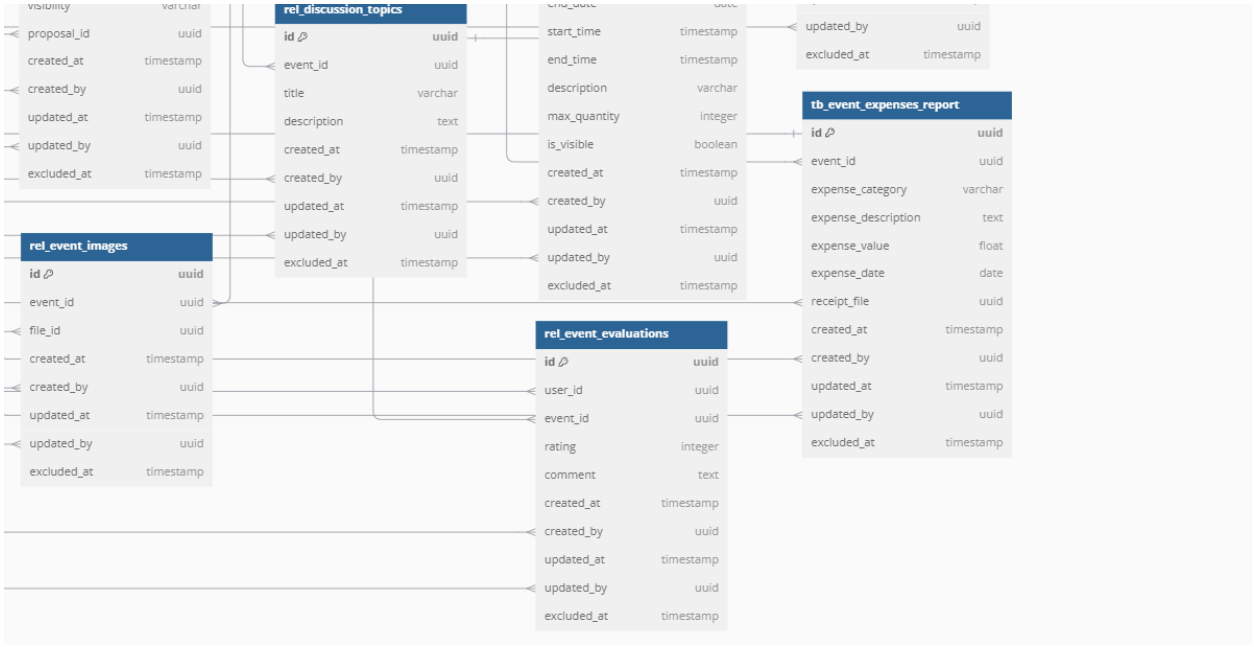
Fonte: Autoria Própria

Apêndice I - Fragmento 5



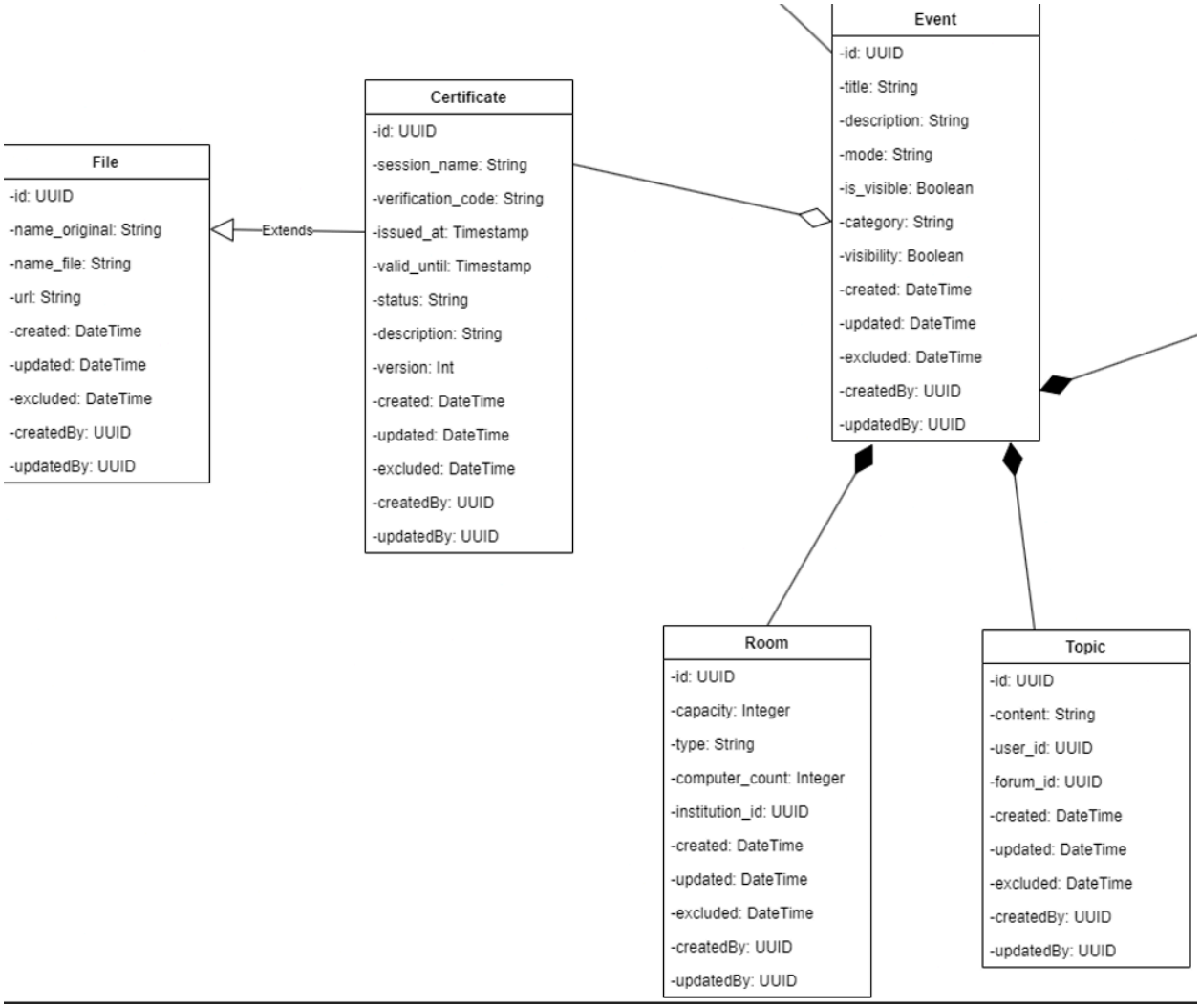
Fonte: Autoria Própria

Apêndice J - Fragmento 6



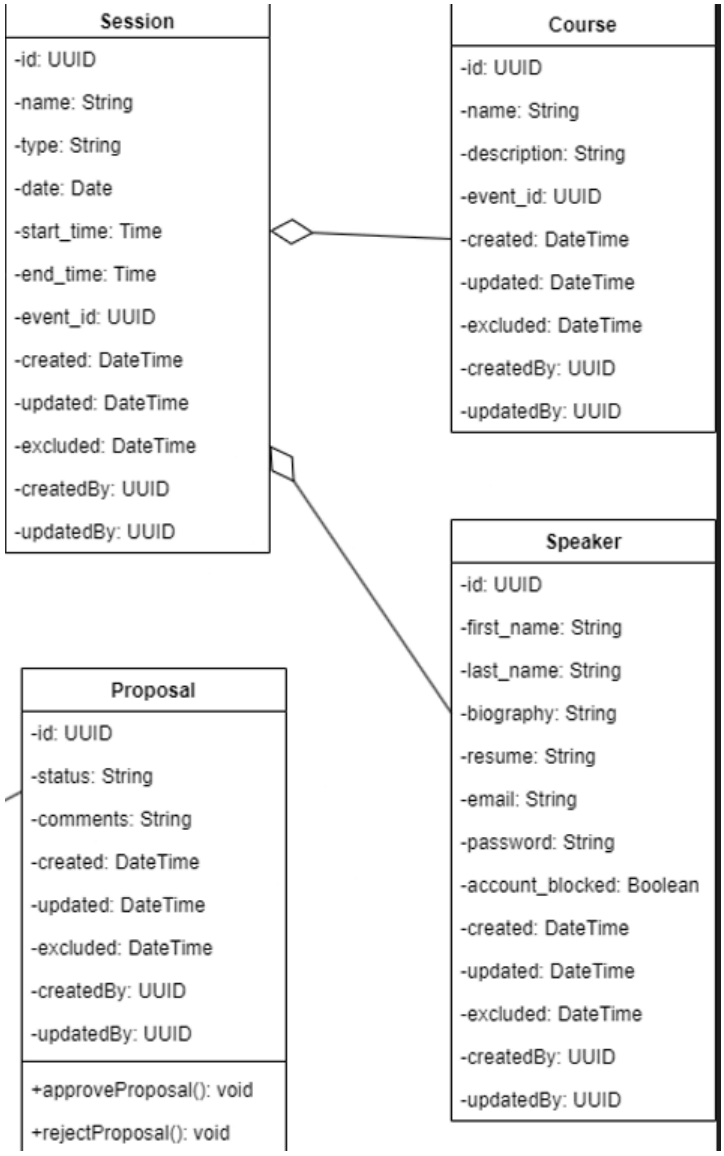
Fonte: Autoria Própria

Apêndice K - Fragmento 1 Diagrama de Classe



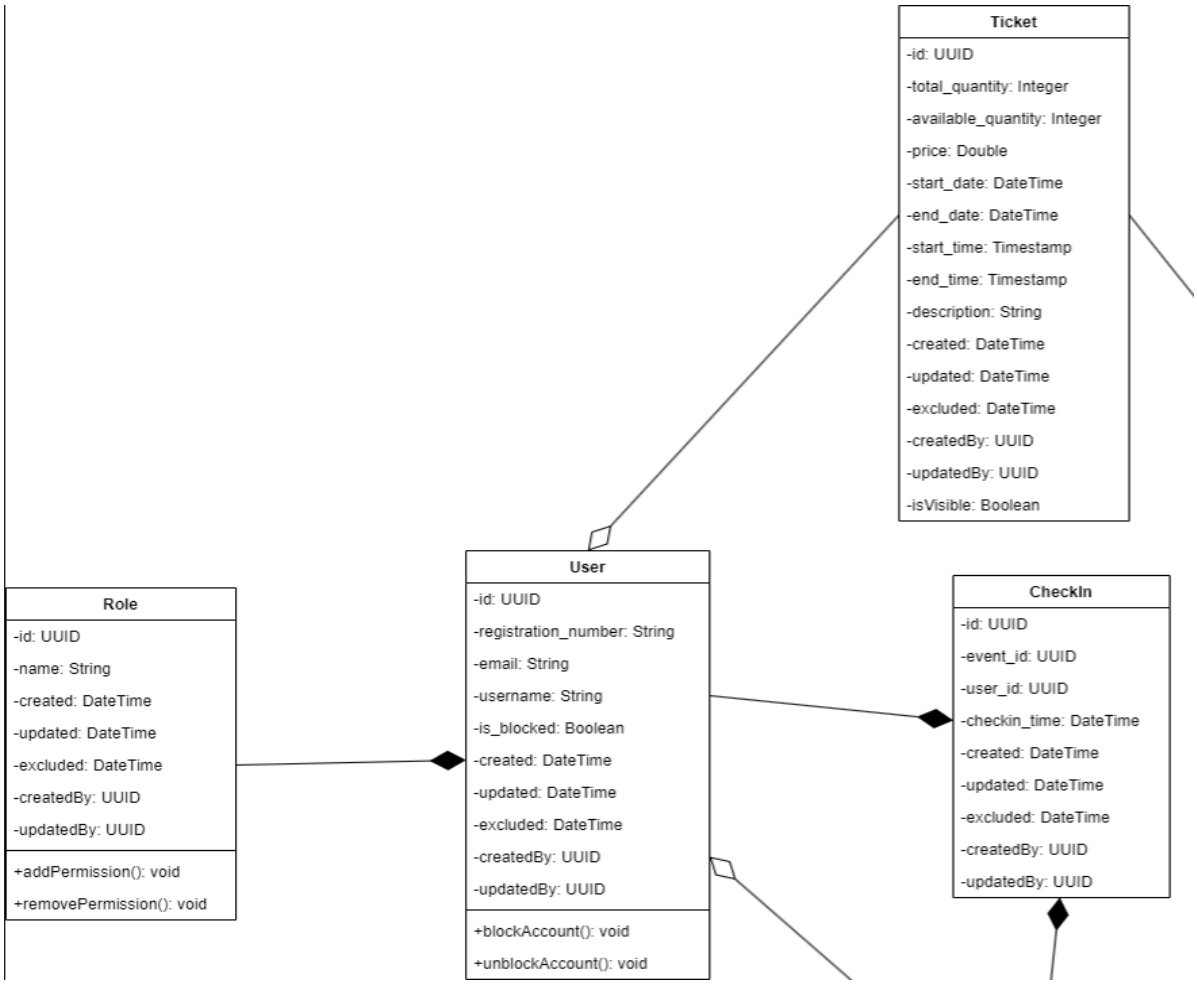
Fonte: Autoria própria

Apêndice L - Fragmento 2 Diagrama de Classe



Fonte: Autoria própria

Apêndice M - Fragmento 3 Diagrama de Classe



Fonte: Autoria própria

Apêndice N - Imagem da capa