UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Rezende Machado

Plataforma de Agendamento de Espaços Esportivos da UFU

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Rezende Machado

Plataforma de Agendamento de Espaços Esportivos da UFU

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

Orientador: Maria Adriana Vidigal de Lima

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2025

Gabriel Rezende Machado

Plataforma de Agendamento de Espaços Esportivos da UFU

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

Trabalho aprovado. Uberlândia, Brasil, 09 de setembro de 2025:

Maria Adriana Vidigal de Lima
Orientadora

Claudiney Ramos Tinoco

Bruno Augusto Nassif Travençolo

Uberlândia, Brasil
2025

Dedico este trabalho à minha família, amigos e professores, os quais me apoiaram ao longo da caminhada acadêmica e me proporcionaram ensinamentos valiosos para meu crescimento pessoal e profissional, ajudando a transformar ignorância em conhecimento e a me tornar uma pessoa melhor e mais sábia.

Agradecimentos

Primeiramente, agradeço a Deus, à minha família e à minha namorada, Júlia Rabelo Cruvinel, pelo apoio constante ao longo da jornada acadêmica, por me guiarem neste caminho de aprendizado e por me darem forças nos momentos de dificuldade, sempre demonstrando amor e dedicação. Espero poder retribuir todo esse carinho ao longo da minha vida.

Agradeço também ao corpo docente da FACOM por despertar em mim o interesse pela área e oferecer uma base sólida de conhecimento. Em especial, sou grato à minha orientadora, professora Maria Adriana Vidigal de Lima, e ao meu orientador de estágio, professor Humberto Luiz Razente, pelo apoio e pela orientação durante esta etapa.

Expresso ainda minha gratidão aos amigos, especialmente Giovanna e Guilherme, por estarem sempre ao meu lado no dia a dia da graduação, e à Associação Atlética da Computação, pela integração nos esportes e eventos acadêmicos. Foi uma honra representar a atlética jogando tênis de mesa durante os quatro anos de graduação.

Por fim, agradeço ao Igor Moraes Mariano, da DIESU, e ao Davi, pela disposição em avaliar e direcionar o desenvolvimento deste projeto, fornecendo a base necessária para a evolução da ideia.

Resumo

No contexto universitário brasileiro, as atividades esportivas desempenham papel fundamental na promoção da socialização e da qualidade de vida dos estudantes. Apesar da relevância, muitas universidades enfrentam dificuldades na organização de reservas dos espaços físicos, o que resulta em processos burocráticos e ineficientes. Exemplo dessa realidade foi identificado na Universidade Federal de Campina Grande (UFCG), onde se verificou a necessidade de digitalização do agendamento de espaços esportivos. De maneira semelhante, a Universidade Federal de Uberlândia (UFU) ainda realiza reservas de ginásios de forma manual e onerosa, sobrecarregando gestores e dificultando o acesso da comunidade acadêmica. Diante desse cenário, este trabalho dá continuidade à proposta de Lacerda (2024), que propõe a criação de uma plataforma digital de agendamentos como alternativa para superar a burocracia e ampliar o acesso, realizando uma modelagem inicial de requisitos e um protótipo visual. Assim, o objetivo é desenvolver e validar uma primeira versão funcional da plataforma web para o agendamento de espaços esportivos da UFU, buscando tornar o processo mais ágil, automatizado e eficiente. A solução considera a integração com os sistemas institucionais presentes, garantindo acesso restrito a alunos, professores e servidores, além de seguir as diretrizes do *Design System* do Governo Federal (DSGov), assegurando usabilidade e acessibilidade. A iniciativa contribui para reduzir a sobrecarga administrativa, otimizar recursos institucionais e democratizar o acesso às práticas esportivas na universidade.

Palavras-chave: Ineficiência, Agendamento, Espaços Esportivos, Universidades, Plataforma.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API *Interface* de Programação de Aplicações (*Application Programming Interface*)

BPMN Modelo e Notação de Processos de Negócios (*Business Process Model and Notation*)

CORS Compartilhamento de Recursos entre Origens (*Cross-Origin Resource Sharing*)

CSS Folhas de Estilo em Cascata (*Cascading Style Sheets*)

CTIC Centro de Tecnologia da Informação e Comunicação

DDD Projeto Orientado ao Domínio (*Domain-Driven Design*)

DIESU Divisão de Esporte e Lazer Universitário

DSGov Padrão Digital de Governo (*Design System do Governo Federal*)

ES Engenharia de Software (*Software Engineering*)

JWT Padrão aberto para criação e verificação de *tokens* baseados em JSON (*JSON Web Token*)

MVP Produto Mínimo Viável (*Minimum Viable Product*)

OMG Grupo de Gerenciamento de Objetos (*Object Management Group*)

REST Transferência Representacional de Estado (*Representational State Transfer*)

SOLID Conjunto de cinco princípios da programação orientada a objetos: *Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation e Dependency Inversion*

SPA Aplicação de Página Única (*Single Page Application*)

SSO Autenticação Única (*Single Sign-On*)

UFAM Universidade Federal do Amazonas

UFCG Universidade Federal de Campina Grande

UFU Universidade Federal de Uberlândia

UX Experiência do Usuário (*User Experience*)

Sumário

1 Introdução

Diversas instituições de ensino superior no Brasil promovem atividades esportivas como forma de incentivar a socialização, a saúde e a qualidade de vida dos estudantes. Entretanto, mesmo entre universidades que dispõem de espaços físicos para essas práticas, ainda persistem problemas relacionados à organização e à gestão dos agendamentos. Em estudo realizado na Universidade Federal de Campina Grande (UFCG), Lucena (2023) identificaram falhas no processo de marcação dos espaços esportivos, destacando a falta de eficiência e a necessidade de soluções digitais para otimizar a gestão.

Na Universidade Federal de Uberlândia (UFU), o cenário é semelhante. O processo atual de reserva de ginásios e quadras ainda ocorre de forma burocrática e lenta, exigindo esforço considerável do setor responsável e gerando insatisfação para a comunidade acadêmica. Essa realidade evidencia a necessidade de modernização do processo, de modo a reduzir a sobrecarga administrativa e oferecer aos usuários uma experiência mais ágil.

Nesse sentido, o desenvolvimento de um sistema web de agendamento se apresenta como uma solução viável e necessária, conforme proposto por Lacerda (2024). Plataformas digitais bem projetadas favorecem a automação e a organização dos processos, tornando-os mais eficientes e integrados. Além disso, a adoção do Padrão Digital de Governo (*Design System do Governo Federal*) (DSGov) contribui para a padronização das *interfaces*, assegurando o cumprimento das diretrizes de usabilidade e acessibilidade nos serviços públicos digitais. Assim, o presente trabalho busca não apenas suprir a demanda institucional da UFU, mas também contribuir para a transformação digital de processos acadêmicos no ensino superior brasileiro.

1.1 Relevância

Como mencionado anteriormente, a prática esportiva dentro das universidades traz diversos benefícios para os seus praticantes, como estudado por Dantas et al. (2018) na Universidade Federal do Amazonas (UFAM). Os autores observaram que a participação dos estudantes de medicina na Atlética Acadêmica trouxe benefícios significativos, tanto para a saúde física quanto para a saúde mental. Essa experiência contribuiu para o alívio do estresse causado pela graduação, promoveu maior interação entre os alunos, fortalecendo laços de amizade e possibilitou a descoberta de novas habilidades. Como resultado, os participantes relataram uma sensação de plenitude, satisfação pessoal e motivação para manter a prática esportiva dentro e fora do ambiente acadêmico. Nesse sentido, é válido o esforço para democratizar e facilitar o acesso à prática esportiva dentro das universidades, incluindo a criação da plataforma para facilitar o processo atual de agendamento de

espaços esportivos da UFU. Além disso, é importante ressaltar a necessidade de evolução digital nas universidades para atender ao DSGov, que tem como intuito unificar os canais digitais governamentais, como também observado por Lacerda (2024).

1.2 Objetivo

O presente trabalho tem como objetivo principal o desenvolvimento e implementação de um sistema *web* para o agendamento de espaços esportivos na UFU. O sistema buscará otimizar o processo de marcação, tornando-o mais acessível, eficiente e automatizado, reduzindo a carga administrativa sobre os responsáveis pela gestão desses espaços. Para alcançar esse objetivo, pretende-se criar uma plataforma intuitiva e analisar a possibilidade de integração com o Autenticação Única (*Single Sign-On*) (SSO) da UFU, o qual garantiria que apenas alunos, professores e servidores autenticados pudessem utilizar o serviço. Além disso, o sistema será desenvolvido seguindo as diretrizes do DSGov, visando alinhamento com os padrões governamentais de usabilidade e acessibilidade. Outro aspecto essencial do trabalho será a validação da eficiência do sistema na prática, avaliando se a solução proposta de fato reduz os desafios enfrentados atualmente no agendamento dos espaços esportivos da UFU. Com isso, espera-se não apenas facilitar o processo para os usuários, mas também fornecer uma base tecnológica que possa ser expandida ou adaptada para outras necessidades institucionais, otimizando o fluxo e o tempo de setores da faculdade.

1.3 Método do trabalho

Para o desenvolvimento do sistema de agendamento de espaços esportivos da UFU, serão utilizados software e *frameworks* alinhados às tecnologias já presentes na universidade, além de considerar fatores como requisitos funcionais, boas práticas e eficiência para definição da arquitetura, componentes e suas integrações, facilitando sua publicação dentro de aplicações institucionais e garantindo que a plataforma atinja os objetivos propostos de otimização do processo atual, manutenção simplificada e necessidades dos usuários atendidas.

Nesse sentido, como a plataforma necessita de um sistema robusto, consistente e de fácil manutenção, é ideal o modelo cliente-servidor, como atestado por Tanenbaum e Steen (2006), que enfatiza ser apropriado para sistemas de serviços centralizados e consumidos por diferentes clientes, promovendo a modularidade e a manutenção facilitada. Nesse contexto, a plataforma foi dividida em três componentes com funcionalidades distintas:

1. **Cliente (*Frontend*)**: responsável pela *interface* visual, que consome dados por meio de requisições ao servidor;

2. **Servidor (*Backend*)**: componente lógico que centraliza as regras de agendamento e o tratamento de erros, com a finalidade de garantir consistência e integrar com a entidade que armazena os dados;

3. **Banco de dados**: entidade responsável por armazenar as informações necessárias.

O servidor será desenvolvido em Java, utilizando o *framework* Spring, que adota o paradigma de programação orientada a objetos, estruturado em classes para melhor modularização (KENDAL, 2009). A linguagem e o *framework* oferecem alta escalabilidade e segurança, além de um ambiente de desenvolvimento ágil e bem documentado. O Spring destaca-se ainda pela confiabilidade, robustez e facilidade de integração com bancos de dados modernos e serviços externos — características amplamente reconhecidas por Anwar e Bairstow (2023). Com isso, para armazenar esses dados orquestrados pelo servidor, será utilizado o banco de dados *open-source* PostgreSQL (não exige licença paga), reconhecido por sua confiabilidade, suporte contínuo, transações complexas e estrutura de banco relacional com tipos de dados complexos como observado por Juba, Vannahme e Volkov (2015), sendo um ponto essencial para economizar no orçamento limitado de instituições públicas.

Desse modo, o cliente (componente visual) será desenvolvido com o *framework* Angular, escolhido por ser uma solução robusta, madura e escalável, especialmente adequada ao desenvolvimento de Aplicação de Página Única (*Single Page Application*)s (SPAs) complexas e de grande porte em contextos institucionais. Conforme destacado por Uluca (2024), o Angular proporciona excelente integração com o servidor e suporte consistente a práticas de arquitetura modular. Além disso, oferece compatibilidade com o DSGov (BRASIL, 2024), garantindo acessibilidade, padronização visual e conformidade com diretrizes de usabilidade em serviços públicos digitais. Com o intuito de melhorar o desempenho e otimizar o uso da banda, a comunicação entre cliente e servidor será feita por meio de requisições via protocolo HTTP, empregando estratégias de cache para reduzir a latência e o tráfego de rede (TANENBAUM; STEEN, 2006). Além disso, será avaliada a viabilidade de integração com o SSO da UFU, possibilitando que apenas usuários autenticados (alunos, professores e servidores) tenham acesso ao sistema e usufruam de um sistema de segurança centralizado, o que facilita a gestão pela instituição, conforme abordado por Chalandar, Darvish e Rahmani (2007). Caso a integração não seja possível, serão consideradas alternativas para garantir a segurança e o controle de acesso.

2 Revisão Bibliográfica

Atividades esportivas são bastante populares dentro das universidades, pois proporcionam não só lazer, mas também benefícios para a saúde e interação social. No entanto, a gestão dos espaços esportivos ainda enfrenta desafios, principalmente quando o agendamento depende de processos manuais, como ligações telefônicas ou registros físicos. Isso pode gerar conflitos de horários, falta de organização e dificuldades para alunos.

Problemas semelhantes já foram identificados em outras universidades, como na UFCG, onde a falta de automação prejudicava a eficiência do processo (LUCENA, 2023). Na UFU, o cenário não é diferente, e é nesse contexto que este trabalho propõe o desenvolvimento de um sistema *web* para otimizar o processo de agendamento, tornando-o mais acessível e eficiente. O trabalho de Lacerda (2024) serviu como referência teórica para a modelagem do sistema.

2.1 Trabalhos Relacionados

Diversos trabalhos abordaram a problemática do agendamento e gestão de espaços esportivos em universidades, propondo soluções para otimizar esse processo. Nesse contexto, há 4 trabalhos que se relacionam fortemente com este artigo:

1. Sistema de agendamentos de horários para os espaços esportivos na UFU - modelagem e prototipagem, 2024, Lacerda, monografia (LACERDA, 2024);
2. Evolução do sistema de agendamento de horários do complexo esportivo da UFCG-SAHCE-UFCG, 2023, Lucena, monografia (LUCENA, 2023);
3. SPACESYNC: Agendamentos de Espaços, 2023, Pieri, monografia (PIERI et al., 2023);
4. Inserção dos estudantes de Medicina nas Atléticas Acadêmicas: promoção de saúde física e mental por meio do esporte, da cultura e da arte, 2018, Dantas, 13º Congresso Internacional Rede Unida, conferência (DANTAS et al., 2018).

O trabalho de Lacerda (2024) apresenta a modelagem e a prototipagem do *frontend* de um sistema de agendamento de horários para os espaços esportivos da UFU, tendo como principal objetivo automatizar o processo de reserva, que antes era realizado de forma manual e onerosa para alunos e pessoal responsável, otimizando o uso dos espaços, facilitando o processo, reduzindo conflitos e tempo gasto. Além disso, o autor utilizou

métodos de engenharia de requisitos e modelagem de processos para levantar as necessidades dos usuários e desenhar soluções viáveis, retratando a *interface* gráfica da aplicação de acordo com o *Design System* do governo brasileiro (BRASIL, 2024). Assim, o trabalho serviu como uma base teórica importante para o desenvolvimento do presente projeto, especialmente no que se refere à estruturação do sistema e à definição de funcionalidades esperadas, *design* do projeto e arquitetura do sistema, alinhando-se com a proposta de melhorar a experiência dos usuários com uma *interface* acessível e responsiva.

Ademais, a pesquisa de Lucena (2023) investiga a evolução do sistema de agendamento no complexo esportivo da Universidade Federal de Campina Grande (UFCG). O estudo revela que a ausência de digitalização nos processos impactava negativamente a eficiência do serviço, gerando atrasos, sobrecargas administrativas e insatisfação entre os usuários, sendo mais um exemplo de que o problema é comum em outras instituições e pode ser resolvido via implementação de um sistema de software. Por meio da análise das versões anteriores do sistema e da proposição de melhorias incrementais, os autores demonstraram como a transição gradual para soluções digitais pode trazer benefícios significativos. Dessa forma, este trabalho dialoga diretamente com o projeto atual, ao evidenciar os desafios similares enfrentados em outras instituições e apontar caminhos para a construção de um sistema mais robusto e escalável, sendo um exemplo de sucesso do modelo apresentado para uma dificuldade análoga enfrentada.

Além disso, o artigo de Pieri et al. (2023) descreve o desenvolvimento do SPACESYNC, uma aplicação para agendamento de espaços em geral, com foco na usabilidade e na flexibilidade, tendo como objetivo facilitar o gerenciamento de reservas por meio de uma plataforma digital intuitiva. A equipe utilizou princípios de *design* centrado no usuário e adotou práticas modernas de desenvolvimento web para garantir a funcionalidade e a adaptabilidade do sistema, desenvolvendo diagramas e modelagem do sistema feitas por uma equipe, tendo uma solução robusta e genérica. O presente projeto se beneficia dessa abordagem ao incorporar elementos de usabilidade e integração com padrões modernos de *design*, como o DSGov (BRASIL, 2024), reforçando a importância de pensar na experiência do usuário durante o processo de desenvolvimento e servindo de exemplo de modelagem de banco de dados e sistema incrementável ao longo do tempo, o qual detém vasta aceitação do projeto e ampla utilização.

Por fim, o estudo de Dantas et al. (2018) — apresentado nos anais do 13º Congresso Internacional Rede Unida — traz uma contribuição significativa ao discutir o papel das Atléticas Acadêmicas na promoção da saúde física e mental dos estudantes no ambiente universitário, tendo como exemplo a Atlética de Medicina da Universidade Federal do Amazonas (UFAM). O trabalho tem como objetivo analisar as atividades promovidas pelas atléticas e sua influência no bem-estar e na integração social dos alunos. Para isso, os autores adotaram uma abordagem qualitativa, por meio da análise da participação

dos estudantes, variavelmente em torno de 15 membros, nessas entidades atléticas e de bateria musical. Os resultados indicam que a atuação das atléticas vai além do incentivo ao esporte: elas promovem pertencimento, melhoram a saúde emocional dos estudantes e fortalecem os laços comunitários dentro da universidade. Embora o estudo não se concentre diretamente no desenvolvimento de sistemas de informação, ele fornece uma base conceitual importante para o presente projeto ao destacar a relevância das atividades esportivas na formação integral do estudante universitário. Desse modo, o estudo reforça a necessidade de desenvolver mecanismos eficazes de apoio a essas práticas, justificando — assim como no caso apresentado por Lucena (2023) — a importância da digitalização e otimização dos processos de agendamento de espaços esportivos, de modo a promover maior participação estudantil e gerar impacto positivo no ambiente acadêmico.

2.2 Conceitos Fundamentais

Os principais conceitos que fundamentam este trabalho são:

- Engenharia de Software, que orienta o desenvolvimento estruturado e eficiente do sistema;
- Engenharia de Requisitos, que define as funcionalidades e restrições que o sistema deve atender;
- Arquitetura de Software, que estabelece a estrutura do sistema com base no modelo cliente-servidor;
- Modelo e Notação de Processos de Negócios (*Business Process Model and Notation*) (BPMN), utilizada para organizar o fluxo de agendamentos de maneira visual;
- *Design* de *Interfaces* e Usabilidade, garantindo que o sistema siga as diretrizes do DSGov (BRASIL, 2024).

2.2.1 Engenharia de Software

A Engenharia de Software (*Software Engineering*) (ES) envolve práticas que ajudam no desenvolvimento de sistemas confiáveis e escaláveis (SOMMERVILLE, 2011). Ela abrange todo o ciclo de vida do desenvolvimento, do planejamento e análise à manutenção do sistema, passando por etapas essenciais como projeto, implementação e testes.

Para atingir seu objetivo central de criar sistemas confiáveis, eficientes e alinhados às necessidades dos usuários, a ES se apoia em diversos métodos e abordagens. Uma estratégia eficiente baseia-se no conceito de Produto Mínimo Viável (*Minimum Viable Product*) (MVP), que permite criar versões iniciais do sistema para teste e refinamento

antes da versão final (PRESSMAN; MAXIM, 2021). O MVP consiste na versão mais enxuta do produto, com funcionalidades mínimas necessárias para validar suas premissas centrais. Essa prática permite coletar *feedback* rapidamente e direcionar o desenvolvimento subsequente com base em dados concretos, reduzindo significativamente o risco de se construir um produto que não atenda às expectativas dos usuários.

Conforme o projeto evolui a partir do MVP, abordagens de *design* como o Projeto Orientado ao Domínio (*Domain-Driven Design*) (DDD) tornam-se essenciais para garantir uma arquitetura robusta e alinhada ao negócio (KHONONOV, 2021). O DDD coloca o domínio do negócio e sua lógica no centro do processo de desenvolvimento, promovendo uma comunicação contínua entre especialistas do negócio e desenvolvedores. O resultado é um software que não apenas atende aos requisitos técnicos, mas reflete com precisão as complexidades e regras do domínio para o qual foi pensado.

A aplicação combinada de práticas como MVP e DDD ajuda a minimizar riscos, assegurar a durabilidade do software e garantir conformidade com as especificações. Além do aspecto técnico, a ES também valoriza a documentação adequada, a comunicação efetiva com os stakeholders e o controle de qualidade. Esses elementos são fundamentais para assegurar que o software final seja seguro, funcional e plenamente adequado ao seu propósito.

Considerando o desenvolvimento do sistema de agendamento, será adotada uma abordagem iterativa, isto é, a cada nova versão, melhorias serão implementadas com base no *feedback* dos usuários. Isso ajudará a garantir que o sistema atenda às necessidades reais da comunidade acadêmica, com entregas graduais até sua conclusão.

2.2.2 Engenharia de Requisitos

A Engenharia de Requisitos é fundamental para garantir que o sistema tenha funcionalidades bem definidas e esteja alinhado às necessidades dos usuários (SOMMERVILLE, 2011). Os requisitos se dividem em:

- Funcionais: descrevem as ações que o sistema precisa executar (por exemplo, permitir que usuários reservem horários nos espaços esportivos);
- Não funcionais: determinam restrições, como segurança e desempenho (BOURQUE; FAIRLEY, 2014).

No caso deste projeto, os requisitos foram levantados a partir da análise do fluxo de agendamento atual da UFU e da proposta de Lacerda (2024), garantindo que o sistema seja funcional e alinhado ao DSGov (BRASIL, 2024).

2.2.3 *Domain-Driven Design* e a Importância do SOLID

O DDD foca na criação de sistemas baseados no entendimento profundo do domínio de negócios, facilitando a construção de modelos que refletem a complexidade e as necessidades da organização (KHONONOV, 2021). A principal proposta do DDD é desenvolver sistemas modulares e flexíveis que evoluam com o domínio, aplicando práticas como os princípios Conjunto de cinco princípios da programação orientada a objetos: *Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation e Dependency Inversion* (SOLID), que garantem código coeso e de fácil manutenção (SINGH; HASSAN, 2015).

Os princípios SOLID configuram boas práticas de *design*, orientado a objetos, com o objetivo de tornar o código mais flexível, reutilizável e fácil de manter. A sigla SOLID representa os cinco princípios fundamentais:

- S - *Single Responsibility* ou Princípio da Responsabilidade Única: cada classe deve ter apenas uma responsabilidade, ou seja, um único motivo para mudar. Esse princípio reduz o acoplamento e facilita a manutenção do código.
- O - *Open/Closed* ou Princípio do Aberto/Fechado: o software deve estar aberto para extensão, mas fechado para modificação. Deve ser possível adicionar novas funcionalidades sem alterar o código existente, por meio de abstrações (herança ou *interfaces*).
- L - *Liskov Substitution* ou Princípio da Substituição de Liskov: as subclasses devem poder substituir suas superclasses sem alterar o comportamento esperado do programa.
- I - *Interface Segregation* ou Princípio da Segregação de *Interfaces*: Nenhuma classe deve ser forçada a implementar *interfaces* que não utiliza. É melhor ter várias interfaces específicas e coesas do que uma única *interface* genérica e extensa.
- D - *Dependency Inversion* ou Princípio da Inversão de Dependência: os módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações. Este princípio permite o desacoplamento entre componentes e facilita a testabilidade e manutenção do sistema.

O trabalho de Fowler (2002) mostra que o uso de padrões contribui para a criação de sistemas robustos. O artigo destaca o *Chain of Responsibility* e a Arquitetura Hexagonal (*Ports and Adapters*), que se alinham bem com os conceitos de DDD, promovendo a modularidade e o desacoplamento das camadas de infraestrutura do núcleo de domínio, permitindo que o sistema seja facilmente adaptado a novas necessidades.

Portanto, ao adotar o DDD e aplicar os princípios SOLID, é possível construir sistemas que sejam não apenas alinhados ao negócio, mas também sustentáveis e de fácil manutenção ao longo do tempo. O SOLID fornece a base teórica para um código de qualidade. A Arquitetura Hexagonal usa esses princípios (especialmente a Inversão de Dependência) para criar uma estrutura desacoplada. Por sua vez, o *Chain of Responsibility* é um padrão de projeto comportamental que permite que uma requisição seja passada ao longo de uma cadeia de objetos manipuladores. Dentro de uma arquitetura Hexagonal, ele organiza de forma eficaz os comportamentos da aplicação, e sua implementação eficaz baseia-se nos princípios SOLID. Em consequência, eles formam um ecossistema coeso para o desenvolvimento de software.

2.2.4 Arquitetura de Software

O sistema de agendamento será baseado na arquitetura cliente-servidor, onde um servidor central gerenciará as requisições feitas pelos clientes, acarretando em um controle de acesso facilitado, melhorando a segurança e permitindo escalabilidade (TANENBAUM; STEEN, 2006).

O *backend* será responsável pelo processamento de dados e armazenamento das informações, enquanto o *frontend* oferecerá uma *interface* intuitiva para usuários realizarem suas reservas. A escolha desse modelo foi baseada no estudo de Lacerda (2024), que demonstrou sua viabilidade para um ambiente universitário. Para garantir eficiência e compatibilidade com outros sistemas institucionais, o sistema usará *Interface* de Programação de Aplicações (*Application Programming Interface*) (API) com Transferência Representacional de Estado (*Representational State Transfer*) (REST), facilitando a comunicação entre diferentes serviços.

2.3 Modelagem de Processos com BPMN

A Modelagem de Processos de Negócio (BPMN) é uma notação padronizada utilizada para representar de maneira gráfica e compreensível os fluxos de trabalho, permitindo que diferentes atores — técnicos ou não — compreendam e analisem os processos. Essa notação foi desenvolvida pelo Grupo de Gerenciamento de Objetos (*Object Management Group*) (OMG) e consolidou-se como padrão de mercado para a documentação de processos organizacionais (Object Management Group (OMG), 2013).

Segundo o livro de Freund e Rücker (2019), o BPMN possibilita traduzir fluxos complexos em diagramas intuitivos.Entre os principais elementos destacam-se:

- **Piscinas (*Pools*) e Raias (*Lanes*):** delimitam os atores ou áreas responsáveis dentro do processo. No fluxo de agendamento da UFU, a piscina representa o pro-

cesso como um todo, e as raias dividem responsabilidades entre a Divisão de Esporte e Lazer Universitário (DIESU) e os usuários

- **Atividades (*Activities*):** representadas por retângulos arredondados, indicam tarefas realizadas, como "Solicitar agendamento" ou "Registrar horário".
- **Eventos (*Events*):** círculos que indicam pontos de início, intermediários ou fim do processo, por exemplo, o evento de "Início da solicitação" e o "Fim do agendamento concluído".
- ***Gateways*:** representados por losangos, são utilizados para modelar pontos de decisão ou ramificação do processo. No caso da UFU, eles determinam se a solicitação pode ser atendida ou se deve ser rejeitada.
- **Objetos de Dados (*Data Objects*):** ícones que indicam informações utilizadas ou geradas no processo, como as planilhas de horários preenchidas pela DIESU.
- **Fluxos de Sequência (*Sequence Flows*):** setas que conectam os elementos, indicando a ordem de execução das atividades.

Essa combinação fornece uma linguagem comum entre áreas de negócio e equipes técnicas, reduzindo ambiguidades e aumentando a clareza no levantamento de requisitos.

Neste projeto, o BPMN será utilizado para mapear o fluxo de agendamentos e definir cada etapa do processo, garantindo que a transição do sistema manual para o digital seja clara e eficiente. O trabalho de Lacerda (2024) já apresentou uma modelagem inicial desse fluxo, que será ajustada conforme as necessidades específicas da UFU para uma primeira versão funcional. Na Figura 1, pode-se analisar os elementos básicos utilizados para notação e sua representação visual:

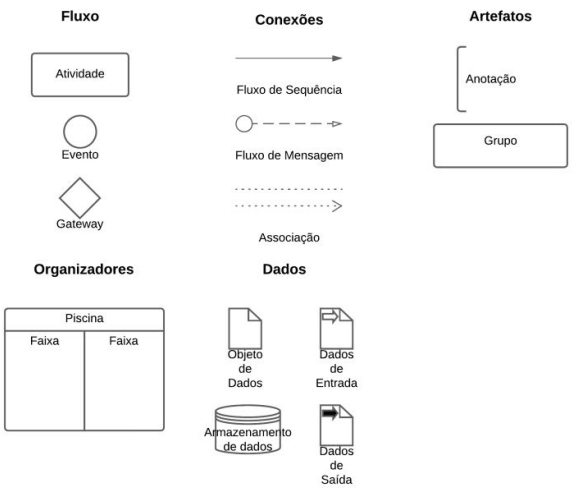

Figura 1 – Os elementos básicos do BPMN. Fonte: Lacerda (2024).

2.4 Design de *Interfaces* e Usabilidade

A Experiência do Usuário (*User Experience*) (UX) é um fator essencial no desenvolvimento de sistemas, e para garantir acessibilidade e padronização, este projeto seguirá as diretrizes do DSGov (BRASIL, 2024). Essa padronização busca tornar a navegação mais intuitiva, especialmente para usuários já familiarizados com outros sistemas institucionais.

Seguindo as práticas de *design* de *interfaces*, o sistema será desenvolvido com um layout responsivo e de fácil usabilidade, garantindo que qualquer usuário possa acessar e utilizar a plataforma sem dificuldades.

3 Desenvolvimento

A construção do artefato proposto inicia-se por uma análise aprofundada do processo vigente e da natureza do problema que ele evidencia. Assim, antes de escolher tecnologias ou definir a arquitetura, é essencial compreender como o agendamento ocorre hoje, quais atores participam, que dados circulam, quais regras e restrições condicionam as decisões e onde surgem atrasos, retrabalhos e inconsistências. Esse diagnóstico gera três insumos centrais para o desenvolvimento: o mapeamento fiel do processo de agendamentos de espaços esportivos atual, a consolidação dos principais pontos de dor e riscos operacionais e a lista priorizada de requisitos para o MVP, já acompanhada de critérios objetivos de aceitação. Com essa base, o projeto avança com escopo realista, alinhamento institucional e foco nas funcionalidades que efetivamente reduzem a fricção do processo.

3.1 Processo Atual

O processo de agendamento de horários para os espaços esportivos da UFU ainda é realizado de maneira predominantemente manual, centralizado na DIESU. A comunicação entre usuários e a administração ocorre por meio de ligações telefônicas, sendo os dados registrados em planilhas eletrônicas para controle dos horários disponíveis. Esse modelo, embora atenda minimamente às demandas por meio do esforço constante da DIESU, revela-se burocrático, restritivo e suscetível a falhas humanas, tornando-se oneroso e passível de inconsistências.

Na prática, a dinâmica atual exige que os interessados realizem contato durante períodos específicos do dia. Em uma primeira tentativa de reserva, o usuário deve fornecer as seguintes informações para fins de cadastro:

- Nome
- Curso
- Telefone
- Matrícula

Após o cadastro inicial, ou caso o usuário já esteja previamente registrado, outras informações são solicitadas para a efetivação do agendamento, tais como:

- Ginásio
- Dia

- Horário
- Quantidade de pessoas

Conforme observado por Lacerda (2024), e representado na Figura 2, o fluxo vigente restringe a autonomia do usuário, que não possui acesso prévio à disponibilidade dos horários em cada espaço esportivo. Além disso, o contato deve ser realizado em períodos específicos do dia, o que gera sobrecarga à equipe administrativa responsável pelas reservas. Dessa forma, constata-se que, embora funcional, o processo atual não acompanha a crescente demanda da comunidade acadêmica, sendo de difícil administração e marcado pela falta de transparência. Esse cenário constitui um dos principais motivadores para o desenvolvimento de uma solução digital mais eficiente.

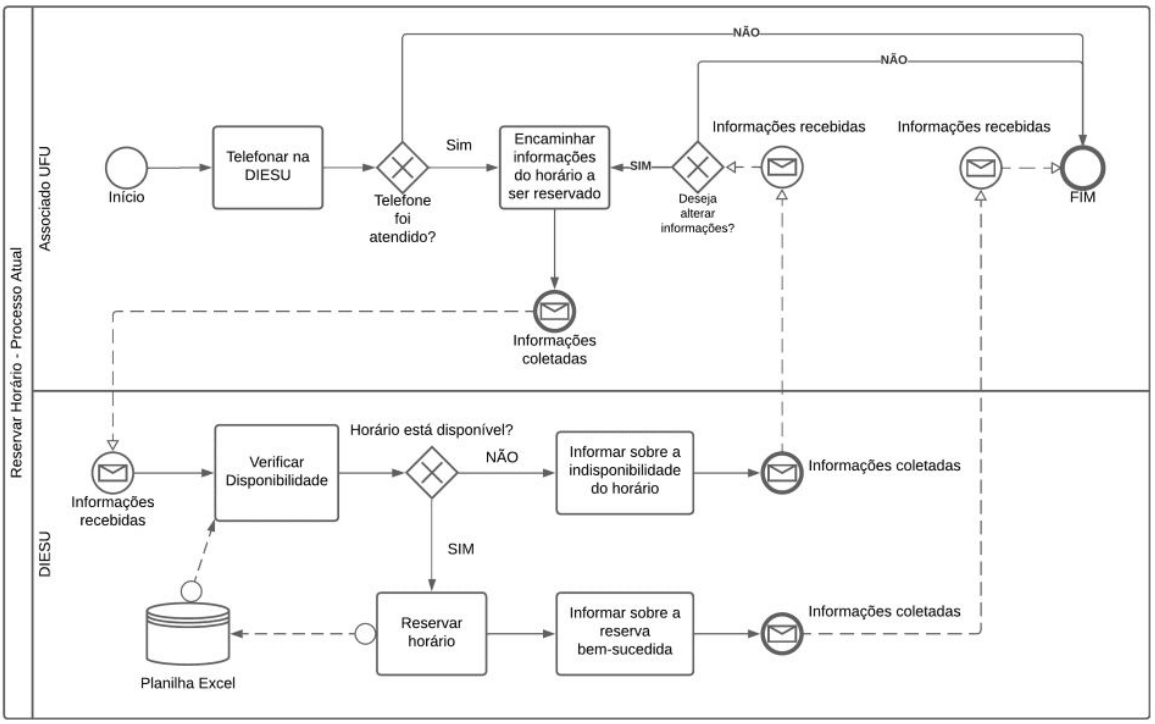

Figura 2 – Fluxo atual do processo de agendamento dos espaços esportivos da UFU. Fonte: Lacerda (2024)

3.2 Etapas do Método de Pesquisa

De acordo com o método proposto por Wazlawick (2009), as etapas abaixo foram definidas para nortear o desenvolvimento do projeto de sistema de agendamento de espaços esportivos na UFU para a otimização do processo atual, o qual nomeei de Schedule-UFU (componentes visuais) e Schedule-Core (servidor):

1. **Levantamento e estudo de trabalhos relacionados de sistemas e eficiência identificada**
   Identificar e pesquisar trabalhos de sistemas de agendamentos dentro das universidades, tal como requisitos funcionais, boas práticas, eficiência, limitações e exemplos de sucesso e falha, com taxa de efetividade no uso para uma base sólida e embasada para o desenvolvimento do projeto para que tenha a maior taxa de sucesso possível na aderência do usuário ao mesmo tempo que otimiza os custos, manutenção e otimiza o tempo gasto com essa tarefa.

2. **Levantamento de requisitos com base nos usuários da UFU**
   Os requisitos foram derivados das dores e oportunidades mapeadas por Lacerda (2024) e refinados por meio de conversas exploratórias com a equipe da DIESU. Não foram aplicados instrumentos formais (entrevistas estruturadas ou questionários); as interações tiveram caráter qualitativo e pragmático, focadas em confirmar a aderência ao cenário atual, levantar restrições operacionais e priorizar funcionalidades essenciais do MVP. O resultado foi uma lista priorizada de requisitos funcionais e não funcionais, alinhada às demandas institucionais e às diretrizes do DSGov. Em iterações futuras, técnicas formais poderão complementar e quantificar essas evidências.

3. **Definição da arquitetura do sistema e tecnologias**
   Definição das tecnologias utilizadas para desenvolvimento de uma primeira versão funcional visando possível integração posterior ao sistema da UFU e publicação em seus servidores. Ademais, modelar arquitetura no sistema diante de requisitos apresentados, com base teórica e justificativa dessas tecnologias e arquitetura do sistema para que seja robusto, eficiente e incrementável.

4. **Desenvolvimento da primeira versão funcional (MVP)**
   Implementar as funcionalidades principais, com maior prioridade elencadas com base nas entrevistas e requisitos levantados, visando um sistema com geração de valor para usuários e administradores, além da possibilidade de validação de funcionamento e uso, como visualização de horários disponíveis, reserva, cancelamento, sendo todas funcionalidades seguindo padrões digitais do governo e da instituição, no caso o DSGov (LACERDA, 2024) e Centro de Tecnologia da Informação e Comunicação (CTIC), para adequação para posterior integração com os sistemas.

5. **Validação do sistema com usuários reais**
   Realizar testes com um grupo de usuários e servidores da UFU, coletando *feedback* sobre usabilidade, funcionalidades e desempenho do sistema, com o intuito de mapear próximos entregáveis e requisitos propostos para primeira versão e deixar aberto para próximas evoluções.

6. **Documentação e redação final do trabalho**
   Com isso, será necessário registrar todas as etapas do projeto, decisões de projeto, dificuldades encontradas e soluções adotadas para elaboração do artigo. Além do mais, elaborar conclusões e resultados dos testes no sistema para promover sugestões para trabalhos futuros relacionados a esse projeto, deixando uma primeira versão base a qual pode ser aproveitado pela instituição e utilizado posteriormente.

3.3 Levantamento de requisitos

O levantamento de requisitos consiste em identificar, negociar e documentar o que o sistema deve fazer e sob quais restrições deve operar, servindo de base para todas as decisões de projeto subsequentes (SOMMERVILLE, 2011). Dado o objetivo de entregar uma primeira versão funcional, adotou-se uma priorização pragmática, privilegiando fluxos essenciais ao uso cotidiano e requisitos não funcionais mínimos de segurança, consistência e conformidade visual. O resultado é um conjunto de requisitos rastreáveis, organizado em funcionais e não funcionais, que orienta a arquitetura proposta e abre espaço para iterações futuras com técnicas formais de elicitação.

3.3.1 Requisitos Funcionais

O levantamento de requisitos é uma etapa crucial no desenvolvimento de qualquer sistema, pois define as funcionalidades e características que o software deve ter para atender às necessidades dos usuários e da organização, os quais são os principais intuitos como analisados por Sommerville (2011). No caso do sistema de agendamento de espaços esportivos da UFU, foi fundamental realizar um levantamento minucioso dos requisitos para garantir que a plataforma atendesse às expectativas da comunidade acadêmica e aos objetivos institucionais da universidade. Desse modo, após reuniões com a DIESU para conferir algumas necessidades e embasamento no levantamento de necessidades realizado por Lacerda (2024), foram definidos os requisitos. Abaixo, foram destacados e detalhados os requisitos funcionais de maior prioridade, visando uma primeira versão do sistema.

Requisito	**Detalhes**
RF001 Autenticação	**e autorização de usuários e administradores**
Atores	Usuário, Administrador
Descrição	O sistema deve permitir que qualquer usuário autenticado na UFU (aluno, professor ou servidor) realize o *login* e tenha o acesso de acordo com o previsto, podendo ser usuário ou administrador.
Pré-Condições	O usuário deve possuir um vínculo ativo com a UFU.
Fluxo Normal	a. O usuário acessa portal; b. O usuário insere suas credenciais e autentica no sistema; c. O sistema identifica o tipo de usuário e redireciona para o sistema do Schedule-UFU da página principal.

Tabela 1 – Requisito Funcional 1 - Autorização e autenticação de usuários. Fonte: Do Autor.

Requisito	**Detalhes**
RF002 Criar	**Agendamento**
Atores	Usuário, Administrador
Descrição	O usuário deve poder acessar as datas disponíveis e as tabelas de agendamento para reservar um espaço esportivo disponível em uma data e horário específico.
Pré-Condições	O usuário deve estar autenticado no sistema.
Fluxo Normal	a. O usuário acessa a página de reservas; b. O usuário seleciona o espaço esportivo, a data e o horário desejados; c. O sistema verifica a disponibilidade e confirma a reserva; d. Se o horário já estiver reservado, o sistema exibe uma mensagem de erro de agendamento.

Tabela 2 – Requisito Funcional 2 - Criar Agendamento. Fonte: Do Autor.

Requisito	**Detalhes**
RF003 Cancelamento	**de Reserva pelo Usuário**
Atores	Usuário
Descrição	O usuário pode cancelar uma reserva feita anteriormente.
Pré-Condições	O usuário deve estar autenticado e ser o responsável pela reserva.
Fluxo Normal	a. O usuário acessa a lista de suas reservas; b. O usuário consegue escolher uma reserva e clica em cancelar o agendamento; c. Após o cancelamento, a reserva é cancelada e o horário fica disponível para novos agendamentos; d. Se o usuário tentar cancelar uma reserva já não agendada, o sistema exibe uma mensagem de erro.

Tabela 3 – Requisito Funcional 3 - Cancelamento de Reserva pelo Usuário. Fonte: Do Autor.

Requisito	**Detalhes**
RF004 Cancelamento	**de Reserva pelo Administrador**
Atores	Administrador
Descrição	O administrador pode cancelar qualquer reserva feita por usuários.
Pré-Condições	O administrador deve estar autenticado no sistema.
Fluxo Normal	a. O administrador acessa a lista de reservas; b. O administrador consegue escolher uma reserva e clica em cancelar o agendamento; c. O sistema solicita confirmação do cancelamento; d. Após a confirmação, a reserva é cancelada e o sistema envia um e-mail notificando o usuário.

Tabela 4 – Requisito Funcional 4 - Cancelamento de Reserva pelo Administrador. Fonte: Do Autor.

Requisito	**Detalhes**
RF005 Filtros de	**Visualização de Agendamento**
Atores	Administrador
Descrição	O usuário pode definir ginásio e dia que horários devem ser exibidos.
Pré-Condições	O usuário deve estar autenticado.
Fluxo Normal	a. O usuário acessa a tela inicial do sistema; b. O usuário poderá visualizar reservas do dia; c. O usuário poderá colocar filtros de ginásio que mostrarão os horários marcados e disponíveis do ginásio.
Extensões	a. Se não houver reservas no dia o sistema mostra tela de agendamento sem seleção de ginásio como vazia.

Tabela 5 – Requisito Funcional 5 - Filtros de Visualização de Agendamento. Fonte: Do Autor.

Requisito	**Detalhes**
RF006 Notificações	**Envio de E-mails Automáticos**
Atores	Administrador, Usuário
Descrição	O sistema deve enviar e-mails automáticos para notificar o cancelamento de uma reserva.
Pré-Condições	O administrador ou usuário deve ter cancelado um agendamento no sistema.
Fluxo Normal	a. O administrador ou usuário acessa a seção de gestão de espaços esportivos; b. O administrador ou usuário cancela um agendamento; c. O sistema processa o cancelamento e envia uma notificação por e-mail ao usuário.
Extensões	a. Se o envio do e-mail falhar, o sistema exibe uma mensagem de erro.

Tabela 6 – Requisito Funcional 6 - Notificações Automáticas. Fonte: Do Autor.

Requisito	**Detalhes**
RF007 Restrição de Datas	**Impedir agendamentos em datas restritas**
Atores	Administrador, Usuário
Descrição	O sistema deve impedir agendamentos em datas bloqueadas
Pré-Condições	O administrador e usuário devem estar autenticados para criar restrições e para criar agendamentos, respectivamente.
Fluxo Normal	a. O administrador acessa a seção de restrições; b. Define datas específicas e ginásios; c. O sistema atualiza a disponibilidade dos ginásios automaticamente; d. Usuários que tentarem criar agendamento em datas bloqueadas receberão mensagem de erro.

Tabela 7 – Requisito Funcional 7 - Restrições Datas (indisponibilidades). Fonte: Do Autor.

3.3.2 Requisitos Não Funcionais

Os requisitos não funcionais do sistema guiam as propriedades e a qualidade das funcionalidades. São eles:

- **RNF001 - Autenticação Segura:** Todos os usuários devem estar autenticados no sistema e possuir vínculo com a UFU (relacionado ao RF001).
- **RNF002 - Consistência de Dados:** O sistema deve garantir integridade das reservas, prevenindo duplicidade e conflitos de horários (RF002, RF003, RF004, RF007).
- **RNF003 - Transparência:** O sistema deve enviar e-mails automáticos ao cancelar agendamentos para usuário ficar ciente e em caso de falha no envio, deve exibir mensagem clara ao usuário na interface (RF006).
- **RNF004 - Usabilidade:** A *interface* deve ser intuitiva, permitindo que qualquer usuário interaja facilmente com as funcionalidades do sistema (RF002, RF003, RF004, RF005, RF007).
- **RNF005 - Compatibilidade com padrões institucionais:** A *interface* deve seguir padrões de *design* e acessibilidade compatíveis com o DSGov (BRASIL, 2024), garantindo consistência visual e experiência adequada (RF002, RF005, RF007).

- **RNF006 - Segregação de Papéis e Acesso:** Usuários comuns não podem acessar funcionalidades administrativas; apenas administradores podem gerenciar espaços, regras de prioridade restrições (RF004, RF005, RF007).
- **RNF007 - Modularidade e Continuidade:** O sistema deve ser modular, permitindo que funcionalidades adicionais sejam integradas ou removidas facilmente, garantindo a continuidade e evolução da primeira versão funcional ao longo do tempo, conforme *feedback* de usuários e necessidades da UFU.

Portanto, os requisitos e funcionalidades implementadas foram simplificados, contemplando uma primeira versão funcional do sistema, com foco nos fluxos essenciais de agendamento e cancelamento. Nesse contexto, recursos avançados — como agendamentos recorrentes, regras de prioridade ou notificações mais complexas — não foram incluídos nesta etapa. Ainda assim, o desenho modular (RNF007), inspirado por princípios de DDD e por padrões arquiteturais — como a Arquitetura Hexagonal e a decomposição em componentes — permite incorporar essas evoluções futuramente sem impactar a base já construída. Isso viabiliza, por exemplo, a adição de um módulo de gerenciamento de materiais esportivos sem alterar a estrutura central do sistema.

3.4 Arquitetura e Tecnologias

Esta seção reúne as decisões arquiteturais e tecnológicas do Schedule-UFU. A arquitetura foi pensada para atender aos requisitos funcionais e não funcionais levantados, e as justificativas das escolhas são apresentadas a seguir, evidenciando como elas sustentam integridade, autenticidade, desempenho, segurança e evolução do sistema.

3.4.1 Arquitetura

Para atender às exigências de integridade e autenticidade, o alvo arquitetural é integrar o sistema ao SSO da UFU, conforme previsto no início do projeto e na modelagem anterior (LACERDA, 2024). Contudo, devido à indisponibilidade momentânea do serviço pelo CTIC, adotou-se, para a primeira versão, uma autenticação provisória baseada em e-mail institucional com emissão de Padrão aberto para criação e verificação de *tokens* baseados em JSON (*JSON Web Token*) (JWT), um padrão aberto para criação de *tokens* compactos e seguros, utilizados para transmitir informações de autenticação entre cliente e servidor. Os *tokens* foram configurados para curta duração, atendendo aos requisitos não funcionais (RNF001, RNF006). Dessa forma, a estratégia mantém segurança mínima para identificação e autorização, com barreiras técnicas claras, ao mesmo tempo em que preserva a meta institucional de integração ao SSO como caminho de evolução natural do sistema.

Por conseguinte, por ter requisitos de modularidade, continuidade e consistência de dados, foi adotada uma arquitetura da plataforma como cliente-servidor, vantagens destacadas no livro de Tanenbaum e Steen (2006). Desse modo, o sistema foi dividido em três componentes com funcionalidades distintas:

1. **Cliente**: responsável pela *interface* visual, e comunicação com o Servidor;
2. **Servidor**: componente lógico que centraliza as regras de agendamento e o tratamento de erros, com a finalidade de garantir consistência e integrar com componente que armazena os dados;
3. **Banco de dados**: entidade responsável por armazenar as informações necessárias de forma otimizada.

Ademais, considerando esses componentes, foi modelado o novo fluxo de agendamento do Schedule-UFU que demonstra evidentemente o ganho de eficiência sobre o fluxo atual, como pode-se observar na Figura 3.

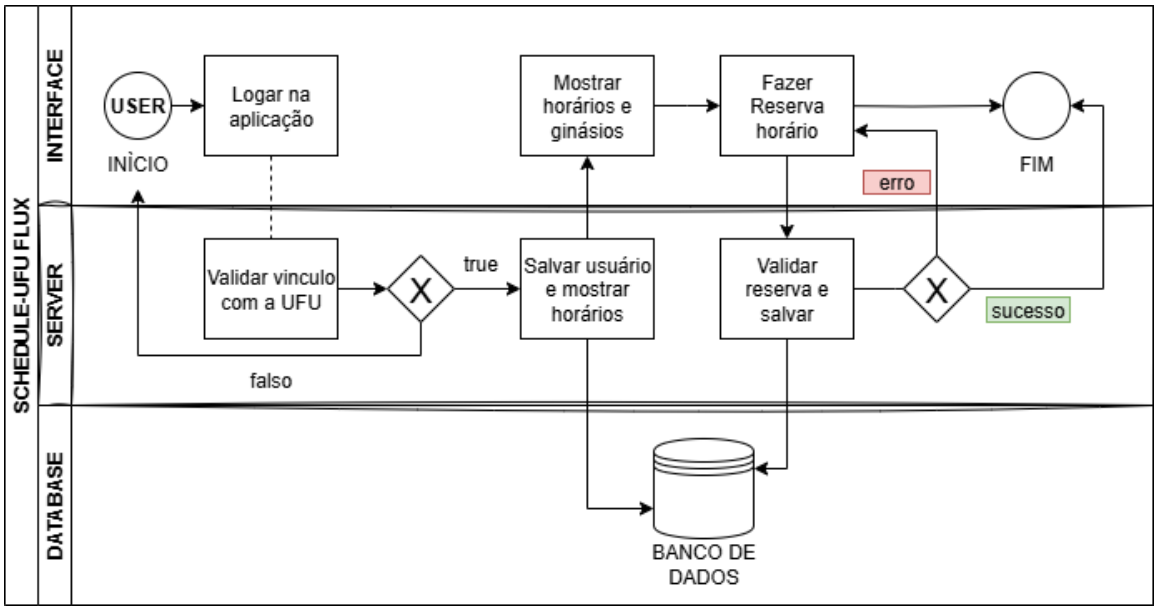

Figura 3 – Novo fluxo de agendamento proposto para o Schedule-UFU. Fonte: Do autor.

3.4.2 Tecnologia

Nesse sentido, para garantir a continuidade e a possibilidade de posterior integração com os sistemas do CTIC, foi adotada a utilização da linguagem Java com o *framework* Spring Boot para o servidor (*backend*), Angular para a *interface* visual (*frontend*) e PostgreSQL para o armazenamento de dados (*storage*).

Em cada um desses componentes foram aplicadas práticas que favorecem a legibilidade do código e a manutenção futura. No *frontend*, seguiu-se o padrão digital de

componentes do governo federal (BRASIL, 2024), assegurando consistência, acessibilidade e identidade visual de plataformas governamentais. Já no servidor, foram incorporadas técnicas de DDD, visando à construção de um sistema robusto e preparado para evolução incremental, o qual é impulsionado pela escolha da tecnologia, como bem observado por Anwar e Bairstow (2023).

Destaca-se, ainda, a aplicação do padrão arquitetural hexagonal (*Ports and Adapters*) no *backend* em Java, o que possibilita a divisão de funcionalidades em casos de uso, promove a modularidade e incentiva boas práticas de desenvolvimento, conforme discutido no texto de Khononov (2021).

3.5 Primeira Versão

Com o intuito de atender aos requisitos levantados e apresentar uma primeira versão minimamente funcional do sistema e o fluxo principal demonstrado na Figura 3, foi elaborada essa versão que busca gerar valor em curto prazo, mas que, ao mesmo tempo, mantenha características de extensibilidade e evolução. Para isso, definiu-se uma arquitetura inicial contemplando cliente ou *interface* gráfica, servidor e banco de dados, seguindo o modelo de Tanenbaum e Steen (2006), o qual foi representado por diagramas que guiam o desenvolvimento.

3.5.1 Componentes

A solução foi estruturada no modelo cliente-servidor, devido às vantagens de modularidade, consistência e escalabilidade descritas por Tanenbaum e Steen (2006). Os principais componentes podem ser observados na Figura 4 e são descritos a seguir:

- **Schedule-UFU (Cliente)**: desenvolvido em Angular, representa a *interface* visual utilizada pelos usuários finais. É responsável por realizar autenticação via e-mail institucional, enviar requisições de agendamento, listagem e cancelamento, bem como disponibilizar telas de administração básicas;
- **Schedule-Core (Servidor)**: construído em Java com o *framework* Spring Boot, concentra a lógica do negócio. Nele foram implementados os controladores de autenticação, agendamento e administração. Também é responsável pela emissão e validação de *tokens* JWT, que garantem a segurança mínima para acesso às rotas;
- **Banco de dados**: implementado em PostgreSQL, executado em container Docker, é responsável pela persistência dos dados relativos a usuários, reservas, ginásios e restrições. Foi configurado para manter integridade e consistência transacional.

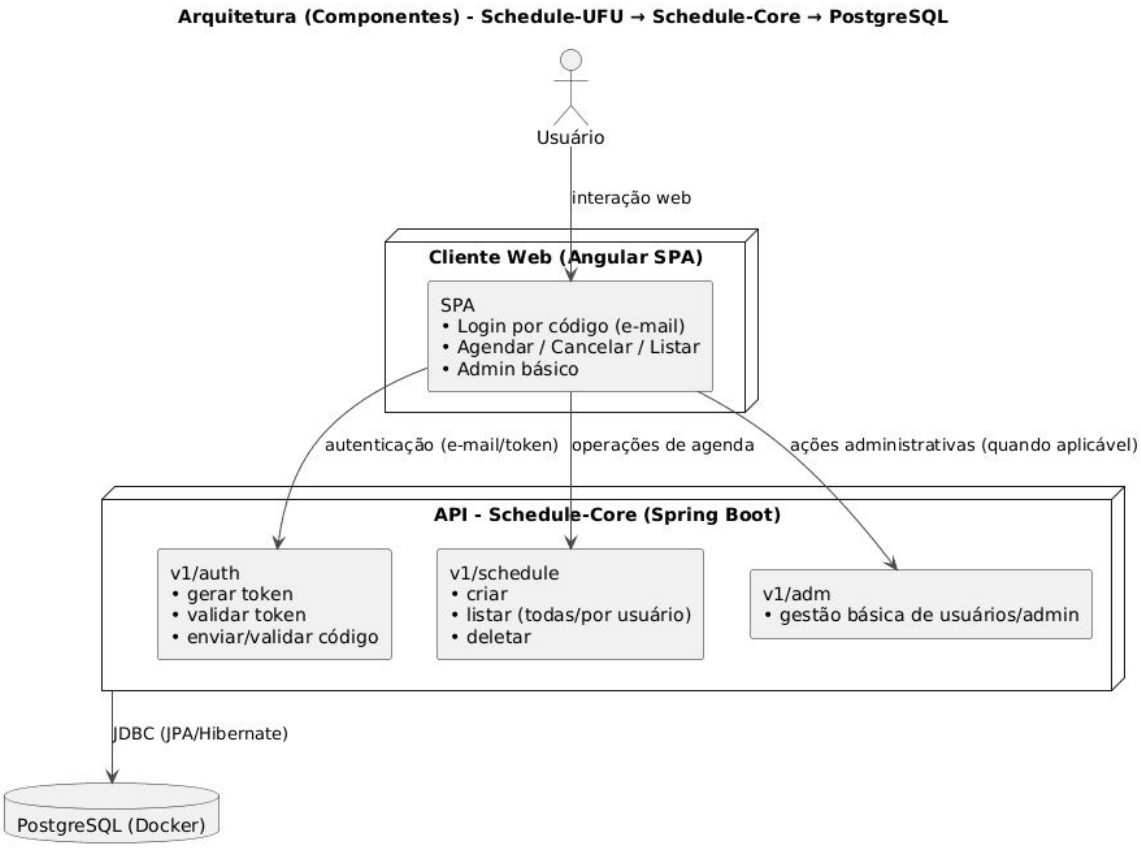

Figura 4 – Componentes. Fonte: Do autor.

Devido a essa separação de responsabilidades, o sistema se torna eficiente e com lógica centralizada, o que facilita a manutenção e a evolução incremental. A centralização das regras de negócio no servidor evita duplicidade de implementações no cliente e garante maior consistência no tratamento das operações, característica essencial em sistemas distribuídos (TANENBAUM; STEEN, 2006). Além disso, ao isolar as camadas, é possível aplicar boas práticas de engenharia de software como o baixo acoplamento e a alta coesão, reduzindo a probabilidade de erros e simplificando futuras alterações, conforme defendem Sommerville (2011), Pressman e Maxim (2021). Essa abordagem também favorece a escalabilidade, uma vez que o servidor pode ser replicado ou ampliado de forma independente do cliente, enquanto o banco de dados garante a integridade transacional. Assim, a primeira versão não apenas atende aos requisitos funcionais imediatos, mas também cumpre requisitos não funcionais de manutenção, modularidade e confiabilidade, assegurando a longevidade da solução e criando a base necessária para evoluções futuras, como a integração ao SSO da UFU e a ampliação das funcionalidades administrativas.

3.5.2 Modelagem Banco de Dados

Para atender às demandas de autenticação, realização de agendamentos e controle de espaços esportivos, e para representar de forma clara e objetiva como esses elementos

se relacionam entre si, foi adotado o modelo relacional normalizado. Essa escolha garante eficiência no acesso às informações, evita redundâncias e mantém a integridade dos dados por meio de chaves primárias e estrangeiras bem definidas.

Como primeira versão do projeto, foram adotadas algumas premissas que buscam equilibrar simplicidade e boa performance, garantindo tempo de resposta adequado e facilidade de manutenção:

- **Duração fixa das reservas**: definiu-se que os horários marcados terão sempre duração de uma hora, com intuito de possibilitar a exibição dos horários disponíveis para o usuário por meio de lógica no servidor. Assim, basta salvar o horário de início da reserva, com validações para checar se está dentro do funcionamento do ginásio, simplificando consultas e verificações de disponibilidade;
- **Geração de horários pelo servidor**: as possibilidades de agendamento são calculadas dinamicamente pelo servidor, de hora em hora, sem necessidade de armazenar combinações inválidas no banco;
- **Identificação de usuários**: para autenticação via JWT, o email é utilizado como chave primária, já que é a credencial usada no envio do código de verificação. A matrícula também é mantida como campo único, garantindo coerência com a identificação acadêmica utilizada pela UFU;
- **Gestão de restrições**: foi criada uma entidade genérica de restrições para contemplar feriados, manutenções ou bloqueios de horários, permitindo flexibilidade sem alterar a estrutura da base;
- **Modelagem simplificada de campus**: como o MVP não exige controle por campus, optou-se por não criar uma entidade específica para isso. O ginásio foi definido como chave principal nesse contexto, suficiente para diferenciar os espaços;
- **Modelo relacional**: pelo baixo volume de dados, mas com necessidade de consistência transacional e relacionamentos claros (1:N entre usuários e reservas, ginásios e reservas, além de ginásios e restrições), a escolha por um banco de dados relacional foi a mais adequada.

No armazenamento de dados, foi selecionado o PostgreSQL, por ser um banco de dados relacional de código aberto, sem custo de licenciamento e já adotado como padrão em diversos portais institucionais da UFU. Além disso, o PostgreSQL é reconhecido pela confiabilidade, suporte contínuo e pela capacidade de lidar com transações complexas e tipos de dados avançados, características ressaltadas por Juba, Vannahme e Volkov (2015). Dessa forma, a escolha garante robustez e escalabilidade, reforçando a viabilidade técnica e financeira do sistema, além de alinhar a solução com práticas institucionais.

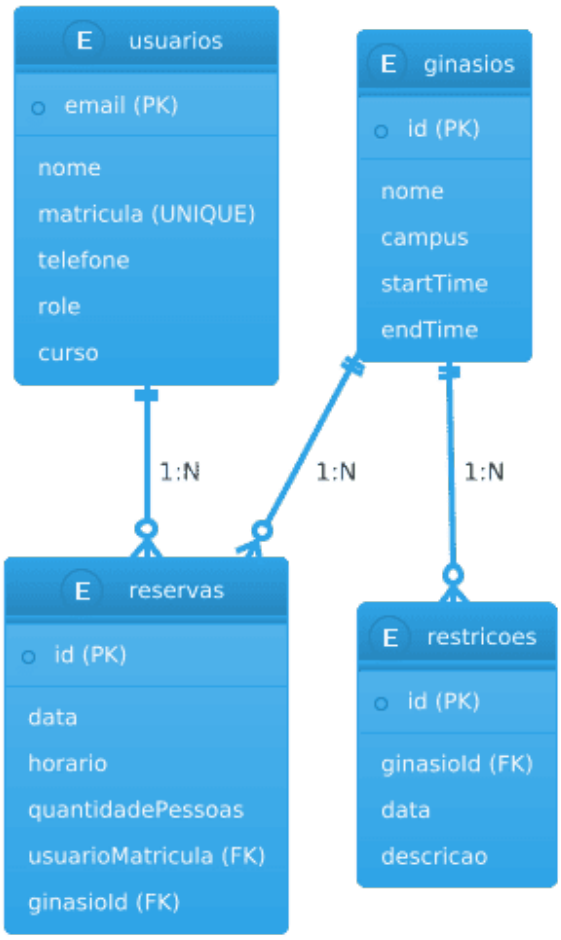

Figura 5 – Modelagem Banco de Dados. Fonte: Do autor.

Essa modelagem inicial garante simplicidade para o MVP e, ao mesmo tempo, preserva espaço para evolução incremental, caso novas entidades ou relacionamentos sejam incorporados em versões posteriores.

3.5.3 *Interface* Visual (Schedule-UFU)

A *interface* visual do sistema foi projetada a partir de protótipos desenvolvidos no Figma por Lacerda (2024), com a padronização dos componentes baseada no DSGov (BRASIL, 2024). Essa escolha assegurou acessibilidade, identidade visual institucional e uniformidade no *design*, em consonância com os padrões já aplicados em diferentes páginas da UFU. O fluxo de interação do usuário — que contempla autenticação por e-mail, seleção do espaço, definição de data e horário, verificação de disponibilidade e confirmação da reserva — pode ser observado na Figura 3.

Importa ressaltar que a camada visual foi concebida com base na proposta de Lacerda (2024), que contemplava os fluxos principais de agendamento para usuário e administrador, além de gestão básica de usuários e *logs* do sistema. Entretanto, com o

intuito de entregar um *MVP*, procedeu-se à simplificação de alguns filtros de tabela, à padronização do uso de formulários e à retirada de recursos administrativos (como registros de *logs* e a categorização de quadras por modalidade esportiva), adaptando o *design* e tornando esta última mais genérica para ampliar a aplicabilidade do sistema sem amarrá-lo a categorias específicas. As decisões preservam a coerência visual e os princípios de acessibilidade definidos pelo DSGov.

O *frontend*, denominado Schedule-UFU, foi desenvolvido utilizando o *framework* Angular, por ser a tecnologia empregada em outros portais da universidade, pela compatibilidade com os padrões do DSGov e por suas características de escalabilidade e eficiência no desenvolvimento de algumas SPA robustas em ambientes institucionais (ULUCA, 2024). Nesse contexto, algumas premissas nortearam o desenvolvimento da *interface*:

- **Estrutura organizada de pastas**: a organização do código foi realizada de forma modular, segmentando funcionalidades para aumentar a manutenibilidade e possibilitar alto nível de reutilização;
- **Componentização**: todos os elementos que não eram exclusivos de uma página foram implementados como componentes independentes e agrupados em pastas próprias, permitindo sua reutilização em diferentes partes da aplicação e assegurando a padronização visual das telas;
- **Padrão de formulários**: como o sistema exige diversas entradas de dados dos usuários, optou-se pelo uso de formulários em modais com validações de formato e obrigatoriedade de campos. O botão de envio é liberado apenas quando todos os requisitos são atendidos, prevenindo inconsistências e mantendo a lógica de regras de validação desacoplada do servidor;
- **Responsividade**: embora os componentes estilizados do DSGov sejam a base visual, a biblioteca *Tailwind Folhas de Estilo em Cascata (Cascading Style Sheets) (CSS)* foi incorporada para ampliar a responsividade e acelerar o desenvolvimento. Ajustes adicionais foram aplicados somente quando necessários, sobretudo em casos de personalização de componentes herdados do DSGov;
- **Separação de perfis**: as telas gerais podem ser acessadas também por administradores, o que lhes permite visualizar o sistema sob a perspectiva dos usuários. Entretanto, as funcionalidades administrativas permanecem restritas apenas a este perfil, assegurando que cada grupo utilize exclusivamente os recursos pertinentes ao seu papel e preservando a segurança do sistema.

A estrutura de diretórios do Schedule-UFU foi organizada da seguinte forma:

Caminho	Detalhes
`schedule-ufu/`	Raiz do *frontend* (Angular).
`schedule-ufu/src/app/`	Código-fonte principal da aplicação.
`schedule-ufu/src/app/components/`	Componentes reutilizáveis da *interface*.
`schedule-ufu/src/app/pages/`	Páginas principais (*login*, reservas, administração).
`schedule-ufu/src/app/services/`	Serviços para comunicação com o Servidor.
`schedule-ufu/src/assets/`	Arquivos estáticos (imagens, ícones, estilos globais).
`schedule-ufu/src/environments/`	Arquivos de configuração de ambiente (dev, prod).

Tabela 8 – Estrutura de pastas do Schedule-UFU. Fonte: Do autor.

3.5.3.1 Autenticação

A tela inicial do sistema solicita que o usuário insira um endereço de e-mail institucional válido, no formato `nome@ufu.br`. Para isso, foi implementada uma validação de máscara que garante tanto o preenchimento do campo quanto a conformidade com o padrão exigido, liberando o botão de envio apenas após a inserção correta (Figura 6). Ao confirmar o envio, o sistema registra o e-mail na memória local do cliente e solicita ao servidor a geração de um código numérico pseudo-aleatório de seis dígitos. Esse código é encaminhado ao usuário por e-mail, tendo como remetente o administrador da aplicação.

Figura 6 – Tela Inicial - E-mail UFU. Fonte: Do autor.

Na sequência, o usuário é redirecionado para a tela de *login* (Figura 8), onde deve informar dados complementares necessários, como matrícula, nome e telefone, além de inserir o código recebido (Figura 7). A *interface* aplica máscaras de validação em todos os campos, garantindo que apenas valores compatíveis sejam aceitos. Após o preenchimento correto, o botão de *login* é habilitado e, ao ser acionado, os dados são enviados ao servidor

juntamente com o e-mail previamente armazenado. O servidor, por sua vez, verifica a correspondência entre o código submetido e o que foi gerado e armazenado temporariamente em memória, retornando uma mensagem de sucesso em caso de validação positiva.

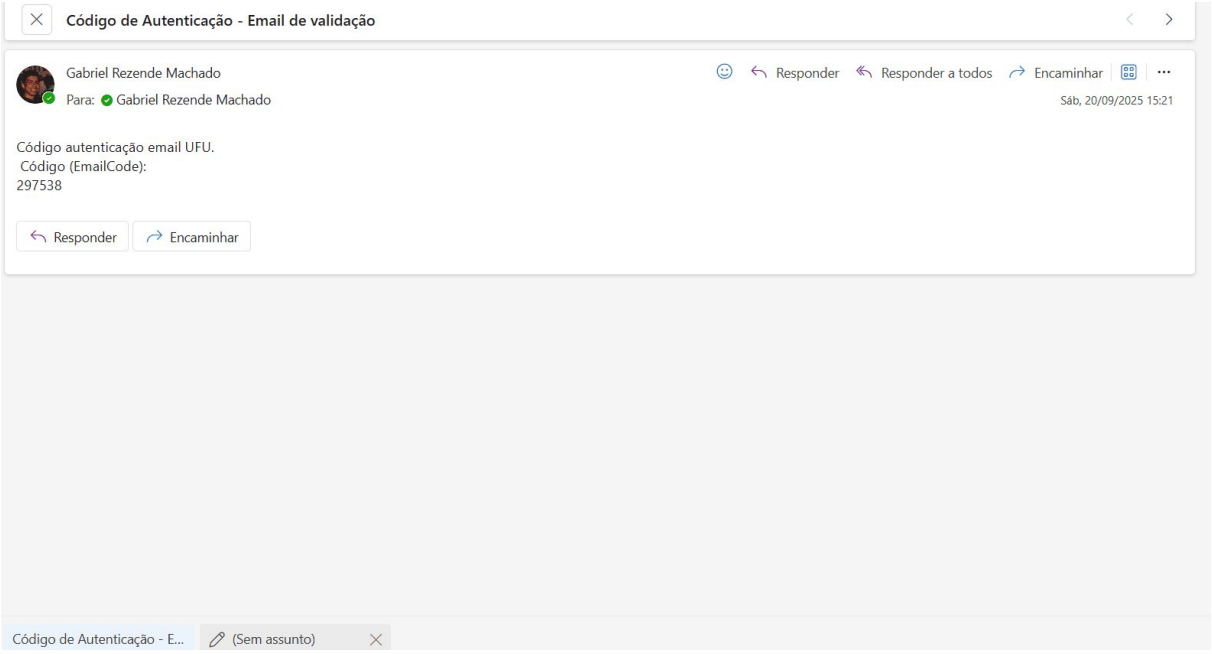

Figura 7 – Código recebido por e-mail. Fonte: Do autor.

Figura 8 – Tela de *Login* - Validação do Código. Fonte: Do autor.

Concluída a validação, o *frontend* combina os dados do formulário com o e-mail do usuário e solicita ao Schedule-Core a geração de um *token* JWT. Esse *token*, válido por 24 horas, é armazenado no cliente juntamente com as informações básicas do usuário

e utilizado em todas as requisições subsequentes ao *backend*, garantindo autenticação e controle de acesso.

3.5.3.2 Visão Usuário

Após a autenticação bem-sucedida, o usuário é direcionado para a página principal do sistema, onde pode visualizar os agendamentos disponíveis. A tela inicial (Figura 9) apresenta os horários organizados por data e ginásio, destacando os períodos já reservados e os intervalos livres, sendo que a visualização se inicia sempre no dia atual, exibindo os agendamentos de todos os ginásios. Mas, caso não existam reservas, a tela permanece vazia e, ao selecionar um ginásio específico, o sistema calcula automaticamente os horários ocupados e disponíveis para o dia escolhido. Dessa forma, essa abordagem de exibição facilita a identificação de espaços livres e auxilia no planejamento das reservas.

Figura 9 – Tela de Agendamentos - Visão Inicial Usuário. Fonte: Adaptado de Lacerda (2024).

Para efetuar um novo agendamento, o usuário deve acionar o botão azul para abrir o formulário de criação (Figura 10), no qual são solicitados dados como ginásio, data, horário e informações adicionais relacionadas à reserva. O sistema aplica validações automáticas em todos os campos, habilitando o botão de envio apenas após o preenchimento correto, impedindo inconsistências e garantindo que somente solicitações válidas sejam processadas pelo servidor.

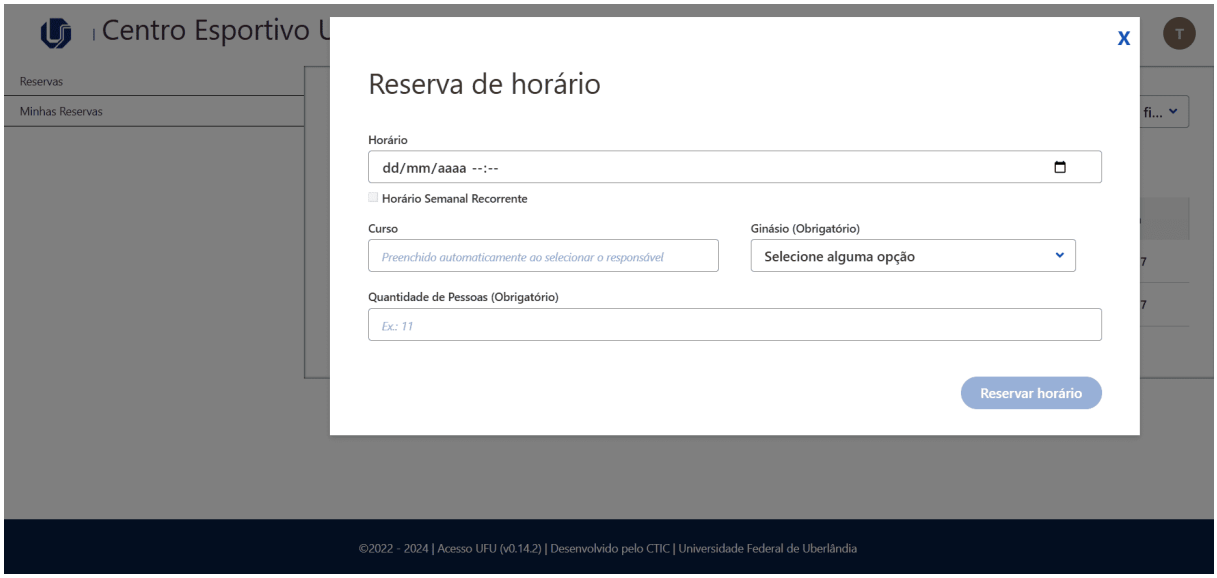

Figura 10 – Tela de Agendamentos - Formulário de Criação de Agendamentos. Fonte: Adaptado de Lacerda (2024).

Além disso, há a seção *Meus Agendamentos* (Figura 11), que apresenta as reservas vinculadas ao usuário autenticado, filtradas por sua matrícula. Um filtro por ginásio está disponível para facilitar a localização de agendamentos específicos.

Figura 11 – Tela de Meus Agendamentos - Visão Usuário. Fonte: Adaptado de Lacerda (2024).

Caso seja necessário cancelar uma marcação, há um formulário de exclusão (Figura 12), acessível por meio do botão vermelho. Nessa tela, o usuário informa os dados do agendamento a ser removido e confirma a operação antes de sua efetivação, o que

assegura maior controle e evita cancelamentos acidentais. É importante destacar que, nesta primeira versão do projeto, não está contemplada a edição de agendamentos. Todavia, o usuário pode visualizar todas as suas reservas e gerenciá-las de forma prática e centralizada nesta aba.

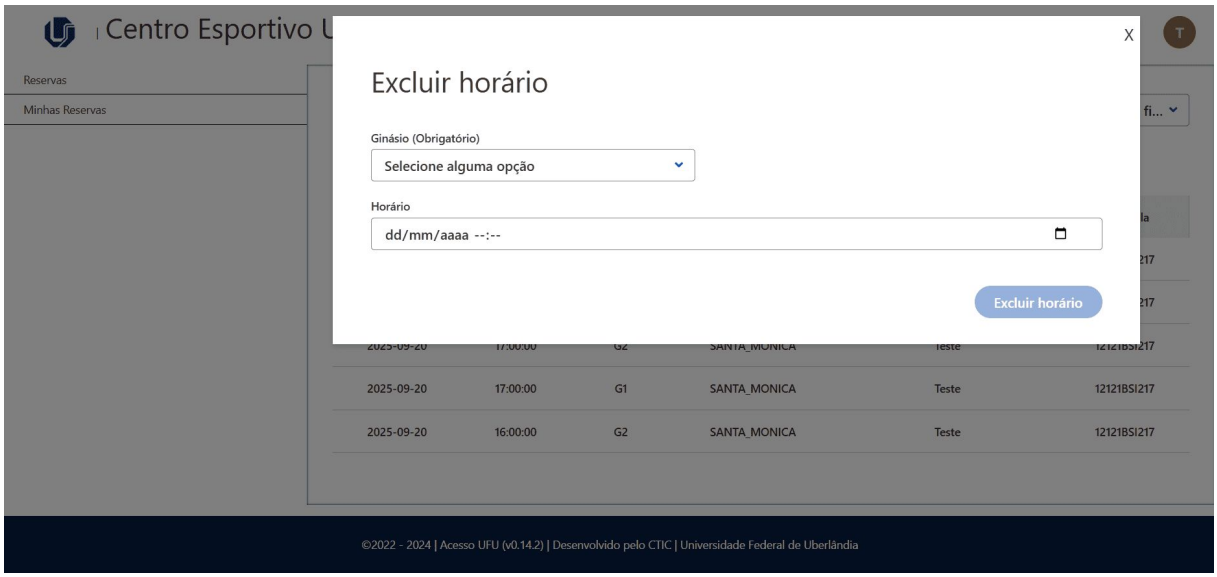

Figura 12 – Tela de Agendamentos - Formulário de Exclusão de Agendamentos. Fonte: Do autor

Desse modo, esse conjunto de funcionalidades garante uma experiência intuitiva e transparente para o usuário, que consegue realizar reservas, acompanhar seus agendamentos e efetuar cancelamentos de forma simplificada.

3.5.3.3 Visão Administrador

Após o fluxo de validação do e-mail, o administrador do sistema tem acesso tanto às abas disponíveis para os usuários quanto a funcionalidades adicionais de criação e exclusão de agendamentos (Figura 9). Por padrão, o administrador é identificado pela matrícula inicial `00000ADM000`, criada automaticamente no servidor a partir do e-mail administrativo configurado. Além dessas funções, o perfil administrativo conta com a aba *Configurações* (Figura 13), que concentra recursos de gerenciamento do sistema. Essa página é dividida em três seções principais: *Geral*, voltada ao controle de agendamentos; *Espaços Esportivos*, destinada à gestão dos ginásios; e *Permissões*, que permite visualizar os acessos criados e conceder privilégios administrativos. Cumpre salientar que essa aba é invisível para perfis comuns e só pode ser acessada mediante *token* de autenticação com permissões administrativas; assim, mesmo que um usuário não autorizado visualize a página, o *token* em uso não permitirá acesso ou manipulação de dados.

A seção *Geral* apresenta uma tabela com todos os agendamentos realizados no sistema, acompanhada de dois botões de ação (Figura 13). O primeiro permite a criação de agendamentos pelo administrador, que pode cadastrar reservas em nome de qualquer usuário a partir do preenchimento de um formulário com os dados necessários (Figura 14). O segundo possibilita a exclusão de horários já reservados, identificados pela matrícula do usuário (Figura 15). Nesse caso, quando o agendamento cancelado corresponder a uma data futura, o servidor dispara automaticamente um e-mail de notificação ao usuário, tendo o administrador como remetente (Figura 16), reforçando a transparência e a confiabilidade do processo.

Figura 13 – Tela de Configurações - Aba Geral. Fonte: Adaptado de Lacerda (2024).

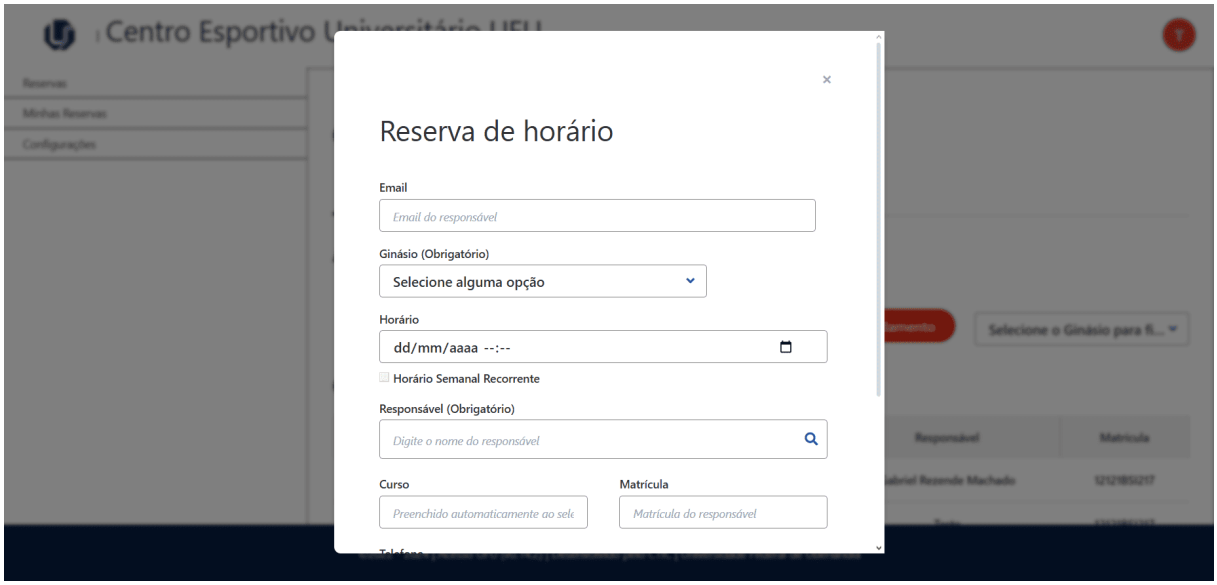

Figura 14 – Tela de Configurações - Formulário Agendamento. Fonte: Adaptado de Lacerda (2024).

Figura 15 – Tela de Configurações - Formulário de Cancelamento de Agendamento. Fonte: Adaptado de Lacerda (2024).

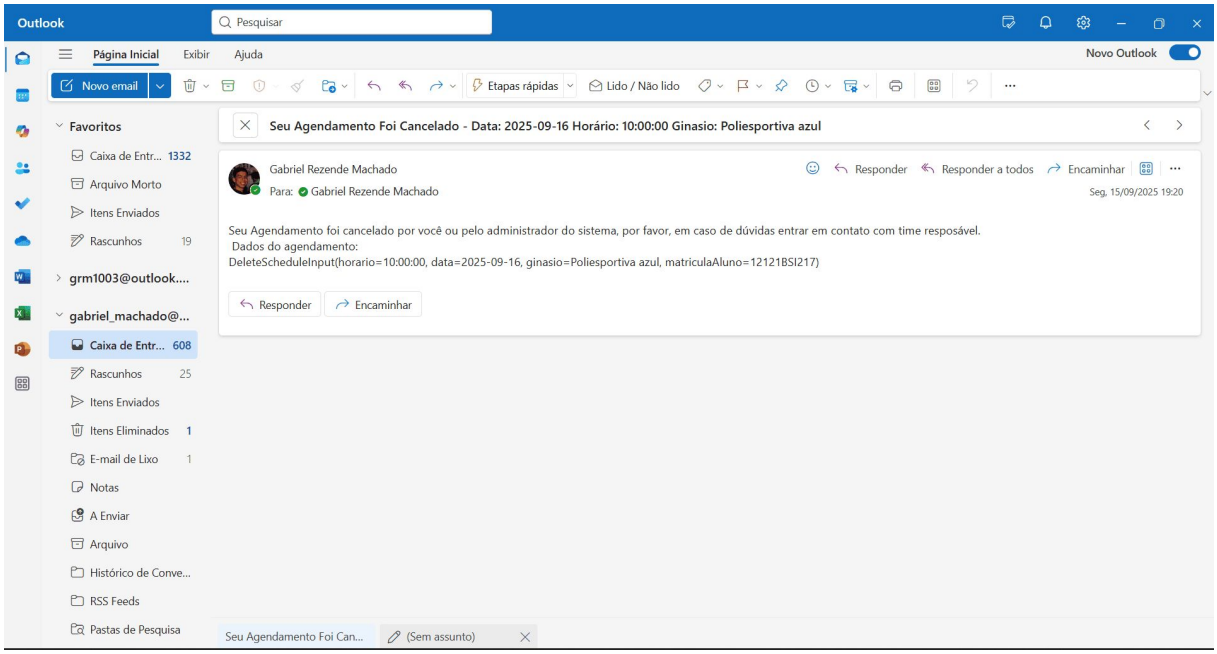

Figura 16 – Aviso de Cancelamento - Formulário de Cancelamento de Agendamento. Fonte: Do autor

Na parte inferior da mesma tela(Figura 13), encontra-se a parte de controle de restrições (Figura 17), utilizada para gerenciar períodos de indisponibilidade dos ginásios, como feriados, manutenções ou eventos institucionais. O administrador pode cadastrar novas restrições por meio de um formulário que exige a indicação de data e ginásio, sendo a descrição opcional (Figura 18). Embora não obrigatória, a descrição auxilia na comunicação de erros e no registro de justificativas. Outrossim, há a possibilidade de remover restrições previamente criadas (Figura 19). Essas operações se integram diretamente ao servidor, que valida as restrições durante o processo de criação de reservas.

Figura 17 – Tela de Configurações - Restrições. Fonte: Adaptado de Lacerda (2024).

Figura 18 – Tela de Configurações - Formulário Criação Restrição. Fonte: Do autor

Figura 19 – Tela de Configurações - Formulário Deletar Restrição. Fonte: Do autor

Nesse contexto, além do gerenciamento geral, existe a seção de administração de espaços esportivos, que apresenta a listagem de todos os ginásios com seus respectivos dados e horários de funcionamento (Figura 20). Nessa área, estão disponíveis duas ações principais: a criação de novos registros de ginásio (Figura 21) e a exclusão de espaços já cadastrados (Figura 22). Ambas seguem o mesmo padrão aplicado nas demais funcionalidades do sistema, exigindo a validação dos campos antes de permitir o envio e a execução do processo.

Figura 20 – Tela de Configurações - Aba Espaços Esportivos. Fonte: Adaptado de Lacerda (2024).

Figura 21 – Tela de Configurações - Formulário Criação Ginásio. Fonte: Adaptado de Lacerda (2024).

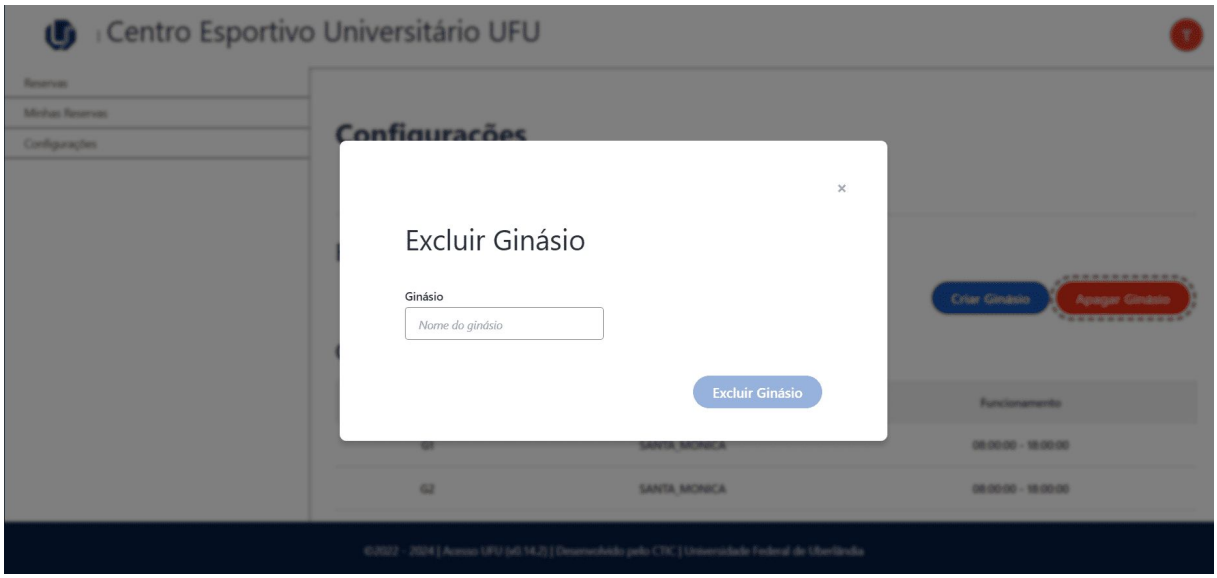

Figura 22 – Tela de Configurações - Formulário de Deletar Ginásio. Fonte: Adaptado de Lacerda (2024).

De forma complementar, há ainda a aba destinada às permissões, voltada à gestão de usuários. Sempre que ocorre a autenticação, os dados do indivíduo são criados ou atualizados automaticamente, contemplando matrícula, nome e demais informações exibidas na tela de *login* (Figura 8). Nesse espaço, o administrador pode visualizar todos os registros cadastrados (Figura 23) e, por meio de um formulário específico (Figura 24), atribuir ou atualizar credenciais administrativas para administradores. Essa funcionalidade possi-

bilita distribuir responsabilidades de gestão entre diferentes membros da equipe ou setores que necessitam de maior nível de acesso.

Figura 23 – Tela de Configurações - Aba Permissões. Fonte: Adaptado de Lacerda (2024).

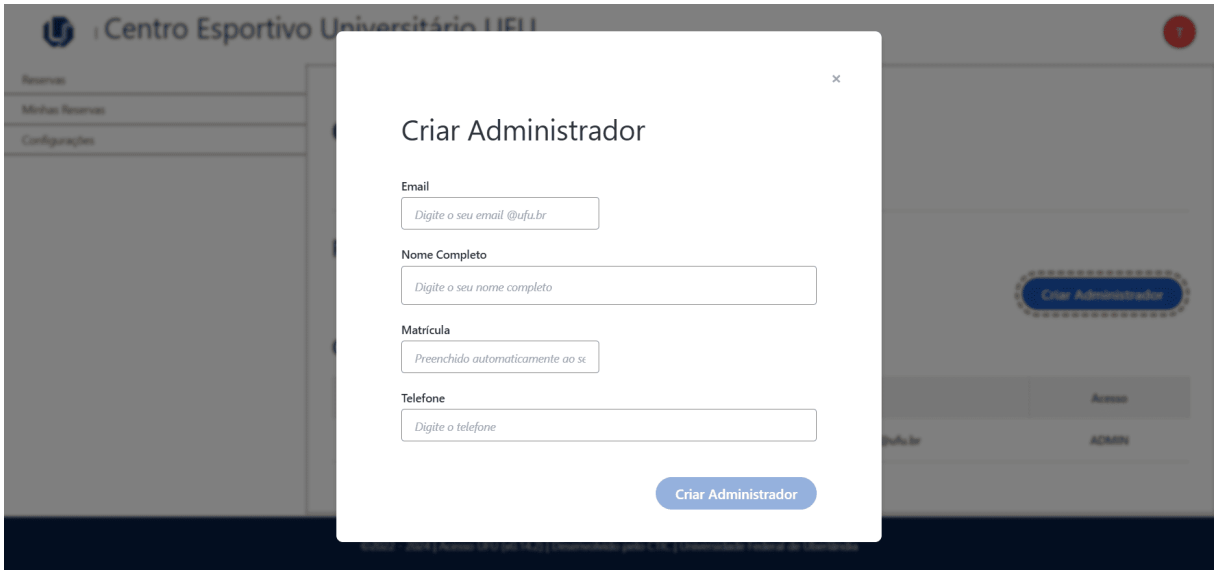

Figura 24 – Tela de Configurações - Formulário de Atualização ou Criação de Administrador. Fonte: Adaptado de Lacerda (2024).

Por conseguinte, o módulo administrativo garante recursos essenciais para a governança da plataforma. Ainda que algumas funcionalidades estejam limitadas nesta primeira versão, o sistema já viabiliza a administração eficaz de espaços, usuários e permissões, representando um avanço significativo em relação às práticas atuais de agendamento.

3.5.4 Servidor (*Schedule-Core*)

O Schedule-Core foi implementado em Java utilizando o *framework* Spring Boot, aproveitando seus recursos de modularização, injeção de dependência e integração nativa com bibliotecas de segurança (ANWAR; BAIRSTOW, 2023). A arquitetura adotada foi a hexagonal (Ports and Adapters), em conformidade com as recomendações de (KHONONOV, 2021), a qual organiza o sistema em duas grandes camadas: lógica de negócio/domínio e infraestrutura, a qual garante que cada funcionalidade seja representada como um caso de uso independente, mantendo o domínio agnóstico às dependências externas.

Caminho	**Detalhes**
`schedule-core/`	Raiz do *backend* (Spring Boot).
`schedule-core/domain/`	Entidades e regras de negócio;
`schedule-core/infrastructure/`	Controllers, Adapters;
`schedule-core/ScheduleCoreApplication`	Classe principal que inicializa a aplicação.

Tabela 9 – Estrutura de pastas do Schedule-Core. Fonte: Do autor.

Nesse modelo, operações como CreateSchedule (Criação de agendamentos) são tratadas como casos de uso distintos, que expõem apenas *interfaces* e não implementações diretas, aplicando o princípio da inversão de dependência (D) do SOLID. Assim, todas as dependências externas, como banco de dados, envio de e-mail e autenticação por geração de *tokens*, são abstraídas em portas (AuthPort, DatabasePort, EmailSenderPort), as quais as implementações concretas dessas portas ficam isoladas na camada de infraestrutura, por meio dos adaptadores correspondentes, garantindo que alterações futuras nessas tecnologias não afetem a lógica da aplicação.

Além disso, os controladores (AuthController, ScheduleController e AdmController) também pertencem à camada de infraestrutura, pois eles são responsáveis por expor os *endpoints* REST e injetam as *interfaces* dos casos de uso, e esse funcionamento não deve se atrelar à lógica de aplicação. Desse modo, essa estratégia possibilita a evolução futura do sistema, com inclusão de novos canais de comunicação ou substituição de dependências, sem impacto no núcleo do domínio.

Entre os pontos de destaque da implementação, é importante ressaltar:

- **Estrutura organizada de pastas**: as camadas de domínio, aplicação e infraestrutura foram claramente separadas, facilitando a compreensão e manutenção do código;
- **Configuração de Compartilhamento de Recursos entre Origens (*Cross-Origin Resource Sharing*) (CORS)**: foi utilizado o parâmetro allowedOrigins

como todos ("*"), permitindo chamadas de diferentes origens durante a fase de prototipação e testes. Embora simples, essa configuração assegurou o funcionamento do cliente Schedule-UFU em diferentes ambientes;

- **Parâmetros administrativos**: a aplicação exige que sejam configurados, via variáveis de ambiente, o e-mail e a senha do administrador. Esses valores são utilizados tanto para envio de mensagens de verificação quanto para a criação do *hash* que assina os *tokens* JWT, além da criação na entidade do banco de dados do administrador no primeiro início;

- **Profiles de execução**: foram definidos diferentes arquivos de propriedades application.properties para controlar a configuração do ambiente. No profile docker, são especificadas as credenciais e o host de conexão ao PostgreSQL em container, garantindo portabilidade e facilidade de execução da primeira versão funcional em qualquer ambiente compatível;

- **Geração dinâmica de horários**: os horários disponíveis para reserva são calculados pelo servidor com base no horário inicial e final de cada ginásio, assumindo a premissa de intervalos fixos de uma hora estabelecida na modelagem do banco de dados;

- **Autenticação baseada em JWT**: os *tokens* são assinados com um *hash* derivado do e-mail do administrador, definido em variáveis de ambiente na inicialização da aplicação. Essa configuração adiciona segurança e flexibilidade ao processo de autenticação;

- **Rotas protegidas**: o filtro de segurança (*Security Filter)* garante que apenas usuários autenticados, portando JWT válidos, tenham acesso às rotas críticas do sistema;

- **Envio de e-mails**: o fluxo de autenticação é complementado pelo envio de códigos de verificação para o e-mail institucional, utilizando a porta EmailSenderPort, abstraída na arquitetura hexagonal;

- **Validador de agendamentos**: foi implementado de forma reutilizável utilizando o padrão de projeto *Chain of Responsibility*, aplicado dinamicamente para encadear validações de disponibilidade, restrições e conflitos de reserva. Esse padrão reforça o princípio da responsabilidade única (S do SOLID), separando claramente cada regra de validação;

- **Princípios SOLID**: além da inversão de dependência (D) e da responsabilidade única (S), a arquitetura também aplica o princípio aberto/fechado (O), já que novos validadores ou serviços podem ser adicionados sem alterar as implementações existentes.

- **Testes unitários**: a camada domain concentra a lógica de negócios relacionada aos agendamentos, e nela foram implementados testes unitários para validar exclusivamente as *interfaces* de casos de uso e o padrão *Chain of Responsibility* aplicado ao processo de validação de agendamentos. Essa estratégia garante que a lógica central seja testada de forma isolada, mantendo o domínio desacoplado de implementações externas e alinhado aos princípios de responsabilidade única e inversão de dependência.

A arquitetura hexagonal aplicada no Schedule-Core organiza cada funcionalidade do sistema como um caso de uso independente, desacoplado de implementações externas. A seguir, apresentam-se os principais casos de uso implementados na primeira versão funcional:

- **Busca Agendamentos (GetSchedulesImpl)**: consulta de horários disponíveis e preenchimento automático de slots livres, com suporte a filtro por ginásio. O algoritmo mapeia os horários já ocupados e retorna uma visão completa das opções restantes para reserva;
- **Busca de Todos os Agendamentos (GetAllSchedulesImpl)**: recuperação de todas as reservas registradas, sem filtros. Destina-se principalmente a administradores que necessitam de uma visão global da utilização dos espaços esportivos;
- **Busca de Agendamentos por Usuário (GetUserSchedulesImpl)**: retorna os agendamentos vinculados a um usuário específico, identificado por sua matrícula. Inclui validações para garantir que a matrícula não seja nula ou vazia;
- **Exclusão de Agendamento (DeleteScheduleImpl)**: cancelamento de reservas já cadastradas, com envio de notificação por e-mail quando a data da reserva ainda não ocorreu. Este fluxo garante maior transparência ao usuário;
- **Criação de Agendamento (CreateScheduleImpl)**: cadastro de novas reservas por meio de um encadeamento de validações (*Chain of Responsibility*), que inclui: data, horário, existência do ginásio, restrições aplicáveis e verificação de duplicidade. Essa abordagem garante consistência e evita conflitos de horários;
- **Busca de Restrições (GetRestricoesImpl)**: consulta de todas as restrições cadastradas, como períodos de manutenção, eventos ou feriados, permitindo aos usuários e administradores visualizar indisponibilidades;
- **Busca de Restrição (DeleteRestricaoImpl)**: remoção de restrições previamente criadas, devolvendo disponibilidade aos espaços esportivos de forma dinâmica;

- **Criação de Restrição (CreateRestricaoImpl)**: cadastro de novas restrições com validações de integridade, como: data não nula, data não passada e verificação da existência do ginásio informado;
- **Busca de Ginásios (GetGinasiosImpl)**: recuperação de todos os ginásios cadastrados, permitindo ao usuário selecionar corretamente o espaço desejado durante o processo de reserva;
- **Exclusão de Ginásio (DeleteGinasioImpl)**: exclusão de ginásios que deixaram de estar disponíveis para reservas. Inclui validação para impedir nomes nulos ou vazios;
- **Criação ou Atualização de Ginásio (CreateOrUpdateGinasioImpl)**: criação de novos ginásios ou atualização de registros existentes em uma única operação. A implementação verifica a existência do ginásio antes de decidir entre criação ou atualização;
- **Busca de Usuários (GetUsersImpl)**: consulta de todos os usuários cadastrados no sistema. Utilizada principalmente para fins de gestão administrativa;
- **Geração de Autenticação (GenerateAuthImpl)**: emissão de *tokens* JWT para usuários autenticados. Inclui a validação dos dados de entrada e atualização automática do cadastro do usuário, caso necessário;
- **Geração de Administrador (GenerateAdmImpl)**: criação ou atualização de usuários com perfil administrativo, incluindo a validação das informações antes da operação no banco de dados;
- **Validação de E-mail (EmailValidatorAcessImpl)**: mecanismo de autenticação em dois fatores baseado em envio de código por e-mail. O caso de uso gera um código numérico seguro, armazena temporariamente para validação e garante que apenas usuários com acesso ao e-mail institucional possam se autenticar.

3.6 Validação com Usuários

Para avaliar a experiência de uso e a adequação das funcionalidades do sistema, foi realizada uma etapa de validação com três participantes: dois estudantes do curso de Sistemas de Informação e um do curso de Medicina Veterinária. A metodologia aplicada incluiu a observação direta durante o uso da plataforma e a técnica de *Thinking Out Loud*, na qual os participantes foram incentivados a verbalizar suas impressões enquanto interagiam com as diferentes telas do sistema, tendo como objetivos agendar um horário e, posteriormente, cancelá-lo. Essa abordagem possibilitou identificar, de forma espontânea, percepções positivas e pontos de melhoria relacionados à usabilidade.

De modo geral, os participantes relataram que a aplicação apresenta uma *interface* simples e intuitiva, permitindo ao usuário visualizar, criar e excluir agendamentos de forma clara, conforme também apontado por Lacerda (2024). A aceitação da comunidade acadêmica de uma ferramenta desse tipo tende a ser elevada, sobretudo com adequada divulgação institucional. A organização das funcionalidades foi considerada adequada, destacando-se como ponto forte a facilidade de navegação e a visibilidade das ações principais. Além disso, ressaltou-se a utilidade do sistema para proporcionar maior controle e transparência sobre os agendamentos.

Alguns usuários, entretanto, relataram confusão inicial quanto à necessidade de selecionar um ginásio específico para visualizar os horários disponíveis, uma vez que a tela inicial apresenta apenas os agendamentos já registrados. Apesar disso, as funcionalidades de marcação e exclusão foram avaliadas como acessíveis e bem implementadas. Entre as sugestões de melhoria, destacaram-se: a substituição do formulário de exclusão por um botão direto (ícone de lixeira) em cada item da tabela de *Meus Agendamentos* (Figura 11), eliminando a necessidade de inserir manualmente os dados do agendamento a ser cancelado; aprimoramentos de responsividade para dispositivos móveis; melhorias nas mensagens de erro, que atualmente são nativas do navegador e exibidas em inglês; e a filtragem para exibir apenas agendamentos futuros.

A validação contou com a participação de um representante da DIESU/UFU, setor responsável pela gestão de agendamentos. O especialista considerou o sistema mais polido do que o esperado e, embora funcional em sua versão atual, indicou oportunidades de evolução antes da publicação oficial. Importa frisar que tais itens — como reestruturar o armazenamento para eliminar a obrigatoriedade de divisão por hora, implementar agendamentos recorrentes e integrar ao CTIC/SSO da UFU para validações institucionais — configuram incrementos evolutivos sobre o núcleo já validado, e não uma readequação fundamental do projeto.

Essa orientação decorre de uma decisão deliberada de priorização de *MVP*, pactuada entre o setor responsável e a equipe de desenvolvimento: entregar primeiro as funcionalidades indispensáveis gerais. Em outras palavras, o escopo e a arquitetura permanecem coerentes; as melhorias sugeridas serão adicionadas progressivamente, sem ruptura de princípios ou refazimento estrutural.

Adicionalmente, a orientadora do trabalho sugeriu ajustes no *design* da *interface* para que esta se aproximasse mais do padrão visual adotado em outros portais institucionais da UFU. Entretanto, como o protótipo foi desenvolvido a partir do *design* proposto por (LACERDA, 2024), tais mudanças não foram implementadas nesta versão inicial, a fim de manter a consistência com o modelo de referência utilizado no desenvolvimento e pelo tempo para desenvolvimento.

Em síntese, a validação com usuários demonstrou que o sistema atendeu às ex-

pectativas iniciais em termos de usabilidade e funcionalidade, sendo considerado simples, funcional e adequado ao propósito proposto. Contudo, o processo também evidenciou oportunidades de aprimoramento que servirão de base para iterações futuras do desenvolvimento, possibilitando a publicação da ferramenta dentro da instituição e contribuindo para a otimização do fluxo atual de agendamentos.

4 Conclusão

O desenvolvimento do sistema *Schedule-UFU* resultou em uma primeira versão funcional que simplifica o processo de agendamento de espaços esportivos da UFU, avançando em relação ao modelo apenas visual proposto por (LACERDA, 2024). A adoção do *Design System* do Governo Federal (BRASIL, 2024), aliada ao uso de tecnologias modernas como Angular no *frontend* e Spring Boot no *backend*, possibilitou a criação de uma solução robusta, escalável e alinhada às práticas atuais de engenharia de software, como arquitetura hexagonal (KHONONOV, 2021), princípios SOLID (FOWLER, 2002) e autenticação JWT, demonstrando viabilidade técnica em ambientes institucionais, aliada à adoção de boas práticas de desenvolvimento.

Na Figura 25, apresenta-se a arquitetura final implementada nessa primeira versão, evidenciando a interação entre os módulos do cliente, servidor e banco de dados.

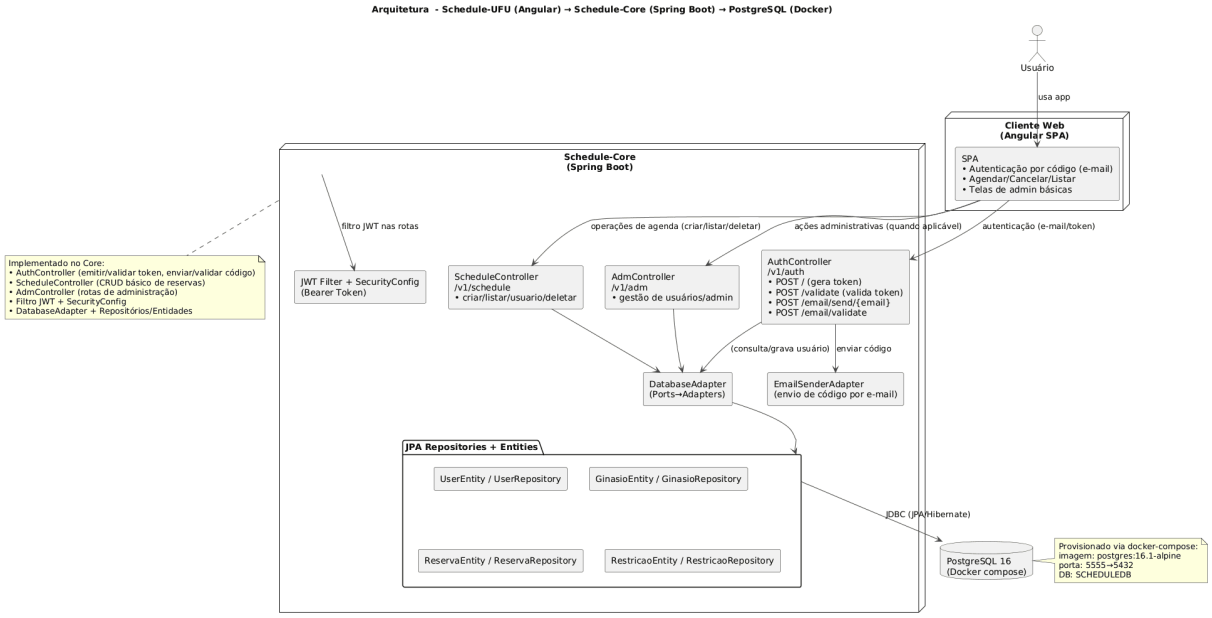

Figura 25 – Arquitetura Final Schedule-UFU e Schedule-Core . Fonte: Do autor

A validação com usuários demonstrou boa aceitação, evidenciando que a comunidade acadêmica valoriza ferramentas que tragam praticidade, controle e transparência ao processo de reservas. Para a DIESU, a informatização representa ganho de tempo em tarefas administrativas e redução de erros humanos, mas ainda que algumas demandas específicas, como agendamentos recorrentes e integrações institucionais, não tenham sido contempladas nesta versão inicial. Entretanto, o projeto também revelou limitações; o banco de dados foi propositalmente simplificado para viabilizar o MVP, a autenticação

restringiu-se ao uso de JWT por falta de integração com o SSO da UFU, e a coleta de opiniões não alcançou todos os perfis de usuários, como ingressantes que desconhecem o processo atual.

Conclui-se, portanto, que o Schedule-UFU atingiu parcialmente os objetivos iniciais, servindo como base sólida para futuras iterações que contemplem melhorias estruturais, integrações institucionais e maior complexidade na modelagem de dados, ainda que não esteja plenamente pronto para uma entrega em ambiente produtivo, acarretando em um passo importante rumo à modernização e otimização do processo de agendamentos da universidade. Como trabalhos futuros, propõe-se a evolução da modelagem de dados, a integração com sistemas institucionais como SSO e dados de matrícula e usuários, a implementação de recursos avançados de agendamento (como recorrência e notificações) e a evolução da *interface* visual em conformidade com os princípios de usabilidade e acessibilidade.

Por fim, destaca-se que a disponibilização pública do código-fonte, detalhada no **Apêndice A**, reforça o compromisso com a transparência e garante que a comunidade acadêmica e administrativa possa dar continuidade e expandir o projeto em versões futuras.

Referências

ANWAR, N.; BAIRSTOW, J. **Spring Boot for Modern Enterprises: Security, Scalability, and Seamless Integration**. 2023. Preprint, ResearchGate. Disponível em: <https://www.researchgate.net/publication/376534715>. Citado 3 vezes nas páginas 14, 33 e 50.

BOURQUE, P.; FAIRLEY, R. E. **Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0**. [S.l.]: IEEE Computer Society, 2014. ISBN 978-0769551661. Citado na página 18.

BRASIL, G. F. do. **Design System do Governo Federal (DSGov)**. 2024. Disponível em: <https://www.gov.br/ds/home>. Citado 9 vezes nas páginas 14, 16, 17, 18, 22, 30, 33, 36 e 56.

CHALANDAR, M. E.; DARVISH, P.; RAHMANI, A. M. A centralized cookie-based single sign-on in distributed systems. In: IEEE. **2007 ITI 5th International Conference on Information and Communications Technology**. [S.l.], 2007. p. 163–165. Citado na página 14.

DANTAS, M. B.; SILVA, L. G. O.; COSTA, L. S. da; MELO, M. S. de; COSTA, F. R. P. da; SOUZA, L. P. de; NEIVA, J. V. S. et al. Inserção dos estudantes de medicina nas atléticas acadêmicas: promoção de saúde física e mental por meio do esporte, da cultura e da arte. In: **13º Congresso Internacional Rede Unida**. [S.l.: s.n.], 2018. Citado 3 vezes nas páginas 12, 15 e 16.

FOWLER, M. **Patterns of Enterprise Application Architecture**. Boston, MA: Addison-Wesley, 2002. Reimpressões posteriores em 2012. ISBN 978-0321127426. Citado 2 vezes nas páginas 19 e 56.

FREUND, J.; RüCKER, B. **Real-Life BPMN (4th Edition): Includes an Introduction to DMN**. [S.l.]: Independently Published, 2019. ISBN 978-1086302097. Citado na página 20.

JUBA, S.; VANNAHME, A.; VOLKOV, A. **Learning PostgreSQL**. [S.l.]: Packt Publishing, 2015. ISBN 978-1783989188. Citado 2 vezes nas páginas 14 e 35.

KENDAL, S. **Object oriented programming using Java**. [S.l.]: Bookboon, 2009. ISBN 978-8776815011. Citado na página 14.

KHONONOV, V. **Learning Domain-Driven Design**. Sebastopol, CA: O'Reilly Media, 2021. ISBN 978-1098100131. Citado 5 vezes nas páginas 18, 19, 33, 50 e 56.

LACERDA, D. T. **Sistema de Agendamentos de Horários para os Espaços Esportivos na UFU: Modelagem e Prototipagem**. Trabalho de Conclusão de Curso (Ciência da Computação), Uberlândia, MG, 2024. Disponível em: <https://repositorio.ufu.br/handle/123456789/44797>. Citado 24 vezes nas páginas 5, 6, 7, 12, 13, 15, 18, 20, 21, 24, 25, 26, 31, 36, 40, 41, 43, 44, 46, 47, 48, 49, 54 e 56.

LUCENA, I. D. **Evolução do Sistema de Agendamento de Horários do Complexo Esportivo da UFCG - SAHCE-UFCG**. Trabalho de Conclusão de Curso (Ciência da Computação), Campina Grande, PB, 2023. Disponível em: <https://dspace.sti.ufcg.edu.br/handle/riufcg/32386>. Citado 4 vezes nas páginas 12, 15, 16 e 17.

Object Management Group (OMG). **Business Process Model and Notation (BPMN) Version 2.0**. 2013. Disponível em: <https://www.omg.org/spec/BPMN/2.0/>. Citado na página 20.

PIERI, B. d.; FRANçA, L. M.; OLIVEIRA, M. F. G. d.; FERREIRA, S. d. S. **SPACESYNC: Agendamentos de Espaços**. Trabalho de Conclusão de Curso (Técnico em Desenvolvimento de Sistemas Integrado ao Ensino Médio), Tupã, SP, 2023. Disponível em: <https://ric.cps.sp.gov.br/handle/123456789/17053>. Citado 2 vezes nas páginas 15 e 16.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**. 9. ed. São Paulo, Brasil: McGraw Hill Brasil, 2021. ISBN 978-6558040118. Citado 2 vezes nas páginas 18 e 34.

SINGH, H.; HASSAN, S. I. Effect of SOLID design principles on quality of software: An empirical assessment. **International Journal of Scientific & Engineering Research**, v. 6, n. 4, p. 1321–1324, 2015. Citado na página 19.

SOMMERVILLE, I. **Software engineering**. 9. ed. Boston, USA: Pearson Education Inc, 2011. ISBN 978-0137035151. Citado 4 vezes nas páginas 17, 18, 26 e 34.

TANENBAUM, A. S.; STEEN, M. V. **Distributed Systems: Principles and Paradigms**. 2. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2006. ISBN 978-0132392273. Citado 6 vezes nas páginas 13, 14, 20, 32, 33 e 34.

ULUCA, D. **Angular for Enterprise Applications — Third Edition: Build Scalable Angular Apps Using the Minimalist Router-First Architecture**. [S.l.]: Packt Publishing, 2024. EBook ISBN: 978-1805125037. ISBN 978-1805127123. Citado 2 vezes nas páginas 14 e 37.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. Rio de Janeiro: Elsevier, 2009. ISBN 978-8535232576. Citado na página 24.

APÊNDICE A – Disponibilização do Código-Fonte

O código-fonte do sistema Schedule-UFU encontra-se disponível em repositórios públicos no GitHub, divididos em duas partes principais:

- **Schedule-UFU (*Frontend* - Angular)**: <https://github.com/grm1003/schedule-ufu>
- **Schedule-Core (*Backend* - Spring Boot e Docker-Compose com PostgreSQL)**: <https://github.com/grm1003/schedule-core>

Cada repositório contém um arquivo `README.md` detalhado, com instruções de execução, descrição das funcionalidades implementadas e explicação da arquitetura utilizada. Esses documentos incluem:

- Orientações de instalação e execução do ambiente;
- Listagem de funcionalidades de autenticação, reservas e administração;
- Estrutura de pastas do projeto;
- Tecnologias utilizadas no desenvolvimento;

A disponibilização pública do código tem como objetivo assegurar transparência, possibilitar reprodutibilidade e incentivar a evolução colaborativa do projeto pela comunidade acadêmica e pelos setores administrativos da UFU.