Sistema de Gerenciamento para Clubes

Matheus Uliana Rossato, Ricardo Frohlich da Silva
Curso de Sistemas de Informação
UFN - Universidade Franciscana
Santa Maria - RS, Brasil
rossato.matheus@ufn.edu.br, ricardo.frohlich@ufn.edu.br

***Resumo*—Este trabalho apresenta o desenvolvimento de um sistema web para gestão de clubes recreativos e esportivos, com o objetivo de apoiar o controle de sócios, reservas de espaços e pagamentos. Muitos clubes ainda realizam esse gerenciamento de forma manual ou em planilhas dispersas, o que dificulta o acesso às informações, gera retrabalho e aumenta a possibilidade de erros. Para mitigar esses problemas, foi proposto e implementado um sistema integrado, acessível via navegador, que centraliza os principais processos administrativos do clube em uma única plataforma. O desenvolvimento foi conduzido utilizando a metodologia Feature-Driven-Development (FDD), organizando o projeto em funcionalidades priorizadas de acordo com as necessidades do clube. A solução foi implementada com back-end em Java Spring Boot, front-end em Angular e banco de dados PostgreSQL, além de integrar serviços externos para envio de e-mails e processamento de pagamentos eletrônicos.**

***Palavras-chave*-Sistemas de Informação, gestão de clubes, FDD, Spring Boot, Angular.**

I. Introdução

As associações esportivas, recreativas e sociais possuem um papel relevante na promoção do lazer, do bem-estar e da integração comunitária, oferecendo aos seus associados uma ampla gama de atividades esportivas, culturais e sociais. Entretanto, apesar da importância desses serviços, muitos clubes ainda enfrentam desafios significativos à gestão de suas operações diárias.

Atualmente, é comum encontrar clubes que utilizam métodos manuais para controle de associados e reservas de espaços, recorrendo a planilhas físicas, documentos impressos ou até mesmo sistemas digitais descentralizados. Essa realidade acarreta diversos problemas, como a dificuldade de manter as informações organizadas e atualizadas, a lentidão no acesso aos dados, o aumento de erros e retrabalho, além de prejudicar a experiência dos próprios associados.

Com o avanço das tecnologias da informação e a crescente demanda por soluções digitais mais eficientes, a informatização dos processos administrativos desses clubes tornou-se uma necessidade estratégica para esses clubes. A adoção de sistemas informatizados possibilita não apenas a otimização do controle e da gestão interna, mas também promove uma maior transparência, agilidade e acessibilidade nas operações, beneficiando tanto a administração quanto os associados.

A. Justificativa

A gestão manual e descentralizada de informações ainda é uma realidade em muitas associações recreativas e esportivas, comprometendo a eficiência e qualidade dos serviços. Nesse contexto, o desenvolvimento de um sistema web específico para a gestão de clubes apresenta-se como uma solução viável e necessária, tendo em vista a melhoria da experiência dos associados e otimização da gestão interna. A informatização dessas atividades não só moderniza os processos como também contribui para a sustentabilidade administrativa da instituição, permitindo um melhor planejamento e gerenciamento.

B. Objetivos

Desenvolver um sistema web para informatizar a gestão de clubes recreativos, esportivos e sociais, permitindo o controle centralizado de associados, reservas de espaços e organização financeira, promovendo maior eficiência no acompanhamento das atividades e organização administrativa.

Para alcançar o objetivo geral, serão necessários os seguintes objetivos específicos:

- Compreender o funcionamento administrativo e operacional de clubes por meio de pesquisas e levantamento de requisitos.
- Analisar e empregar boas práticas de engenharia de software e gerência de projetos.
- Estudar e aplicar conceitos de desenvolvimento, banco de dados e ferramentas de apoio ao projeto.
- Projetar e implementar os módulos principais do sistema.

II. Referencial Teórico

A. Sistemas de Informação

Os Sistemas de Informação (SI) desempenham um papel fundamental no suporte às operações, à gestão e à tomada de decisão nas organizações. Segundo Laudon e Laudon [3] um sistema de informação pode ser definido tecnicamente como um conjunto de componentes inter-relacionados que coletam, processam, armazenam e distribuem informações destinadas a apoiar a tomada de decisões, a coordenação e o controle em uma organização. Esses sistemas são essenciais para transformar dados brutos em informações úteis, permitindo que gestores identifiquem problemas, analisem cenários e desenvolvam soluções mais eficientes.

Além do apoio à administração, os sistemas de informação também exercem papel estratégico, possibilitando às organizações obterem vantagens por meio da melhoria e eficiência operacional, da automação de processos e integração de setores. Para O'Brien e Marakas [8], os SI não apenas aumentam a produtividade, mas também influenciam

diretamente a qualidade dos serviços prestados e a capacidade de inovação da organização.

Dentro da classificação dos sistemas de informação, destacam-se os Sistemas de Informação Gerencial (SIG) e os Sistemas de Apoio à Decisão (SAD), ambos voltados à melhoria do processo decisório nas organizações. Os SIG têm como principal função coletar, processar, armazenar e distribuir informações operacionais e gerenciais, geralmente por meio de relatórios estruturados que auxiliam no controle e no monitoramento das atividades internas. Já os SAD são direcionados ao suporte de decisões mais complexas e menos estruturadas, utilizando ferramentas analíticas, projeções e modelos de simulação para apontar tendências, prever cenários e sugerir soluções.

Destaca-se que alguns sistemas amplamente utilizados no contexto organizacional, como os sistemas de Gestão de Relacionamento com o Cliente *(CRM, Customer Relationship management)* e os sistemas de Planejamento de Recursos Empresariais *(ERP, Enterprise Resource Planning)*, podem ser classificados como SIG, embora também apresentem funcionalidades que se aproximam dos SAD. Essas soluções não apenas organizam e integram dados relevantes das operações, como também oferecem recursos para análises estratégicas que contribuem para decisões mais assertivas. [3] [8]

Desse modo, evidencia-se que as empresas estão sempre tentando melhorar a eficiência de suas operações a fim de conseguir maior lucratividade. Das ferramentas que os administradores dispõem, as tecnologias e os sistemas de informação estão entre as mais importantes para atingir altos níveis de eficiência e produtividade nas operações, especialmente quando combinadas com mudanças no comportamento da administração e nas práticas de negócio.

1) CRM (Customer Relationship Management): O *Customer Relationship Management*, ou Gestão de Relacionamento com o Cliente, é uma abordagem estratégica voltada à compreensão e antecipação das necessidades dos clientes, com o objetivo de gerenciar e analisar as interações com os clientes ao longo do ciclo de vida do relacionamento.

De acordo com Kotler e Keller [6], o CRM envolve a gestão cuidadosa de informações detalhadas sobre os clientes e todos os pontos de contato com eles, de modo a maximizar a fidelização e a satisfação.

Segundo Buttle e Maklan [7] o CRM é uma filosofia de negócios apoiada por tecnologia, cujo foco está na construção de valor mútuo entre a organização e o cliente. Através de sistemas informatizados, é possível integrar dados com o objetivo de criar uma visão unificada dos clientes.

No contexto de clubes, o uso de um sistema CRM pode ser particularmente benéfico para o gerenciamento de sócios, controle de mensalidades, envio de comunicados e monitoramento das participações em eventos.

2) ERP (Enterprise Resource Planning): O *Enterprise Resource Planning*, ou Sistema de Planejamento dos Recursos Empresariais, é um sistema de gestão empresarial que integra todas as áreas de uma empresa, como finanças, vendas, produção, logística, recursos humanos, entre outras, em uma única plataforma. Laudon e Laudon [3] dizem que os sistemas ERP proporcionam uma base tecnológica unificada que centraliza dados e processos organizacionais, promovendo maior eficiência, padronização e integração entre os departamentos.

No contexto de clubes, a utilização de um sistema ERP pode incluir desde o controle financeiro e a gestão de pagamentos até o acompanhamento da participação dos sócios em eventos, manutenção de infraestrutura, e alocação de recursos humanos.

O'Brien e Marakas [8] apontam que os ERPs modernos também incluem módulos para relacionamento com clientes (CRM), tornando-se plataformas completas de gestão. Isso reforça a importância de considerar um ERP personalizado ou adaptado à realidade de clubes, integrando diferentes áreas em um sistema coeso.

3) Segurança e Proteção de Dados: A proteção de dados dos usuários é um aspecto essencial no desenvolvimento de sistemas web, especialmente quando envolvem informações de caráter pessoal, como no caso do sistema proposto para a gestão de clubes recreativos. Informações como nome, cpf, dados de contato, endereço, fazem parte do escopo de dados sensíveis que exigem cuidados técnicos e legais específicos.

No cenário brasileiro, a Lei Geral de Proteção de Dados (LGPD) - Lei n° 13.709/2018 [19] - estabelece normas que regulamentam como os dados pessoais devem ser coletados, armazenados, tratados e descartados. Essa legislação exige que o tratamento dos dados seja feito com base em princípios como transparência, finalidade, necessidade e segurança, garantindo os direitos dos titulares e a responsabilidade do controlador dos dados. [20]

B. Tecnologias

1) Java: Java é uma linguagem de programação orientada a objetos amplamente utilizada no desenvolvimento de aplicações corporativas devido à sua portabilidade, robustez e escalabilidade. Desenvolvida inicialmente pela Sun Microsystems em 1995 e posteriormente adquirida pela Oracle. [9]

Java possui uma vasta biblioteca de APIs e frameworks, além de uma grande comunidade ativa, facilitando o desenvolvimento de aplicações seguras, escaláveis e de facil manutenção.

2) Spring Boot: Spring Boot é um framework baseado no ecossistema Spring que visa simplificar a criação de aplicações Java, ele elimina a necessidade de configurações extensas ao fornecer um conjunto de convenções e dependências pré-configuradas, configurações automáticas, integração facilitada com bancos de dados, segurança, serviços REST e outras tecnologias modernas. [10] Sendo assim, permite que os desenvolvedores criem aplicações robustas e escaláveis com menos esforço, concentrando-se na lógica de negócio em vez de configurações complexas, fazendo com que seja amplamente adotado no desenvolvimento de APIs RESTful e sistemas empresariais. [11]

3) Spring Security: Spring Security é um framework voltado à implementação de autenticação, autorização e proteção contra vulnerabilidades em aplicações Java. Ele fornece uma infraestrutura robusta e extensível para lidar com segurança

em aplicações web, incluindo suporte a autenticação via formulários, tokens JWT(JSON Web Token) e controle de acesso baseado em perfis de usuários. [12]

Uma das principais vantagens do Spring Security é sua integração nativa com o Spring Boot, o que permite adicionar mecanismos de segurança de forma simples e configurável, sem comprometer a escalabilidade ou a flexibilidade da aplicação.

4) Angular: Angular é um framework de desenvolvimento front-end mantido pelo Google, utilizado para a construção de aplicações web dinâmicas, modulares e escaláveis. Baseado na linguagem TypeScript, Angular adota práticas modernas de engenharia de software, como injeção de dependência, componentização, roteamento e binding de dados, o que oferece a separação de responsabilidades e melhora a manutenibilidade do código. [13]

5) PostgreSQL: O PostgreSQL é um sistema gerenciador de banco de dados relacional de código aberto, orientado a padrões SQL e com ênfase em robustez, integridade e extensibilidade. Amadurecido para cargas corporativas, atende de aplicações menores a sistemas de grande porte, com alta confiabilidade, desempenho consistente e grande ecossistema de ferramentas e bibliotecas. [14]

6) Kanban e Trello: O kanban é um método visual para gestão de tarefas que permite controlar o fluxo de trabalho por meio de quadros divididos em colunas. Cada coluna representa um estágio do processo, como ”A Fazer”, ”Em Desenvolvimento”, ”Em Testes”e ”Concluído”. Esse modelo oferece uma visão clara e objetiva do que está sendo feito, do que já foi entregue e do que ainda precisa ser iniciado. [18]

Para implementar esse método, foi utilizado o Trello, uma ferramenta digital baseada no conceito de quadros e cartões. No Trello, cada cartão representa uma funcionalidade ou tarefa, podendo ser movido entre as colunas conforme seu andamento. Além disso, o Trello permite adicionar descrições, etiquetas, datas de entrega, comentários e checklists, permitindo aplicar um modelo de construção por funcionalidades com mais controle, segmentando as entregas e mantendo o foco nas prioridades de cada ciclo de desenvolvimento.

7) FDD: FDD, ou *Feature Driven Development*, é uma metodologia ágil de desenvolvimento de software que visa priorizar entregas de funcionalidades com base em ordens de proridades estabelecidas pelo cliente. A lista de funcionalidades, além de servir como guia para planejamento e desenvolvimento, pode ser utilizada para verificar o progresso do projeto. [15]

Para isso o método é composto por 5 processos que detalham o planejamento e desenvolvimento:

- Desenvolver um Modelo Abrangente, que consiste em realizar um estudo detalhado do domínio do negócio e definir o escopo do projeto;
- Construir uma Lista de Funcionalidades é a etapa em que é feito o levantamento hierárquico de requisitos em conformidade com as necessidades;
- Planejar por Funcionalidades, as funcionalidades identificadas são organizadas em iterações e detalhadas, estabelecendo um plano de projeto que guiará o desenvolvimento;
- Projetar por Funcionalidades, cada funcionalidade é detalhadamente projetada, incluindo a criação de modelos de interface e diagramas, garantindo uma base sólida para a implementação;
- Construir por Funcionalidades, é o último processo, que consiste no desenvolvimento do código de acordo com o plano de projeto. [15]

C. Estrutura de Sistemas Web

1) Front-end: O front-end é a parte do sistema que o usuário enxerga e com a qual interage. Tudo aquilo que aparece na tela do computador ou do celular, como botões, menus, formulários, campos de texto, cores e imagens, faz parte do front-end. Neste trabalho, o front-end é onde o associado irá visualizar os espaços disponíveis no clube, consultar reservas, verificar mensalidades e realizar novas solicitações.

Além de oferecer uma interface agradável, o front-end também precisa estar bem organizado e para isso foi utilizado o framework Angular que facilita a construção de interfaces modernas e responsivas, adaptando-se a diferentes tamanhos de tela.

2) Back-end: O back-end representa a parte interna do sistema, que o usuário não vê. É como o ”motor” por trás da interface. Quando o usuário faz uma ação no front-end, como confirmar uma reserva ou consultar seus dados, é o back-end que executa essas ações, verifica se estão corretas e organiza as informações que devem ser armazenadas ou exibidas. Ele também é responsável por garantir que regras definidas pela administração sejam seguidas, como impedir reservas duplicadas ou calcular corretamente os valores.

Neste projeto, o back-end será implementado com a linguagem Java utilizando o framework Spring Boot, que fornece a estrutura adequada, integração facilitada com o banco de dados e ferramentas de segurança.

3) API (Application Programming Interface): A API funciona como uma ponte entre o front-end e o back-end. É por meio dela que os dados são enviados e recebidos entre as duas partes. Por exemplo, quando o associado clicar em ”verificar reservas”, o front-end envia esse pedido para o back-end por meio da API. O back-end processa a solicitação, busca as informações no banco de dados e devolve a resposta, que será exibida ao usuário.

O uso de APIs facilita a comunicação entre as partes do sistema, tornando possível que o sistema cresça e se conecte a outras ferramentas no futuro. Além disso, ela ajuda a manter o código mais organizado e separado, o que melhora a manutenção.

III. Trabalhos Relacionados

Nesta seção, serão apresentados trabalhos relacionados ao sistema proposto, que possuem características semelhantes ao presente estudo. O objetivo é fornecer uma base de conhecimento que contribua para a elaboração e organização deste projeto.

A. Trabalhos acadêmicos

Nesta subseção serão apresentados trabalhos acadêmicos pesquisados no formato de artigo ou trabalhos de conclusão de curso para serem relacionados a este trabalho.

1) Sistema de Gestão para Clínicas Odontológicas: O trabalho de Frederico [1] propõe o desenvolvimento de um sistema web para a gestão de uma clínica odontológica, permitindo que os profissionais agendem consultas, armazenem dados dos pacientes e gerem relatórios com informações sobre atendimentos e agendamentos.

O objetivo do projeto é planejar e desenvolver uma plataforma web que auxilie no gerenciamento de clínicas odontológicas, garantindo o armazenamento e a organização centralizada de informações essenciais.

Para o desenvolvimento, foram utilizadas as tecnologias NodeJS no back-end, React no front-end e PostgreSQL para o gerenciamento dos dados. Além disso, o projeto seguiu boas práticas do processo de desenvolvimento ágil FDD para garantir uma abordagem estruturada e eficiente.

Embora o domínio do sistema seja distinto, sua proposta é similar à do presente projeto, pois busca substituir métodos convencionais baseados em documentos físicos por uma plataforma digital que centraliza o gerenciamento da clínica e seus dados.

2) Tem Jogo - Uma plataforma Web para empresas de locação de quadras poliesportivas: O trabalho de Arthur [2] propõe o desenvolvimento de uma plataforma web destinada à gestão de empresas que atuam na locação de quadras esportivas. A justificativa para essa iniciativa baseia-se no crescimento anual desse mercado no Brasil, impulsionado pelo aumento da população que busca a prática esportiva. O sistema desenvolvido permite que essas empresas realizem o cadastro e gerenciamento de seus clientes, além de acompanhar o histórico de interações e utilização dos serviços. Dessa forma, buscando oferecer um controle mais eficiente sobre os dados dos usuários e a administração das locações.

As tecnologias escolhidas para o desenvolvimento da plataforma incluem ReactJS para construção de interfaces, PHP para o back-end e MySQL para os dados relacionais. Além disso, foram empregadas metodologias e ferramentas UML para estruturar o desenvolvimento, garantindo uma arquitetura clara, dinâmica e bem definida.

A plataforma Tem Jogo foca na locação de quadras e organização de partidas, apresentando semelhanças com o sistema proposto, uma vez que ambos visam otimizar a gestão dos espaços disponíveis dentro dos clubes, permitindo que os associados façam reservas de maneira prática e organizada. Também, destaca-se a similaridade no modelo operacional da plataforma, que adota a abordagem SaaS (Software as a Service) e incorpora funcionalidades de um CRM.

B. Aplicativos e programas comerciais

Nesta subseção serão apresentados aplicativos ou programas pesquisados que são comercializados e tem relação a este trabalho.

1) Easy Quadras - Aplicativo para Clubes Privados: O Easy Quadras [4] é um aplicativo móvel desenvolvido para facilitar o processo de aluguel e pagamento online de quadras esportivas, atendendo tanto clubes privados quanto usuários em busca de espaços para prática esportiva. Sua proposta é oferecer uma plataforma simples e rápida para reservas, permitindo que os usuários busquem clubes e esportes, agendem horários, comparem preços e realizem pagamentos online.

C. Considerações Sobre os Trabalhos Relacionados

Os trabalhos e sistemas analisados são de grande relevância para o desenvolvimento deste projeto, pois oferecem soluções eficazes para a gestão de clubes e empresas, além de abordagens inovadoras para a locação de espaços. Cada uma dessas plataformas apresenta pontos positivos que servirão como base para a construção do sistema proposto. No entanto, também apresentam limitações que este trabalho se propõe a superar.

A partir dessa análise, o objetivo deste projeto é integrar os aspectos mais vantajosos das soluções existentes, criando um sistema com o diferencial da centralização total da gestão dos clubes. Este sistema buscará atender de forma abrangente à administração de associados, ao controle financeiro e, especialmente, ao gerenciamento eficiente das reservas de espaços, oferecendo uma ferramenta robusta e intuitiva.

IV. Proposta

Este projeto propõe o desenvolvimento de um sistema web voltado à informatização da gestão de clubes recreativos, esportivos e sociais. O objetivo principal é substituir métodos manuais, como planilhas e documentos físicos, por uma solução digital centralizada que ofereça maior eficiência, controle e transparência para a administração, além de melhorar a experiência dos associados na utilização dos serviços oferecidos pelo clube.

O sistema será composto por diferentes módulos, cada um projetado para atender às necessidades específicas dos perfis de usuários envolvidos, como administradores, associados e funcionários. Um dos principais módulos é o de Reserva de Espaços, que possibilitará a visualização da disponibilidade dos espaços, a solicitação e confirmação de reservas pelos associados, bem como o registro automático das informações relacionadas às reservas, como data, horário e responsável. O sistema também enviará notificações à administração e associado sobre a reserva efetuada.

Outro módulo importante será o de Gestão de Associados, que permitirá o cadastro e a atualização de dados dos membros do clube, o controle da situação financeira de cada associado, incluindo as mensalidades em aberto e quitadas, geração de relatórios e definição de permissões de acesso conforme o perfil do usuário.

No módulo Financeiro, será possível realizar o controle de mensalidades e demais taxas cobradas pelo clube, permitindo que os associados acompanhem seus extratos financeiros diretamente pelo sistema, contando ainda com integração a formas de pagamentos online.

Portanto, o sistema será estruturado seguindo a estrutura *full stack*, conforme mostra a Figura 1, com separação entre as camadas de apresentação, lógica de negócio e persistência de dados.

O front-end será desenvolvido com Angular, oferecendo uma interface amigável e responsiva para os usuários. A lógica de funcionamento será tratada no back-end, com a utilização do framework Spring Boot em Java, responsável pelo processamento das regras e integração com o banco de dados.

A comunicação entre essas camadas ocorrerá por meio de API, garantindo organização, segurança e escalabilidade no sistema. O armazenamento dos dados será feito em um banco de dados PostgreSQL, permitindo consultas eficientes e persistência das informações.

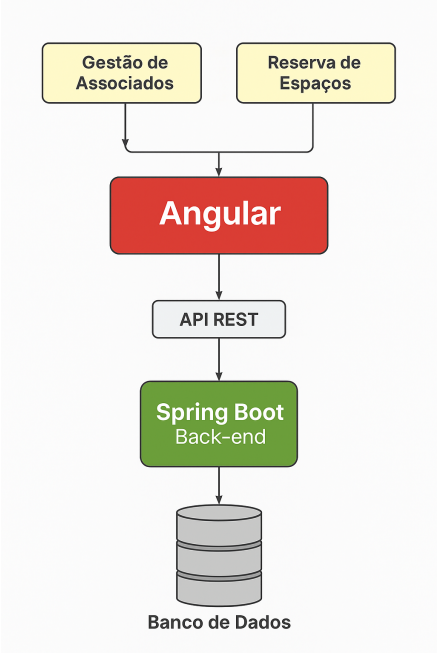


Figura 1: Estrutura Geral do Sistema.

V. Metodologia

Esse trabalho adotou as boas práticas da metodologia ágil FDD aliadas ao uso da linguagem de modelagem UML (Unified Modeling Language). A UML é uma linguagem padronizada e amplamente utilizada para representar visualmente sistemas de software, permitindo a visualização, detalhamento, construção e documentação dos diversos componentes do sistema. [16]

A. Desenvolver um Modelo Abrangente

Esta é a fase inicial da metodologia, na qual se busca compreender o domínio do negócio e definir o escopo do projeto. Para isso, é elaborado o diagrama de domínio, apresentado na Figura 2, focado em entidades dó módulo de reservas de espaços, fornecendo uma visão geral dos principais elementos envolvidos no sistema e as relaçoes relevantes do mundo real que servirão de base para o desenvolvimento da solução.

No contexto do sistema, foram definidos dois perfis principais de usuário: Sócio e Administrador. O Sócio é responsável por realizar ações relacionadas à reserva de espaços, como visualizar horários disponíveis, selecionar o espaço desejado, confirmar a reserva, consultar reservas e, caso necessário, cancelar uma reserva. Esse fluxo foi pensado para ser simples, assim permitindo que o usuário realize suas operações de forma autônoma e eficiente.

Já o o Administrador possui funcionalidades exclusivas de gestão, como o gerenciamento dos espaços e a visualização de relatórios, garantindo uma visão mais ampla da utilização do sistema. Essa separação foi definida para garantir segurança e integridade de informações.

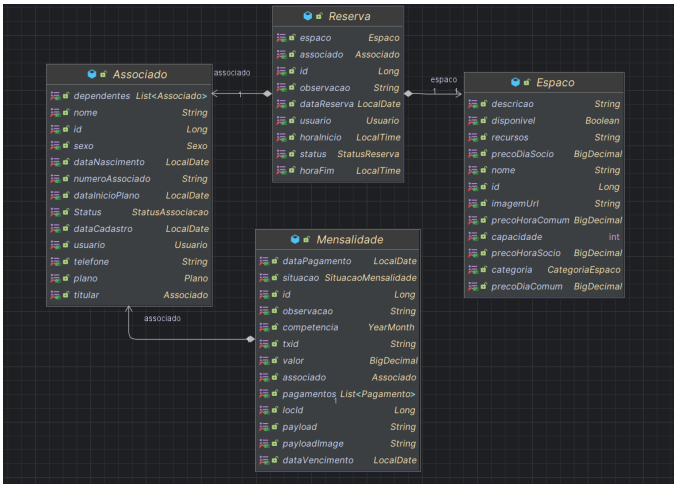


Figura 2: Diagrama de Domínio.

Na Figura 3, apresenta-se um diagrama de atividade que representa o fluxo principal do sistema, evidenciando como os processos do clube serão automatizados, destacando a atividade de realização de reservas. Essa representação permite visualizar como o sistema contribuirá para a organização das operações internas, promovendo maior controle sobre o uso dos espaços.

B. Construir uma Lista de Funcionalidades

Na segunda etapa da metodologia FDD, é realizada a elaboração de uma lista detalhada de funcionalidades, a partir da análise dos requisitos do sistema. Essa lista contempla os requisitos funcionais, Tabela 1, que descrevem os serviços e comportamentos esperados do sistema - ou seja, o que ele deve fazer para atender às necessidades dos usuários.

Além disso, são considerados os requisitos não funcionais, Tabela 2, que dizem respeito a atributos de qualidade, como desempenho, segurança, usabilidade e escalabilidade.

A definição clara dessas funcionalidades é essencial para organizar o desenvolvimento por etapas menores e mais gerenciáveis, conforme recomenda a abordagem orientada a funcionalidades. [17]

C. Planejar por Funcionalidades

A terceira etapa da metodologia FDD consiste em planejar o desenvolvimento do sistema. Com base na lista de funcio-

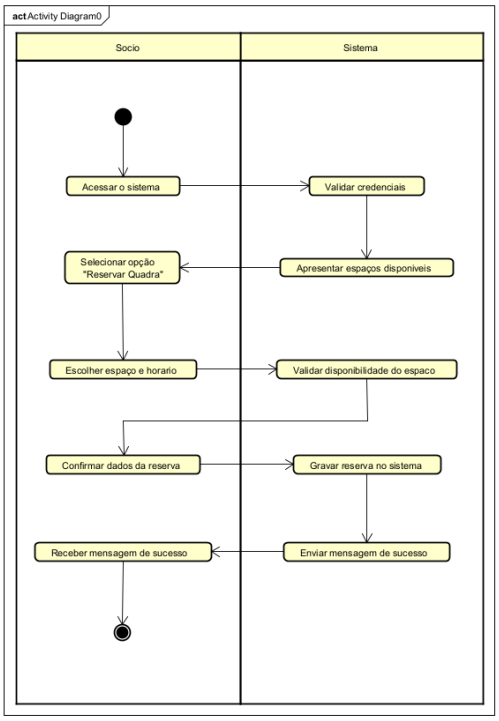


Figura 3: Diagrama de Atividade.

Tabela I

REQUISITOS FUNCIONAIS	
RF01	Permitir o login de sócios e administradores
RF02	Permitir o cadastro, edição e exclusão de sócios (CRUD)
RF03	Permitir que sócios realizem reservas de espaços disponíveis
RF04	Permitir a consulta e listagem das reservas feitas por período e espaço
RF05	Permitir o registro e visualização de pagamentos realizados pelos sócios
RF06	Permitir ao administrador visualizar relatórios de uso dos espaços e receitas do clube
RF07	Enviar notificações aos sócios sobre as confirmações ou pendências de reservas

Tabela II

REQUISITOS NÂO FUNCIONAIS	
RNF01	O sistema será disponibilizado como uma aplicação web
RNF02	O sistema deverá ter uma interface responsiva
RNF03	O sistema deverá permitir a inserção de dados de maneira segura

nalidades previamente estruturada, é definida uma sequência lógica de implementação, considerando a complexidade, dependências e prioridade de cada funcionalidade. Esse planejamento visa otimizar o fluxo de trabalho e garantir entregas incrementais e funcionais ao longo do desenvolvimento. [17]

D. Projetar por Funcionalidades

A quarta etapa compreende a modelagem detalhada do sistema, na Figura 4 temos como exemplo o módulo de Reservas, representado pelo conjunto de Reserva, ReservaRepository, ReservaService, ReservaController e DTOs. A entidade Reserva liga Espaco ao reservante. O ReservaRepository oferece consultas por espaço/usuário e detecção de sobreposição. A ReservaService centraliza regras, valida funcionamento, definições, ações, cálculos e geração da grade de horários. O ReservaController expõe essas operações via API REST. Os DTOs separam domínio e contrato de API.

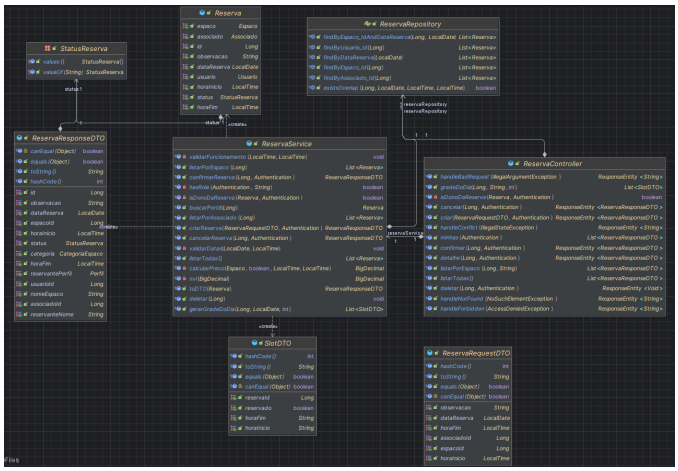


Figura 4: Diagrama de Classes.

E. Construir por Funcionalidades

A etapa de Construir por Funcionalidades é considerada a fase prática mais importante da metodologia FDD. É nesse momento que as funcionalidades previamente planejadas, organizadas e detalhadas nas fases anteriores passam a ser implementadas no sistema. [15]

1) Visão Geral do Ambiente e Arquitetura: O sistema foi implementado seguindo a proposta full-stack definida na fase conceitual: back-end em Java/Spring Boot, front-end em Angular e Postgres como SGBD. A comunicação entre camadas ocorre por API, autenticação via JWT *(Json Web Token)* e autorização por perfis(administrador, sócio, comum). Com base nos requisitos foram implementados CRUDs e consultas com validações de domínio e filtros para Usuários, Espaços, Mensalidades e Reservas, usuários com perfis e autenticação JWT. Mensalidades com geração por competência, controle de status e pagamento via pix. Reservas com grade de horários, prevenção de conflitos, regras por perfil/categoria do espaço e confirmação por email.

2) Funcionalidades: Uma das funcionalidades centrais do sistema é a reserva de espaços, que permite consultar disponibilidade, criar solicitações e acompanhar o status com regras de negócio claras. No método criarReserva(), apresentado no Listing 1, o backend válida campos obrigatórios, ordenação temporal, janela de funcionamento e antecedencia mínima, verifica se o espaço está dísponivel e impede conflitos de horário via consulta de sobreposição. A autorização é aplicada por perfil e categoria do espaço, determinando também o dono da reserva a partir do *Authentication*. O preço é calculado conforme categoria: por blocos de 30/60min em espaços esportivos ou diária em espaços de festa. Após as validações, a reserva é persistida com status inical PENDENTE, podendo ser confirmada pelo ADMIN (disparando um email de confirmação) ou cancelada dentro das janelas definidas. A UI consome ainda uma grade de horários gerada pelo serviço para exibir intervalos livres/ocupados do dia.

```
1      @Transactional
2  public ReservaResponseDTO criarReserva(
       ReservaRequestDTO req, Authentication auth) {
3      // Validacoes de entrada e tratamentos de erro
        omitidos para brevidade
4
5      Espaco espaco = espacoRepository.findById(req.
       getEspacoId()).orElseThrow();
6
7      boolean isAdmin = hasRole(auth, "ROLE_ADMIN");
8      boolean isSocio = hasRole(auth, "ROLE_SOCIO");
9
10     // Definio do dono da reserva (associado ou
       usurio comum)
11     Associado associado = /* busca do associado,
       omitida */;
12     Usuario usuarioComum = /* busca do usurio
       comum, omitida */;
13
14     LocalTime inicio = req.getHoraInicio().
       minusMinutes(BUFFER_MINUTES);
15     LocalTime fim    = req.getHoraFim().
       plusMinutes(BUFFER_MINUTES);
16
17     // Verificao de conflito de horrio
18     boolean conflito = reservaRepository.
       existsOverlap(
19         espaco.getId(), req.getDataReserva(),
       inicio, fim
20     );
21     if (conflito) {
22         throw new IllegalStateException("Ja existe
        reserva nesse horario para este espaco");
23     }
24
25     // Clculo de preo com base no tipo de espao e
       perfil do usurio
26     BigDecimal precoAplicado = calcularPreco(
27         espaco,
28         isSocio || associado != null,
29         req.getHoraInicio(),
30         req.getHoraFim()
31     );
32
33     // Criacao e persistncia da reserva
34     Reserva r = new Reserva();
35     r.setAssociado(associado);
36     r.setUsuario(usuarioComum);
37     r.setEspaco(espaco);
38     r.setDataReserva(req.getDataReserva());
39     r.setHoraInicio(req.getHoraInicio());
40     r.setHoraFim(req.getHoraFim());
41     r.setStatus(StatusReserva.PENDENTE);
42     r.setObservacao(req.getObservacao());
43
44     return toDTO(reservaRepository.save(r));
45 }
```

Listing 1: ReservaService (criarReserva)

A função gerarGradeDoDia, apresentada no Listing 2, é um dos elos lógicos para criar uma reserva, onde ocorre a geração de disponibilidade construindo uma grande diária de intervalos para o espaço em uma data informada. Primeiro é normalizado e o tamanho que cada slot terá e é aplicado os horários de funcionamento padrão. Em seguida, o sistema busca as reservas do dia e calcula a quantidade de slots a partir da duração do expediente. Para cada slot gerado, a ocupação é determinada por regra de sobreposição, onde existe conflito se `tIni < r.fim && tFim > r.inicio`. O retorno é uma lista de SlotDTO como inicio, fim, um flag ocupado e, quando aplicável, o id da reserva que bloqueia o intervalo.

```
1      public List<SlotDTO> gerarGradeDoDia(Long
       espacoId, LocalDate data, int passoMinutos) {
2          final int passo = Math.max(15, Math.min(
       passoMinutos, 240));
3          final LocalTime abre = OPEN_TIME != null ?
        OPEN_TIME : LocalTime.of(8, 0);
4          final LocalTime fecha = CLOSE_TIME != null
        ? CLOSE_TIME : LocalTime.of(22, 0);
5
6          if (!fecha.isAfter(abre)) {
7              return List.of();
8          }
9
10         List<Reserva> reservasDoDia =
       reservaRepository
11                 .findByEspaco_IdAndDataReserva(
       espacoId, data)
12                 .stream()
13                 .filter(r -> r.getStatus() !=
       StatusReserva.CANCELADA)
14                 .toList();
15
16         long totalMin = java.time.Duration.between
       (abre, fecha).toMinutes();
17         if (totalMin <= 0) return List.of();
18
19         int steps = (int) (totalMin / passo);
20         steps = Math.min(steps, 96);
21
22         List<SlotDTO> slots = new java.util.
       ArrayList<>(steps);
23
24         for (int i = 0; i < steps; i++) {
25             LocalTime tIni = abre.plusMinutes((
       long) i * passo);
26             LocalTime tFim = tIni.plusMinutes(
       passo);
27
28             Reserva sobreposta = reservasDoDia.
       stream()
29                     .filter(r -> tIni.isBefore(r.
       getHoraFim()) && tFim.isAfter(r.getHoraInicio
       ()))
30                     .findFirst()
31                     .orElse(null);
32             slots.add(new SlotDTO(
33                     tIni.toString(),
```

```
                    tFim.toString(),
                    sobreposta != null,
                    sobreposta != null ?
    sobreposta.getId() : null
            ));
        }
        return slots;
    }
```

Listing 2: ReservaService (gerarGradeDoDia)

O envio de mensagens foi encapsulado no EmailService, acionado na transição de PENDENTE para CONFIRMADA da reserva no ReservaController.confirmar(). O serviço compõe templates simples com dados do espaço, data e horários, e dispara via SMTP *Simple Mail Transfer Protocol* autenticado usando credenciais e remetentes configurados por variáveis de ambientes. No trecho apresentado no Listing 3, o método realiza a validação do destinatário, define o assunto da email com dados da reserva, é criada uma mensagem em formato HTML e envia a mensagem pelo servidor SMTP.

```

public void enviarConfirmacaoReserva(Reserva r,
    ReservaResponseDTO dto)
        throws MessagingException,
    UnsupportedEncodingException {

    Usuario dest = null;
    if (r.getAssociado() != null && r.getAssociado
    ().getUsuario() != null) {
            dest = r.getAssociado().getUsuario();
        } else if (r.getUsuario() != null) {
            dest = r.getUsuario();
        }
        if (dest == null || dest.getEmail() ==
    null || dest.getEmail().isBlank()) {
            throw new IllegalArgumentException("
    Destinatrio sem e-mail para a reserva " + r.
    getId());
        }

    // Omitidos a Montagem do corpo HTML a partir
    de um template e a obtencao de dados de
    usuario e reserva;

    String subject = "Reserva confirmada #" + r.
    getId();

    // Criao e envio do e-mail em formato HTML
    MimeMessage msg = mail.createMimeMessage();
    MimeMessageHelper helper = new
    MimeMessageHelper(msg, "UTF-8");
    helper.setFrom("desenvolvimento051@gmail.com",
     "AABB Julio de Castilhos");
    helper.setTo(dest.getEmail());
    helper.setSubject(subject);
    helper.setText(html, true);

    mail.send(msg);
}
```

Listing 3: EmailService (enviarConfirmacaoReserva)

O pagamento de mensalidades via pix, apresentado no Listing 4, foi integrado com a API EFÍ implementada como um client dedicado, parametrizado por clientId, clientSecret e certificate. O endpoint /api/v1/pix/pagar cria a cobrança imediate e devolve txid, locId, qrcode e imageQrcode. No front, um modal exibe o QR e inicia polling do status da transação, quando a cobrança é confirmada, o back-end atualiza a mensalidade para PAGA e a tela é recarregada.

```
public JSONObject criarQrCode(PixRequestPayload
    pix) {
        var body = new JSONObject()
                .put("calendario", new JSONObject
    ().put("expiracao", 3600))
                .put("valor", new JSONObject().put
    ("original", normalizaValor(pix.valor())))
                .put("chave", pix.chave());

        try {
            EfiPay efi = new EfiPay(configuracoes)
    ;
            JSONObject cob = efi.call("
    pixCreateImmediateCharge", new HashMap<>(),
    body);

            long locId = cob.getJSONObject("loc").
    getLong("id");
            var params = new HashMap<String,
    String>();
            params.put("id", String.valueOf(locId)
    );
            JSONObject qr = efi.call("
    pixGenerateQRCode", params, new JSONObject());

            return new JSONObject()
                    .put("txid", cob.getString("
    txid"))
                    .put("locId", locId)
                    .put("qrcode", qr.getString("
    qrcode"))
                    .put("imagemQrcode", qr.
    getString("imagemQrcode"));

        } catch (EfiPayException e) {
            return new JSONObject()
                    .put("erro", e.
    getErrorDescription())
                    .put("codigo", e.getError());
        } catch (Exception e) {
            return new JSONObject().put("erro", e.
    getMessage());
        }
    }
```

Listing 4: PixService (criarQrCode)

VI. Resultados Obtidos

Como resultado deste trabalho, foi desenvolvido um sistema web para gestão de clubes que integra cadastro de usuários, administração de espaços, reservas com validações e controle de mensalidades. A solução centraliza as informações dos perfis, garantindo autenticação via JWT, autorizações por papéis e uma experiência coesa entre back-end(Spring Boot) e front-end(Angular) e banco de dados PostgreSQL.

Na Figura 5, a tela de espaços apresenta um catálogo visual dos ambientes disponíveis para uso, organizado em cards responsivos compostos com dados sobre cada espaço. As ações ficam claras, próximas do conteúdo: Ver Detalhes abre informações completas do espaço, enquanto Reservar leva direto ao fluxo de agendamento. O layout privilegia leitura rápida e fácil acesso para a reserva.

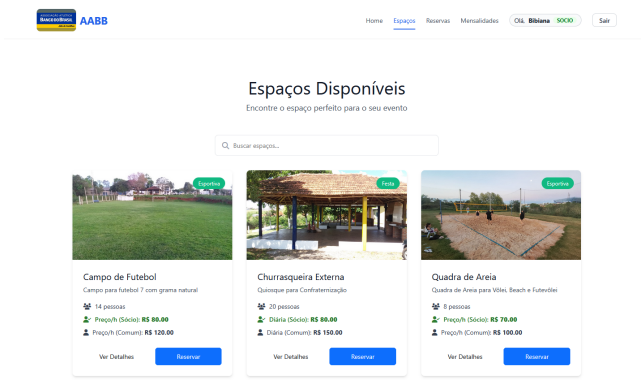


Figura 5: Tela de Espaços.

A tela de Reservar, Figura 6, reúne em um único fluxo as informações do espaço e o agendamento. À esquerda, o usuário visualiza o card do espaço com suas informações. À direita, o painel Informações da Reserva traz o seletor de data, o controle de intervalo (tamanho do slot) e a grade de horários do dia, onde os blocos livres aparecem habilitados e os já ocupados são destacados (ex: 13:00-14:00 ”Reservado”). Ao clicar em um bloco livre, os campos Hora início e Hora fim são preenchidos automaticamente e o usuário ainda pode registrar uma observação. As ações Confirmar Reserva e Voltar finalizam o processo, mantendo o foco em poucos passos.

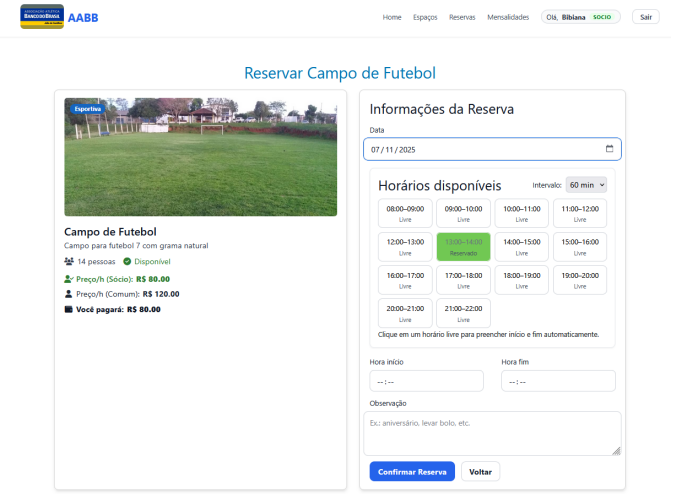
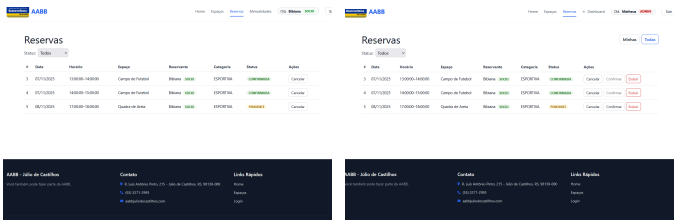


Figura 6: Tela de Reserva.

Na Figura 7, apresentamos a tela de Reservas na visão de sócio e na visão de admin, onde em ambas apresenta a mesma grade de colunas com as informações da reserva, porém com ações condicionadas ao perfil. Na visão do sócio, a listagem exibe apenas as suas reservas, com filtro por status, realce visual de status da reserva e a ação de Cancelar. Já na visão do administrador, a tabela mostra todas as reservas do clube e libera as ações Confirmar, Cancelar e Excluir, permitindo administrar o fluxo ponta-a-ponta. Quando o administrador confirma uma reserva nesta tela, o sistema dispara automaticamente o email de confirmação ao reservante, registrando o novo status da reserva. Em ambas as visões, o design prioriza legibilidade com badges de status, botões contextualizados e feedback após cada operação.



(a) Sócio **(b)** Admin

Figura 7: Telas de Reservas

A tela de Mensalidades, Figura 8, apresenta em formato de tabela as cobranças de mensalidades do sócio por competência, com colunas de competência, vencimento, valor e situação (ABERTA/PAGA/ATRASADA), além da coluna Ações. Para itens em aberto, o botão Pagar aciona um model de Pix que exibe o QR Code da cobrança e o campo de ”Copia e cola”, com botão para copiar o payload. No front-end, após a emissão, é iniciado um polling ao back-end para consultar o status da cobrança e ao receber a confirmação, o modal é fechado e a tabela é recarregada com a situação PAGA, garantindo feedback ao usuário.

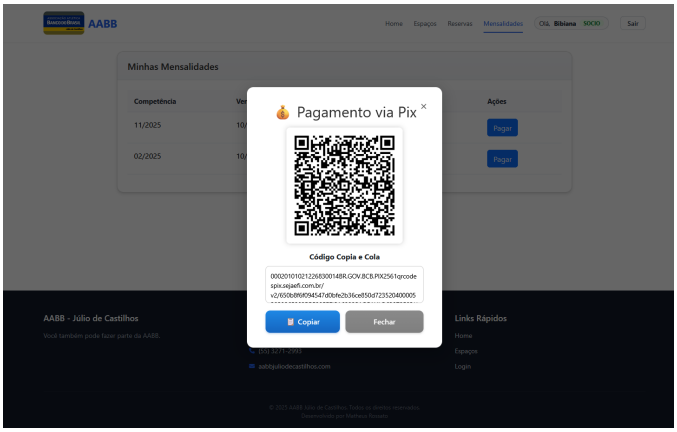


Figura 8: Tela de Mensalidades - Pix.

No Dashboard do Administrador, Figura 9, o sistema reúne os painéis operacionais que sustentam a gestão do clube. Pelo menu Gerenciar, o admin acessa: Usuários, para CRUD de contas e perfis, controle de acesso e ativação. Espaços, com listagem tabular de nome, categoria, capacidade, disponibilidade e preços, além de ações rápidas de Novo, Editar e Excluir. Financeiro, onde acompanha mensalidades por competência, status e pode auditar pagamentos. Gerar Mensalidades, que dispara a criação das cobranças de mensalidades com base nas regras definidas. Em conjunto, esses painéis conferem ao administrador visão centralizada e ações em poucos cliques, reduzindo retrabalho e padronizando processos.

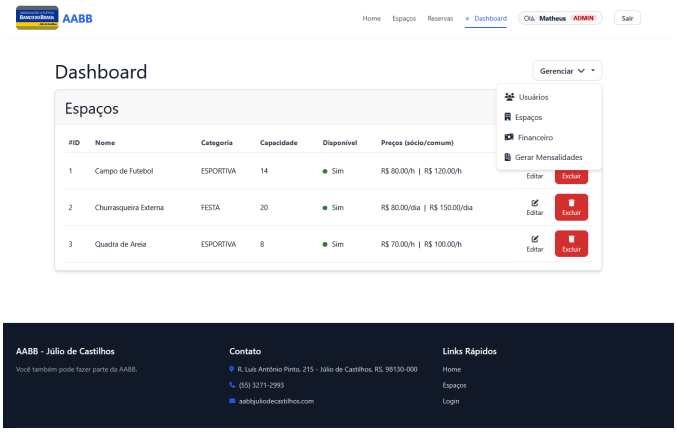


Figura 9: Tela de Dashboards.

VII. Considerações Finais

Este trabalho apresentou o desenvolvimento de um sistema web para gestão de um clube, cobrindo ponta-a-ponta os principais processos: autenticação e perfil, cadastro de usuários e sócios, reservas de espaços com regras de negócio e prevenção de conflitos, controle financeiro com geração de mensalidades e pagamento via Pix, além de painéis administrativos e relatórios operacionais e comunicação por email nas confirmações de reserva. A solução foi construída em estrutura full stack com Spring Boot , Angular e PostgreSQL, expondo API com JWT para autenticação e aplicando validações de domínio e filtros. Do ponto de vista técnico, destacam-se o motor de grade de horários, pagamentos via Pix e disparo de emails transacionais.

A adoção do processo FDD contribuiu para organizar o trabalho em funcionalidades entregáveis, mantendo rastreabilidade entre requisitos e implementação. A cada incremento, priorizou-se clareza das regras, consistência de dados e experiência do usuário, visível nas telas de espaços, reservas, mensalidades e dashboard. Na infraestrutura, escolhas como Postgres, padronização de DTOs e tratamento em controllers e services elevaram a robustez e a manutenibilidade do código.

Como resultado, obteve-se uma aplicação funcional que integra cadastro, reservas, cobrança e comunicação em um único fluxo, reduzindo tarefas manuais e oferecendo visibilidade operacional ao administrador. O sistema atende aos requisitos propostos, mostrando-se apto ao uso e evolução incremental.

Referências

[1] HARTMANN, Frederico G. ”Sistema De Gestão Para Clínicas Odontológicas”(2021)

[2] LOPES, Arthur Reginatto. ”Tem jogo - Uma Plataforma Web para Empresas de Locação de Quadras Esportivas”(2018)

[3] LAUDON, Kenneth C.; LAUDON, Jane P. ”Sistemas de Informação Gerenciais- 11° Edição (2014)

[4] https://www.easycancha.com/pt-BR

[5] . Ramos Braidotti. “Funcionalidades do ERP no padrão indústria 4.0”. Em: (2023).

[6] KOTLER, Philip; KELLER, Kevin Lane. Administração de Marketing. 14. ed. São Paulo: Pearson Prentice Hall, 2012.

[7] BUTTLE, Francis; MAKLAN, Stan. Customer Relationship Management: Concepts and Technologies. 3. ed. New York: Routledge, 2015.

[8] O’BRIEN, James A.; MARAKAS, George M. Sistemas de Informação. 10. ed. São Paulo: McGraw-Hill, 2013.

[9] C. S. Horstmann, Core Java Volume I—Fundamentals, 9th ed. Prentice Hall, 2012

[10] C. Walls, Spring in Action, 5th ed. Manning Publications, 2016.

[11] F. Gutierrez, Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices, Apress, 2020.

[12] M. Ferguson, Spring Security in Action, Manning Publications, 2020.

[13] S. Seshadri, Angular: Up and Running, 2nd ed. O’Reilly Media, 2020.

[14] A. Milani. PostgreSQL - Guia do Programador. Novatec Editora Ltda, 2008

[15] FG Silva, Sandra CP Hoentsch e Leila Silva. “Uma análise das Metodologias Ágeis FDD e Scrum sob a Perspectiva do Modelo de Qualidade MPS. BR”. (2010).

[16] Fundação Bradesco. Linguagem de Modelagem Unificada (UML). Fundação Bradesco, 2016.

[17] Ian Sommerville. Engenharia de Software. Pearson, 2018

[18] G. P. Castilho, Kanban: Aprenda como aplicar o método ágil mais usado no mundo na gestão de equipes e projetos, 2ª ed., São Paulo: Editora DVS, 2018.

[19] Lei Geral de Proteção de Dados Pessoais (LGPD) - Lei n° 13.709/2018. Presidência da República. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm

[20] A. B. Silva, Proteção de Dados Pessoais: A função e os limites do consentimento, 2ª ed., Belo Horizonte: Fórum, 2021.