MindHub: Sistema de Agendamento e Gerenciamento de Consultas Terapêuticas Online - Uma Implementação Prática

Gabrielli Valelia Sousa da Silva
Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerias
gabriellivalelia@gmail.com

***Resumo*—Este trabalho apresenta o desenvolvimento de um sistema web de agendamento de consultas terapêuticas, baseado na arquitetura limpa e na programação orientada a objetos. O objetivo do projeto é conectar psicólogos e pacientes de forma simples e acessível, promovendo a facilidade no agendamento e na gestão de atendimentos psicológicos online.**

I. Introdução

A saúde mental tem se consolidado como uma prioridade global nas agendas de políticas públicas e de pesquisa. Relatórios recentes da Organização Mundial da Saúde (OMS) indicam que mais de um bilhão de pessoas em todo o mundo vivem com algum tipo de transtorno mental, sendo os transtornos de ansiedade e depressão os mais prevalentes [1]. No Brasil, este cenário apresenta desafios semelhantes.

Segundo reportagem da revista Veja Saúde [2], um dos principais fatores que comprometem a continuidade do tratamento psicoterapêutico está relacionado a barreiras práticas, como limitações de tempo e dificuldades financeiras. Nesse contexto, a telepsicologia, ou terapia online, tem se consolidado como uma modalidade de atendimento eficaz e acessível, demonstrando equivalência à terapia presencial em diversos casos e reduzindo barreiras geográficas e temporais [3].

A partir desse avanço, diversas plataformas *healthtech* emergiram como *marketplaces* voltados à saúde mental. Soluções como Vittude [4] e Zenklub [5] comprovam a viabilidade e a alta demanda por sistemas que promovem a conexão entre pacientes e psicólogos.

Inserido nesse contexto, o presente trabalho apresenta o *Mindhub*, um protótipo de sistema web para o agendamento e gerenciamento de consultas terapêuticas online. O objetivo principal é demonstrar uma implementação prática que sirva como estudo de caso da aplicação de conceitos de Programação Orientada a Objetos (POO) no desenvolvimento de uma solução *healthtech*, abrangendo a modelagem de entidades centrais (Paciente, Psicólogo e Consulta) e os casos de uso essenciais do sistema.

Este artigo está estruturado da seguinte forma: a Seção II descreve a metodologia de desenvolvimento, incluindo a arquitetura do sistema e a modelagem das classes. A Seção III apresenta os principais resultados e discute os desafios enfrentados na implementação. Por fim, a Seção IV apresenta as conclusões e propõe possíveis direções para trabalhos futuros.

II. Metodologia

O desenvolvimento do *MindHub* seguiu uma metodologia fundamentada nos princípios da Engenharia de Software Orientada a Objetos, priorizando a separação de responsabilidades, modularidade e escalabilidade. Esta seção descreve os métodos, ferramentas e tecnologias empregados ao longo do processo de desenvolvimento, incluindo o fluxograma de funcionamento, a arquitetura do sistema, a modelagem das entidades e a definição da interface gráfica.

A. Fluxograma do Sistema

O sistema contempla dois fluxos principais de usuários: o fluxo do psicólogo e o fluxo do paciente. Cada fluxo abrange etapas específicas de cadastro, login e utilização das funcionalidades da plataforma:

- *Psicólogo*: realiza o cadastro e login na plataforma, informa suas especialidades e abordagens terapêuticas, define seus horários de atendimento disponíveis, visualiza e gerencia suas consultas agendadas e, adicionalmente, pode contribuir com a comunidade por meio da publicação de conteúdos no blog *MindHub*.
- *Paciente*: realiza o cadastro e login na plataforma, utiliza filtros para encontrar profissionais adequados às suas necessidades, agenda consultas online, visualiza e gerencia seus agendamentos e acessa conteúdos informativos disponibilizados no blog.

A Figura 1 apresenta o fluxograma geral do sistema, descrevendo as principais páginas da interface e as ações realizadas em cada etapa do processo de utilização. Esse diagrama foi elaborado na fase inicial de concepção e serviu como referência para a definição dos casos de uso e da modelagem das entidades descritas nas seções subsequentes.

B. Tecnologias e Infraestrutura

O sistema foi implementado como uma aplicação web *full-stack*, composta por dois módulos principais: *backend* e *frontend*.

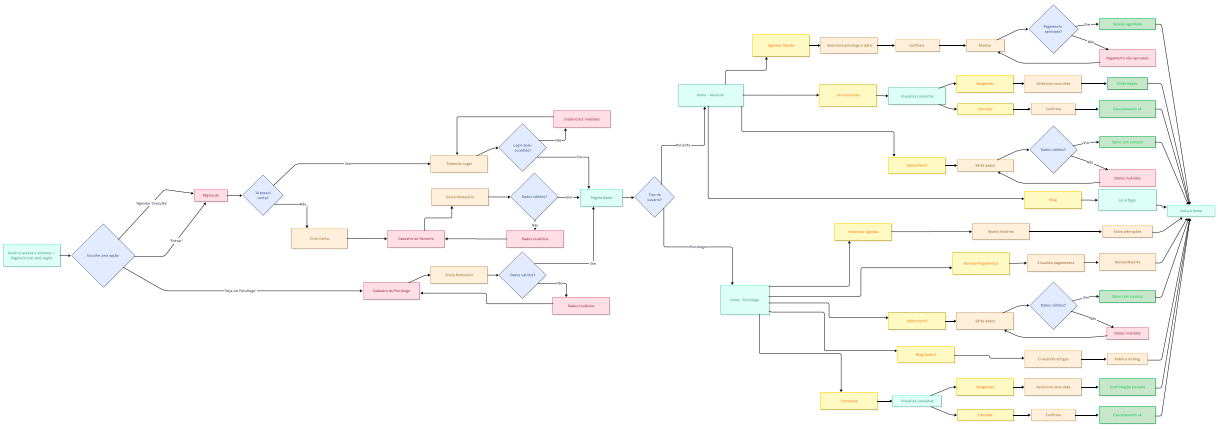

Figura 1. Fluxograma geral do sistema MindHub.

1) Backend: O módulo *backend* foi desenvolvido em Python 3.13, utilizando o framework FastAPI [6] para construção de APIs REST. As principais tecnologias empregadas incluem:

- **FastAPI**: framework assíncrono de alto desempenho para APIs RESTful, com validação automática de dados (Pydantic) e documentação interativa (OpenAPI/Swagger);
- **Beanie**: ODM (*Object-Document Mapper*) assíncrono para MongoDB, que permite o mapeamento entre classes Python e documentos persistidos;
- **MongoDB**: banco de dados NoSQL orientado a documentos, adequado para armazenar estruturas complexas como perfis de psicólogos e suas disponibilidades;
- **Redis**: sistema de cache em memória utilizado para gerenciamento de sessões e tokens de autenticação;
- **Dishka**: contêiner de injeção de dependências que promove baixo acoplamento e facilita a testabilidade;
- **PyJWT**: biblioteca para geração e validação de tokens JWT (*JSON Web Tokens*), viabilizando autenticação *stateless*;
- **Bcrypt**: algoritmo de hash criptográfico para armazenamento seguro de senhas.

O ambiente de desenvolvimento foi gerenciado com `uv`, um gerenciador de pacotes e ambientes virtuais de alta performance. A infraestrutura de banco de dados foi containerizada via Docker Compose, assegurando portabilidade e replicabilidade do ambiente.

2) Frontend: O *frontend* foi desenvolvido utilizando React 19.1.1 com JavaScript moderno (ES6+), seguindo a arquitetura de Single Page Application (SPA). Nesta abordagem, a aplicação é carregada uma única vez no navegador, e as transições entre diferentes telas são gerenciadas pelo React Router DOM, sem necessidade de recarregar a página completa. As principais bibliotecas utilizadas foram:

- **React**: biblioteca baseada em componentes reutilizáveis para construção de interfaces de usuário;
- **React Router DOM**: controle de rotas e navegação no cliente;
- **Material-UI (MUI)**: biblioteca de componentes baseada no Material Design, garantindo responsividade e consistência visual;
- **TanStack Query**: gerenciamento de estado do servidor e cache de requisições;
- **React Hook Form + Zod**: manipulação e validação de formulários com esquemas tipados;
- **Zustand**: gerenciamento de estado global de forma simples e performática;
- **Axios**: cliente HTTP para comunicação com o *backend*;
- **Styled Components**: estilização de componentes via CSS-in-JS.

O projeto empregou o Vite como ferramenta de *build*, possibilitando recarregamento rápido durante o desenvolvimento e otimizações de desempenho para produção.

C. Arquitetura do Sistema

A arquitetura do *backend* foi estruturada segundo os princípios da *Clean Architecture* [7]. Essa abordagem organiza o sistema em camadas concêntricas, cada uma com responsabilidades bem definidas, de modo a proteger as regras de negócio de detalhes técnicos como frameworks e bancos de dados. O resultado é um sistema mais flexível, testável e de fácil manutenção.

A Figura 2 apresenta o diagrama de camadas da Arquitetura Limpa, ilustrando a separação entre domínio, casos de uso e infraestrutura.

1) Camada de Domínio (`domain/`): Essa camada representa o núcleo da aplicação, contendo entidades, objetos de valor e regras de negócio. É completamente independente de frameworks e define o comportamento essencial do sistema.

As principais entidades são:

- **Entity**: classe base que define o conceito de entidade com identidade única (*Identity Pattern*);
- **User**: classe abstrata representando um usuário genérico, com atributos como nome, e-mail, CPF, telefone, data de nascimento e cidade;
- **Patient**: subclasse de `User` representando pacientes;

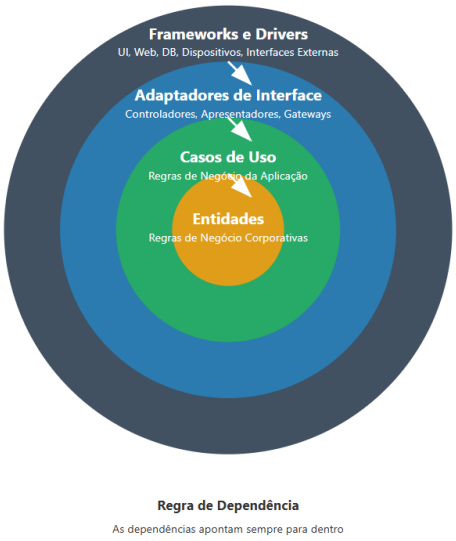

Figura 2. Diagrama de camadas da Arquitetura Limpa [8].

- **Psychologist**: subclasse de `User` representando psicólogos, com atributos adicionais como CRP, especializações e valor de consulta;
- **Appointment**: representação de uma consulta agendada, com data, horário, psicólogo responsável, paciente e status;
- **Availability**: disponibilidade de horários do psicólogo;
- **PixPayment**: pagamentos via Pix, com QR Code, chave e status;
- **Content**: artigos e conteúdos publicados por psicólogos.

Os objetos de valor encapsulam validações e garantem imutabilidade, como `Email`, `CPF`, `Password` e `PhoneNumber`. A classe `Entity` utiliza o padrão *Identity*, garantindo unicidade por meio de `UniqueEntityId`. As regras de negócio são aplicadas em métodos internos, como `schedule()` na classe `Availability`, que valida disponibilidade antes do agendamento.

2) Camada de Aplicação (`application/`): Essa camada coordena os casos de uso e a comunicação entre o domínio e a infraestrutura. Cada operação é implementada como um caso de uso específico, seguindo o padrão *Use Case*.

Os principais casos de uso incluem:

- **Paciente**: `CreatePatient`, `UpdatePatient`, `GetPatient`;
- **Psicólogo**: `CreatePsychologist`, `UpdatePsychologist`, `GetPsychologists`;
- **Consulta**: `ScheduleAppointment`, `ConfirmAppointment`, `CancelAppointment`, `CompleteAppointment`, `RescheduleAppointment`;
- **Disponibilidade**: `CreateAvailabilities`, `DeleteAvailability`;
- **Conteúdo**: `CreateContent`, `UpdateContent`, `DeleteContent`, `GetContents`;
- **Sessão**: `Login`, `GetCurrentUser`.

Essa camada também define os DTOs (*Data Transfer Objects*) e as interfaces de repositórios, promovendo o princípio da Inversão de Dependência (DIP).

3) Camada de Infraestrutura (`infra/`): A camada de infraestrutura contém as implementações concretas de persistência, autenticação e controladores HTTP. É a camada mais externa e depende das camadas internas, mas nunca o contrário.

Principais componentes:

- **Persistence**: repositórios Beanie/MongoDB para persistência e mapeamento entre entidades e documentos;
- **HTTP**: controladores FastAPI responsáveis por expor endpoints REST e orquestrar casos de uso;
- **Config**: arquivos de configuração (MongoDB, Redis, JWT e variáveis de ambiente);
- **Providers**: injeção de dependências e integração com serviços externos.

O fluxo segue: Requisição HTTP → Controlador → Caso de Uso → Repositório → Banco de Dados, e o retorno ocorre no sentido inverso. Para exemplificar, a Figura 3 mostra o fluxo para o caso de uso de criar um Paciente.

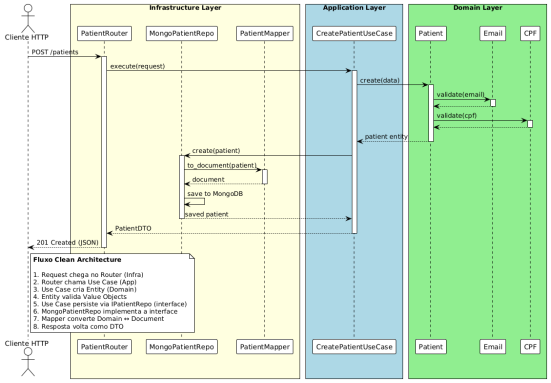

Figura 3. Exemplo de fluxo completo: Criar Paciente

D. Modelagem Orientada a Objetos

A modelagem aplicou os princípios fundamentais da Programação Orientada a Objetos:

- **Encapsulamento**: controle de acesso via propriedades, garantindo consistência de estado;
- **Herança**: hierarquia `User` → `Patient`/`Psychologist`, compartilhando comportamento comum;
- **Polimorfismo**: implementação do método abstrato `get_user_type()` de forma distinta nas classes `Patient` e `Psychologist`, que herdam de `User`. Também presente nas múltiplas implementações de interfaces de repositório (`IPatientRepo`, `IUserRepo`), com versões em memória para testes e persistência em MongoDB;
- **Composição**: entidades compostas, como `Appointment`, que contém `Patient`, `Psychologist` e `PixPayment`.

Foram empregados padrões de projeto clássicos: *Repository*, *Value Object*, *Use Case*, *Dependency Injection* e *DTO*. O diagrama UML completo do sistema encontra-se no Apêndice A.

E. Desenvolvimento da Interface Gráfica

O *frontend* adota o padrão de componentes *Container/Presentational*, organizando o código em:

- **Pages**: páginas correspondentes às rotas;
- **Components**: componentes reutilizáveis (menus, modais, formulários);
- **Services**: comunicação com a API;
- **Stores**: estado global (autenticação, notificações);
- **Utils**: funções auxiliares e rotas protegidas.

A autenticação é baseada em JWT armazenados localmente, com interceptadores Axios para inserção automática do token. O sistema de notificações, implementado com Zustand, fornece mensagens de sucesso, erro e informação em toda a aplicação.

As principais páginas são: Home, Login/Registro, Buscar Psicólogos, Agendar Consulta, Minhas Consultas, Adicionar Disponibilidades, Conteúdos, Escrever Conteúdo e Editar Perfil. Toda a interface é responsiva e adaptável a diferentes dispositivos, utilizando componentes flexíveis do MUI e *media queries* CSS.

F. Tratamento de Dados Temporais

O gerenciamento de fusos horários foi tratado de forma centralizada: o *backend* armazena datas em UTC, enquanto o *frontend* opera no fuso local (GMT-3). Conversões são realizadas nas fronteiras do sistema:

- Envio de dados: `date.toISOString()` converte automaticamente para UTC;
- Recebimento de dados: `new Date(isoString)` converte para o fuso local do navegador.

As estratégias metodológicas descritas nesta seção, desde a definição da arquitetura até a modelagem orientada a objetos, serviram de base para a implementação prática do sistema. Os resultados obtidos a partir dessas decisões são apresentados e discutidos na Seção III.

III. Resultados e Discussão

Esta seção apresenta os resultados obtidos com o desenvolvimento do sistema *MindHub*, com ênfase na verificação das funcionalidades implementadas e na validação dos dados processados. O objetivo é demonstrar o correto funcionamento do sistema em conformidade com os requisitos definidos na Seção II.

A. Visão Geral do Sistema Implementado

O sistema *MindHub* foi desenvolvido com base nos princípios da Arquitetura Limpa e da Programação Orientada a Objetos (POO). A aplicação encontra-se funcional, permitindo a interação completa entre os dois perfis de usuários: psicólogos e pacientes.

As principais funcionalidades implementadas e testadas foram:

- Cadastro e autenticação de pacientes e psicólogos;
- Edição de perfis e atualização de informações cadastrais;
- Definição e gerenciamento de horários disponíveis por psicólogos;
- Agendamento, cancelamento e reagendamento de consultas por pacientes;
- Confirmação de consultas e gerenciamento de agenda pelo psicólogo;
- Publicação e leitura de conteúdos no blog integrado à plataforma.

Todas as rotas e interações entre as camadas de aplicação e infraestrutura foram validadas com sucesso, garantindo a consistência entre as operações do frontend e as respostas do backend. As subseções a seguir apresentam dois dos fluxos testados, com ênfase na experiência dos dois perfis principais do sistema.

1) Fluxo de Login de Psicólogo e Gerenciamento de Horários: Ao acessar o sistema, o usuário é inicialmente direcionado para a página inicial *Pre Login*(Figura 4). Dessa página, conforme ilustrado no fluxograma geral do sistema (Figura 1), é possível navegar para a página de cadastro ou para a página de login. Para este fluxo, o caminho seguido foi o de login do psicólogo.

A Figura 5 mostra o comportamento do sistema diante de uma tentativa de login com credenciais inválidas, enquanto a Figura 6 apresenta o alerta de sucesso e o redirecionamento automático para a página *Home* após uma autenticação válida.

Na página *Home*, o layout é adaptado de acordo com o tipo de usuário, exibindo informações personalizadas para psicólogos e pacientes. Ao clicar no menu do cabeçalho (*Header*), o psicólogo tem acesso à opção "Gerenciar Disponibilidades", como mostrado na Figura 7. Selecionando essa opção, o usuário é direcionado para a página de gerenciamento de horários (Figura 8), onde pode configurar sua agenda conforme as regras abaixo:

- São exibidos intervalos de uma hora, das 05h00 às 23h00, para todos os dias da semana, permitindo que o psicólogo selecione os horários disponíveis;
- Consultas têm duração pré-definida de 50 minutos;
- Não é possível selecionar dias ou horários já passados;
- Não é possível remover horários que já possuam consultas não canceladas associadas.

Ao concluir a configuração, basta clicar no botão "Salvar" para que a requisição correspondente seja enviada ao *backend*, atualizando as informações no banco de dados. O *backend* realiza validações adicionais para garantir integridade e segurança. Em caso de sucesso, é exibido um alerta positivo e o psicólogo é redirecionado para a *Home*. Em caso de falha, o sistema emite um alerta de erro acompanhado de uma mensagem descritiva, orientando o usuário sobre o problema.

2) Fluxo de Cadastro de Paciente e Agendamento de Consulta: Partindo novamente da página inicial, apresentada na Figura 4, ao clicar no botão *"Agende uma consulta"*, o usuário

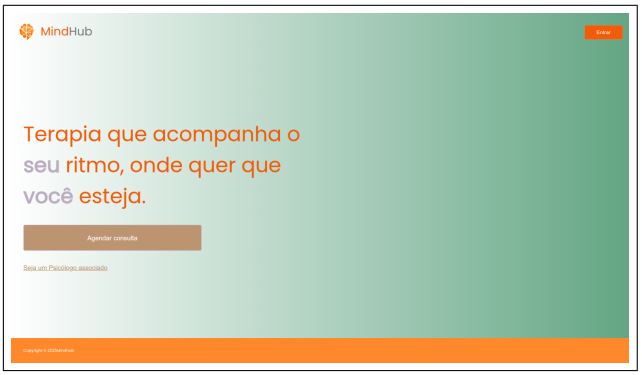

Figura 4. Página inicial (pré-login) do sistema MindHub.

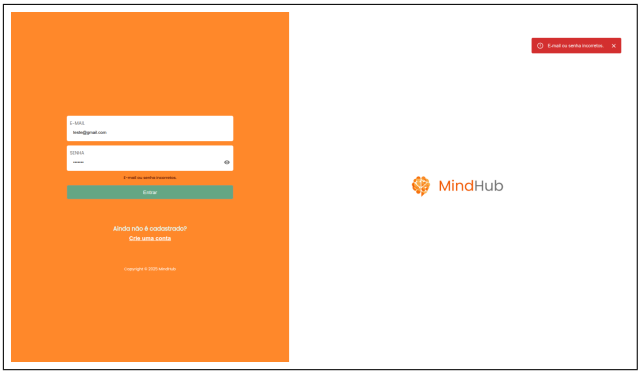

Figura 5. Tratativa de erro para login com credenciais inválidas.

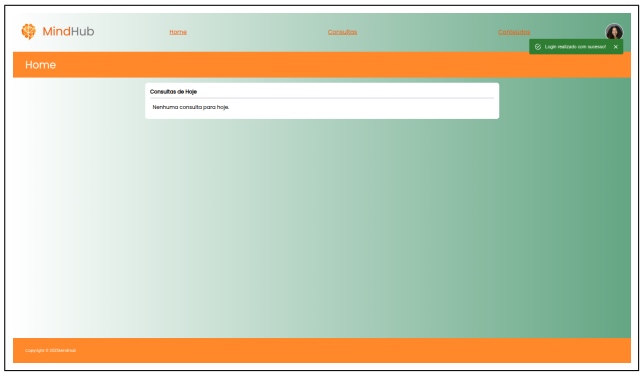

Figura 6. Alerta de sucesso e redirecionamento após login válido.

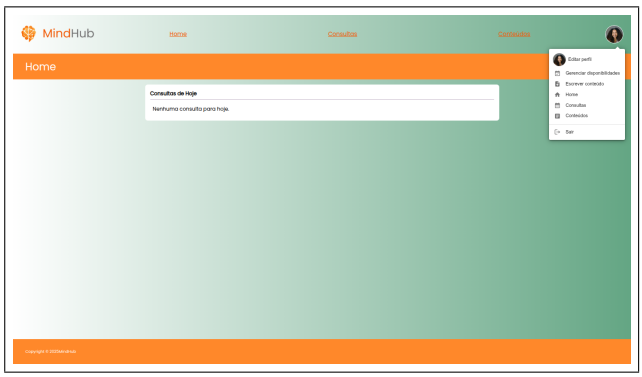

Figura 7. Menu do cabeçalho exibindo a opção de gerenciamento de horários.

é direcionado para a página de login. Nessa tela, é possível acessar a opção *"Crie uma conta"*, que leva o usuário à página de cadastro.

Durante o preenchimento do formulário, são realizadas diversas validações no *frontend*, implementadas com a biblioteca `Zod`. A Figura 9 ilustra alguns exemplos de validações — como verificação de formato de e-mail, tamanho mínimo de senha e confirmação de campos obrigatórios. Ao clicar em *"Cadastrar"* com todos os dados validados, é feita a requisição ao *backend* para criação de um novo registro de paciente.

Em caso de sucesso, é exibido um alerta positivo, e o usuário é redirecionado para a *Home* (Figura 10), cuja interface é personalizada para o perfil de paciente. Em caso de erro, o sistema exibe um alerta de falha com uma mensagem descritiva e intuitiva, orientando o usuário a corrigir o problema.

A partir da página *Home*, ao clicar no botão *"Agendar consulta"*, o paciente é redirecionado para a página de busca e seleção de profissionais (Figura 11). Nessa tela, são listados psicólogos cadastrados com diferentes especialidades, abordagens, públicos atendidos e valores por sessão. Além disso, o paciente dispõe de filtros dinâmicos para refinar sua busca conforme suas preferências.

Após selecionar o profissional desejado, o paciente escolhe um horário disponível e clica em *"Agendar"*. Nesse momento, é exibido um modal de confirmação contendo um resumo da consulta — incluindo nome e abordagem do psicólogo, data, horário e valor — conforme ilustrado na Figura 12.

Ao clicar em *"Prosseguir para pagamento"*, o sistema redireciona o paciente para a página de pagamento (Figura 13). Nessa página, são gerados um código e um QR Code *PIX* para simular o processo de pagamento. É importante destacar que, nesta versão do sistema, tais dados têm caráter apenas ilustrativo, não realizando transações reais.

Após a confirmação do pagamento (por meio do botão *"Concluído"*), a consulta é automaticamente adicionada à lista de compromissos tanto do paciente quanto do psicólogo, com o status *"Aguardando confirmação"* (Figura 14). Os passos subsequentes incluem a confirmação do recebimento do pagamento pelo psicólogo e, posteriormente, a realização da sessão.

B. Validação das Funcionalidades

Durante a fase de testes funcionais, foram realizadas simulações dos fluxos descritos no fluxograma do sistema (Figura 1), verificando o comportamento esperado em cada etapa. As validações incluíram:

- Verificação de credenciais durante o processo de login e controle de acesso;
- Validação de campos obrigatórios nos formulários de cadastro e atualização de dados;
- Bloqueio de agendamentos em horários já ocupados ou fora da disponibilidade do psicólogo;

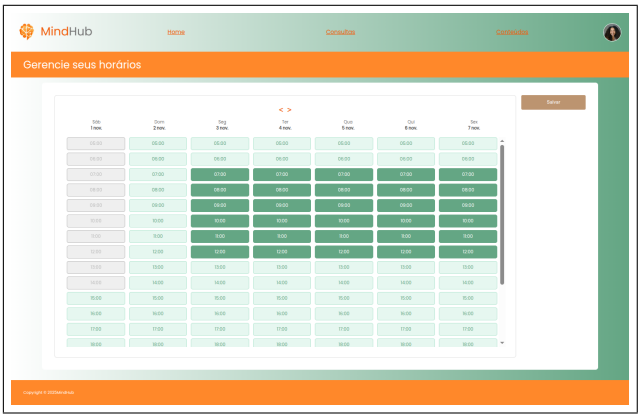

Figura 8. Página de gerenciamento de disponibilidades do psicólogo.

Figura 9. Exemplo de validações no frontend durante o cadastro (implementadas com Zod).

- Confirmação e atualização do status das consultas (pendente, confirmada, cancelada);
- Persistência correta das informações no banco de dados MongoDB, com integridade mantida entre entidades relacionadas.

Em todos os casos, os resultados corresponderam ao comportamento esperado. Os dados inseridos foram devidamente validados no frontend por meio de esquemas `Zod`, e novamente validados no backend através das classes `DTOs` e dos modelos Pydantic. Esse duplo processo de validação garantiu a integridade e a segurança dos dados trafegados entre cliente e servidor.

C. Discussão dos Resultados

Os testes realizados demonstraram que o sistema cumpre corretamente os requisitos definidos e que a arquitetura adotada favorece o isolamento entre camadas e a clareza do código. A validação de dados em múltiplos níveis (frontend e backend) contribuiu significativamente para a prevenção de erros de entrada e inconsistências de persistência.

O uso de princípios de Orientação a Objetos mostrou-se eficaz na organização das classes e na extensão de funcionalidades. A estrutura modular resultante facilita a manutenção e futuras expansões do sistema, como a integração com novos métodos de pagamento ou serviços externos.

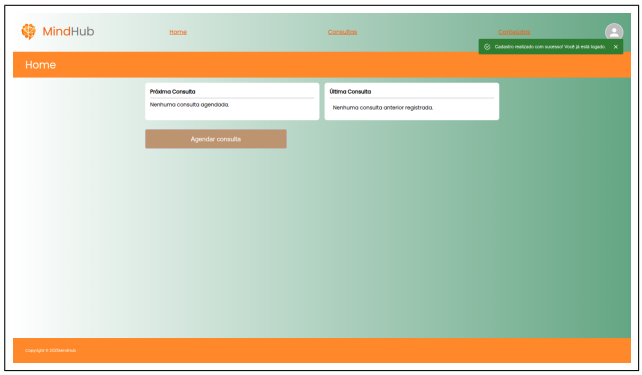

Figura 10. Página home personalizada para o perfil de paciente.

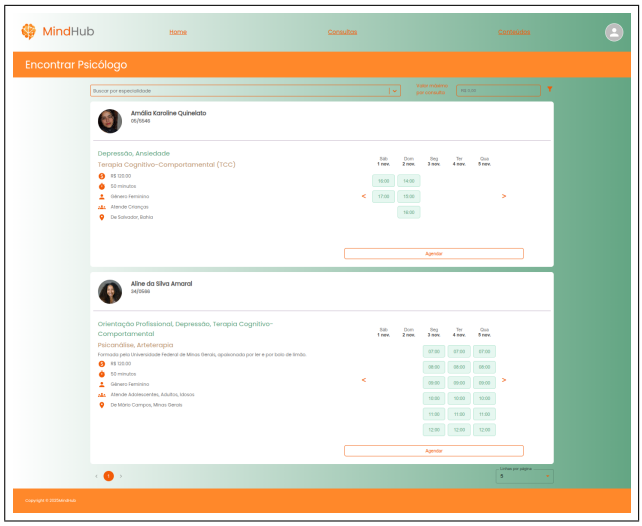

Figura 11. Lista de psicólogos disponíveis com filtros dinâmicos de busca.

Assim, o sistema *MindHub* demonstrou funcionamento consistente, validação robusta dos dados e aderência às boas práticas de desenvolvimento orientado a objetos, confirmando a viabilidade técnica da proposta.

IV. Conclusões e Trabalhos Futuros

A. Conclusões

Este trabalho apresentou o desenvolvimento do protótipo funcional do MindHub, um sistema web de agendamento de consultas terapêuticas. O objetivo principal, conforme definido na Introdução, foi utilizar este projeto como um estudo de caso prático para a aplicação da Programação Orientada a Objetos (POO) na modelagem de um domínio de *healthtech*.

Conclui-se que o objetivo foi alcançado com sucesso. A modelagem das entidades centrais — `Paciente`, `Psicólogo` e `Consulta` — e a implementação dos casos de uso essenciais (Épicos 1, 2 e 3) demonstraram a eficácia da POO para organizar a complexidade do domínio. O resultado é um sistema coeso, modular e funcional, que valida a arquitetura de software e o design de classes propostos na Metodologia.

Como um estudo de caso acadêmico, o projeto cumpriu seu papel ao implementar o fluxo central de um *marketplace* de

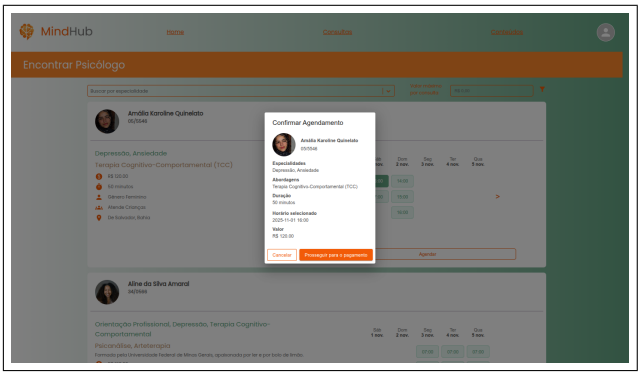

Figura 12. Modal de confirmação de agendamento de consulta.

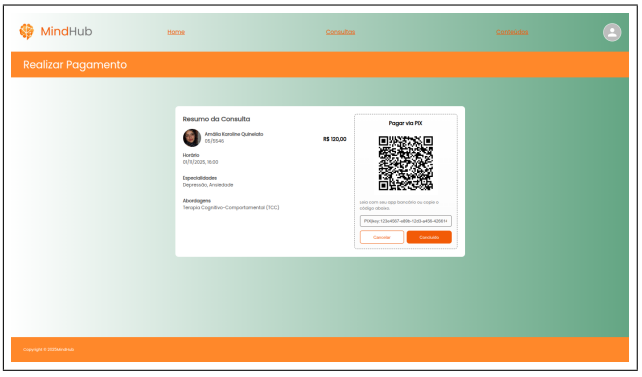

Figura 13. Página de simulação de pagamento via PIX.

saúde mental. As principais limitações do trabalho residem no escopo do protótipo: funcionalidades críticas para um produto real, como o sistema de pagamento e a geração de links de vídeo, foram simuladas para focar o esforço de desenvolvimento na arquitetura e na modelagem.

B. Trabalhos Futuros

Com base no protótipo desenvolvido e nas limitações identificadas, uma série de evoluções e trabalhos futuros podem ser propostos para expandir a funcionalidade e a robustez do MindHub:

- **Sistema de avaliação de psicólogos:** Implementar um mecanismo que permita ao paciente registrar uma avaliação sobre o profissional após a consulta, de modo que essas informações fiquem disponíveis na listagem de psicólogos, contribuindo para a transparência e a melhor tomada de decisão por parte de novos pacientes.
- **Integração com Gateway de Pagamento Real:** Substituir a simulação de QR Code PIX por uma integração completa com uma API de pagamento (como Stripe ou PagSeguro), automatizando o fluxo financeiro e a mudança de status das consultas.
- **Evolução do Sistema de Recomendação:** Implementar um algoritmo de ”match”mais sofisticado, superando a busca por palavras-chave diretas e utilizando um sistema de *tags* ponderadas ou explorando técnicas de filtragem colaborativa baseadas no histórico de agendamentos.

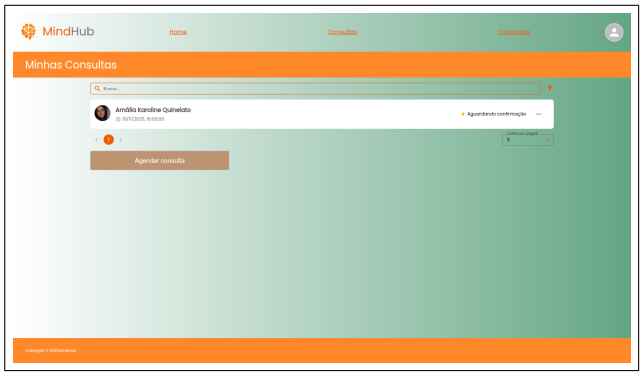

Figura 14. Página de consultas.

- **Integração com APIs de Terceiros:** Implementar a integração com as APIs do Google Calendar e Google Meet, automatizando a criação de eventos na agenda dos usuários e a geração de links únicos para as sessões online.
- **Expansão do Módulo de Conteúdo (Blog):** Incluir um fluxo de revisão e aprovação por parte de um administrador.
- **Conformidade com a LGPD e Segurança:** Realizar uma auditoria de segurança completa e implementar todas as diretrizes da Lei Geral de Proteção de Dados (LGPD), uma etapa crítica dado o manuseio de dados sensíveis de saúde.
- **Desenvolvimento de Aplicação Móvel:** Criar uma versão *mobile* da plataforma, utilizando tecnologias híbridas ou nativas, para melhorar a acessibilidade e a experiência do usuário em dispositivos móveis.

APÊNDICE A
DIAGRAMA UML COMPLETO

Devido à complexidade e à quantidade de classes resultantes da aplicação dos princípios da Arquitetura Limpa, o diagrama UML completo do sistema *MindHub* foi dividido em seções, de modo a facilitar a leitura e a compreensão das relações entre as classes.

A. Camada de Domínio

As Figuras 15, 16 e 17 apresentam, respectivamente, os diagramas UML dos núcleos **Usuário**, **Agendamento** e **Conteúdo**, que compõem o domínio da aplicação.

A classe abstrata *ValueObject*, destacada nos diagramas, foi mantida propositalmente sem atributos ou métodos. Essa decisão de projeto visa permitir um agrupamento sintático e semântico — qualquer classe que herda de *ValueObject* indica tratar-se de um objeto de validação (como `Email`, `CPF`, `Password`), e não de uma entidade de negócio, que herda de *Entity*. Essa distinção reforça a clareza conceitual entre *entidades* e *objetos de valor* dentro do domínio.

B. Camada de Aplicação

As Figuras 18, 19, 20, 21 e 22 detalham a camada de aplicação, que orquestra os casos de uso e implementa os

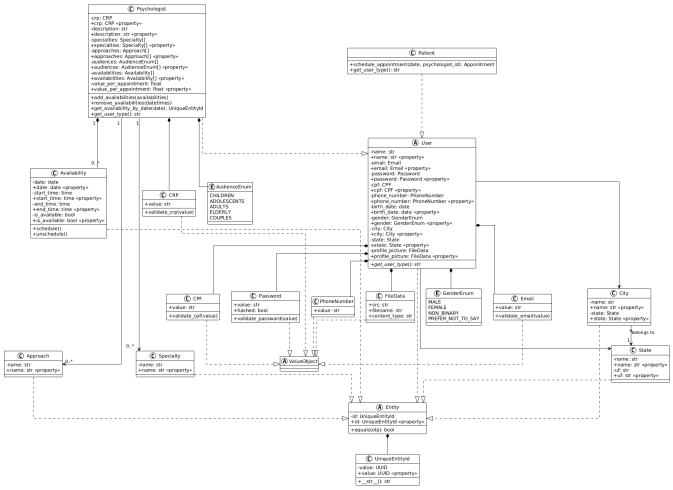

Figura 15. Camada de Domínio — Núcleo Usuário

fluxos de negócio. Cada núcleo representa um conjunto de casos de uso relacionados, de acordo com o princípio de coesão de responsabilidade.

C. Camada de Infraestrutura

Por fim, a Figura 23 apresenta uma visão simplificada da camada de infraestrutura, que abrange os componentes responsáveis pela persistência de dados, exposição de endpoints HTTP e integração com serviços externos.

APÊNDICE B
REPOSITÓRIO E EXECUÇÃO DO PROJETO

O código-fonte do projeto *MindHub* está disponível em:

- *Frontend*: https://github.com/gabriellivalelia/mindhub-frontend;
- *Backend*: https://github.com/gabriellivalelia/mindhub-backend.

As instruções de execução estão descritas no arquivo `README.md` do respectivo repositório.

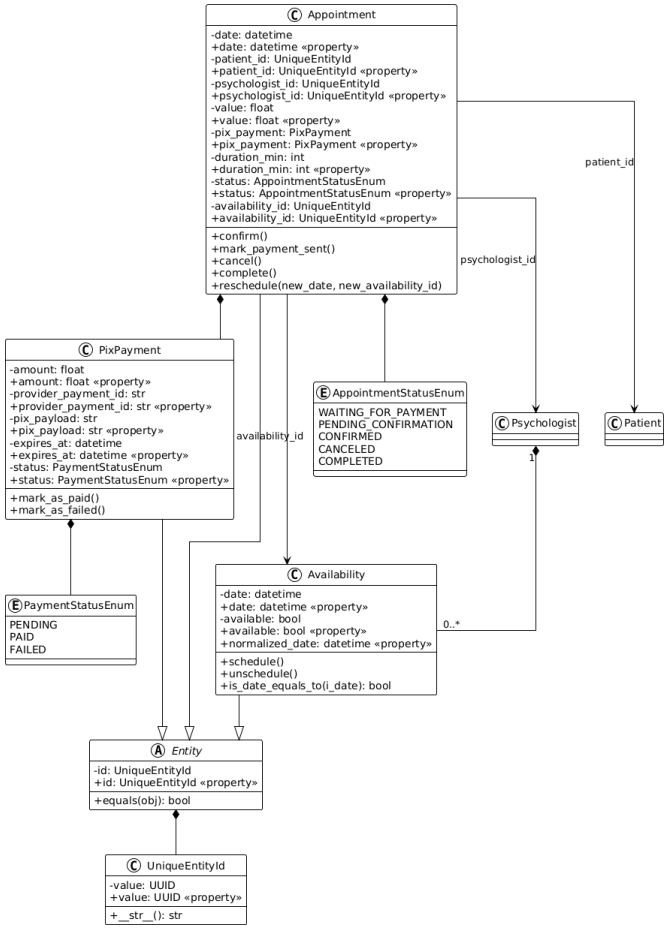

Figura 16. Camada de Domínio — Núcleo Agendamento

REFERÊNCIAS

[1] O. M. da Saúde (OMS), "Over a billion people living with mental health conditions – services require urgent scale-up." https://www.who.int/news/item/02-09-2025-over-a-billion-people-living-with-mental-health-conditions-services-require-urgent-scale-up, Sep 2025. Acessado em: 30 de outubro de 2025.

[2] V. Saúde, "A constância na terapia ainda é um grande desafio para os pacientes," 2023. Acesso em: 12 out. 2025.

[3] J. Bains *et al.*, "Therapy Without Borders: A Systematic Review on Telehealth's Role in Expanding Mental Health Access." medRxiv (Pré-publicação), Jul 2024. doi: 10.1101/2024.07.30.24311208. Acessado em: 30 de outubro de 2025.

[4] Vittude, "Vittude: Terapia online com psicólogos de todo o Brasil." https://www.vittude.com, 2025. Acessado em: 30 de outubro de 2025.

[5] Zenklub, "Zenklub: Sua saúde mental onde você estiver." https://zenklub.com.br, 2025. Acessado em: 30 de outubro de 2025.

[6] S. Ramírez, "Fastapi: Modern, fast (high-performance) web framework for building apis with python." https://fastapi.tiangolo.com, 2024. Acessado em: 30 out. 2025.

[7] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Boston, MA: Prentice Hall, 2017.

[8] "Diagrama de camadas da arquitetura limpa." https://lh7-rt.googleusercontent.com/docsz/AD_4nXeBPclD_6dEBAPSojtty79IMrqD45OhNWLt3crijBqs8Z8ofqfD3B46Si1hPBQIFW-0O99FRj-bMyKwOHCKh_2_gOmtTKbT9tv0kFD0JEA7kqtTVkOc85l_3urx2UDRN8A3Nt903w?key=pDWDZmaaqorvZcAzGyVOCrFs. Acessado em: 12 out. 2025.

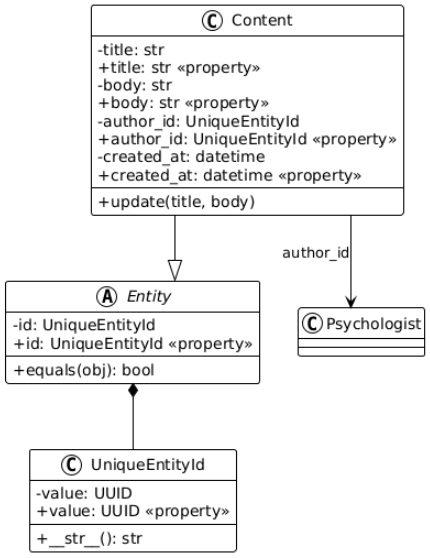

Figura 17. Camada de Domínio — Núcleo Conteúdo

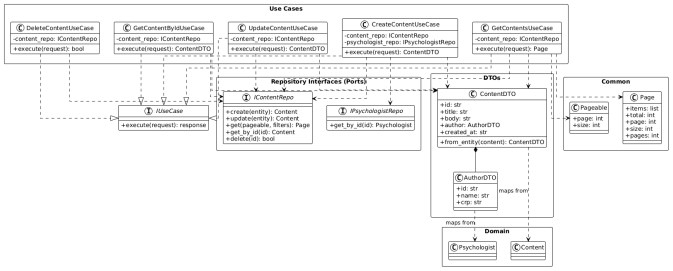

Figura 22. Camada de Aplicação — Núcleo Conteúdo

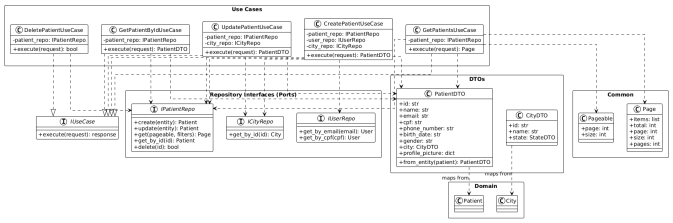

Figura 18. Camada de Aplicação — Núcleo Paciente

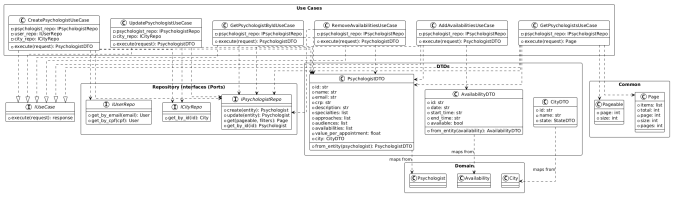

Figura 19. Camada de Aplicação — Núcleo Psicólogo

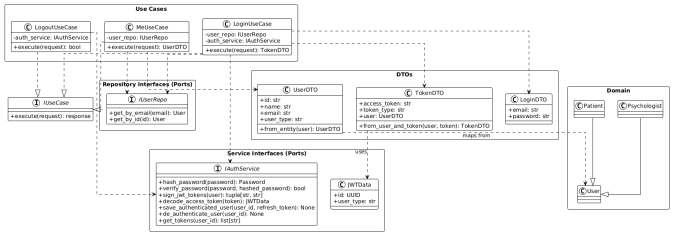

Figura 20. Camada de Aplicação — Núcleo Autenticação

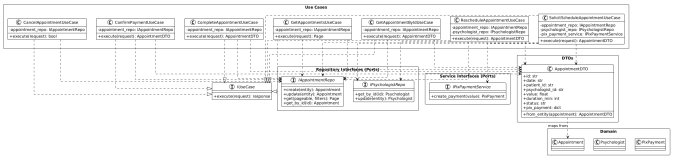

Figura 21. Camada de Aplicação — Núcleo Agendamento

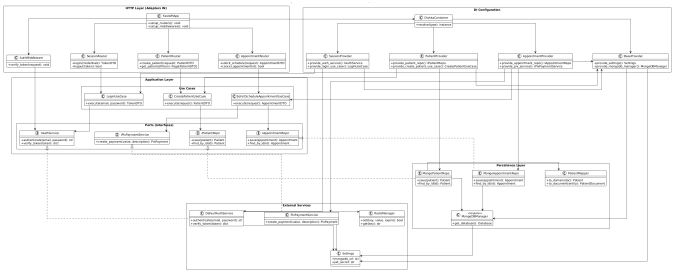

Figura 23. Camada de Infraestrutura — Componentes e Relações Principais