RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Uni-FACEF Centro Universitário Municipal de Franca
Vol. 13 Nro. 1 2023

SISTEMA DE TRIAGEM E ATENDIMENTO PARA CLÍNICA-ESCOLA DE PSICOLOGIA UNI-FACEF: PROJETO E IMPLEMENTAÇÃO

Pedro Gonzales Correa
Graduado em Engenharia de Software - Uni-FACEF
devPedroGonzales@gmail.com

Fábio Henrique de Souza Mariano
Graduado em Sistemas de Informação - Uni-FACEF
fabio.henrique1056@hotmail.com

Prof. Me. Carlos Alberto Lucas
Docente do Uni-FACEF
projetos@profcarloslucas.com.br

Resumo: Impulsionada pelo avanço tecnológico a gestão de informação de pacientes, no contexto da saúde, vem passando por evolução notável. Tradicionalmente, o registro de informações dos pacientes era realizado em papel, um processo moroso e suscetível a erros além de não garantir segurança sobre os dados sigilosos dos pacientes. Assim sendo, a transição para sistemas de triagem e atendimento surgiu como solução necessária, visando aprimorar a eficiência, segurança e qualidade do atendimento psicológico. Ao combinar os princípios da engenharia de software e as melhores práticas de desenvolvimento de software, este trabalho tem como propósito explorar a concepção, criação do projeto e implementação de um sistema de triagem e atendimento especificamente adaptado às necessidades da clínica de ensino de psicologia Uni-FACEF que visa acompanhar todo o processo de atendimento dos pacientes.

Palavras-chave: clínica-escola; desenvolvimento de software; engenharia de software.

Abstract: Driven by technological advancements, patient information management in the context of healthcare has been undergoing remarkable evolution. Traditionally, patient information was recorded on paper, a time-consuming and error-prone process that did not ensure the security of patients' confidential data. Consequently, the transition to electronic health record systems emerged as a necessary solution aimed at enhancing the efficiency, security, and quality of psychological care. By combining software engineering principles and best software development practices, this work aims to explore the conception, project design, and implementation of an electronic health record system specifically tailored to the needs of the Uni-FACEF psychology teaching clinic, which seeks to monitor the entire patient care process.

Keywords: teaching clinic; software development; software engineering.

1 Introdução

Nos últimos anos, a tecnologia tem desempenhado um papel fundamental na melhoria dos processos de gestão e atendimento na área da saúde. Em particular, os sistemas de triagem e atendimento têm ganhado destaque como uma solução eficiente para a organização, armazenamento e recuperação das informações clínicas dos pacientes. Esses sistemas permitem o registro eletrônico de dados de atendimento, facilitando o acesso, e compartilhamento de informações entre os profissionais de saúde, além de garantir segurança dos dados sensíveis, resultando em uma melhor qualidade do processo de atendimento e uma assistência mais integrada.

No contexto da Clínica-Escola de Psicologia, a implementação de um sistema de triagem e atendimento surge como uma oportunidade de aprimorar a gestão dos registros clínicos dos pacientes e potencializar o aprendizado dos estudantes de psicologia. A utilização de tecnologias na área da saúde proporciona benefícios significativos, como a melhoria da comunicação entre os profissionais e o aumento da segurança e confidencialidade das informações.

Além disso, a adoção de um sistema de triagem e atendimento na Clínica-Escola de Psicologia contribui para a padronização dos registros clínicos, facilitando a análise de dados e a realização de pesquisas científicas. Conforme apontado por Harper (2002) em seu livro, a redução da quantidade de processos em um projeto tecnológico resulta em um melhor aproveitamento dos recursos disponíveis.

O projeto aqui apresentado tem como objetivo mostrar o processo de desenvolvimento do sistema de triagem e atendimento para a Clínica-Escola de Psicologia da Uni-FACEF, tendo foco na implementação das melhores práticas de desenvolvimento, respeitando os processos da engenharia de software e boas práticas em UI/UX.

2 Referencial Teórico

2.1 Psicologia

Para Mischel (2014), a psicologia é a ciência que estuda o comportamento humano e animal e os processos mentais que o acompanham. Seguindo essa mesma base, os

autores Dennis Coon e John Mitterer (2010, p. 2) definem que "a psicologia é o estudo científico do comportamento humano e dos processos mentais que o tornam possível".

Já Bandura (1997, p. 3), um dos principais teóricos da Psicologia, afirma que "a Psicologia não é apenas o estudo do comportamento, mas também o estudo dos processos mentais que estão por trás do comportamento". Para entender esses processos mentais, a Psicologia utiliza várias abordagens teóricas e metodológicas, que oferecem diferentes visões sobre o comportamento humano.

2.2 Terapia

A terapia é uma intervenção que tem como objetivo ajudar as pessoas a superarem problemas emocionais, comportamentais ou relacionais, e a melhorar sua qualidade de vida. Segundo Corey (2017, p. 3), a terapia é "um processo colaborativo, em que o terapeuta e o cliente trabalham juntos para identificar e mudar padrões de pensamento, emoções e comportamentos que impedem o crescimento e o bem-estar emocional". A terapia pode ser oferecida em diversos contextos, incluindo clínicas, hospitais, consultórios particulares, e pode ser realizada individualmente, em grupo, ou em família.

Existem várias abordagens teóricas em terapia, cada uma com suas próprias técnicas e princípios. A clínica Uni-FACEF oferece acompanhamento em algumas dessas abordagens, sendo uma delas a terapia cognitivo-comportamental, uma das abordagens mais conhecidas e utilizadas em todo o mundo, que segundo Beck (2011), concentra em identificar e mudar padrões de pensamento negativos e comportamentos disfuncionais que contribuem para os problemas emocionais do paciente. Outra abordagem presente dentre os tipos de acompanhamento é a psicanálise, que se concentra na compreensão das dinâmicas inconscientes da mente humana. Essa teoria postula que os pensamentos, sentimentos e comportamentos são moldados por forças inconscientes, como desejos reprimidos e traumas do passado. Como observado por Freud (1915), a psicanálise busca explorar o "inconsciente" por meio da interpretação dos sonhos, associações livres e análise do paciente, visando trazer à tona questões subjacentes e promover a compreensão e resolução de conflitos psicológicos.

Os autores Norcross & Lambert (2018) dizem que independentemente da abordagem utilizada, o terapeuta deve estabelecer uma relação de confiança e empatia com o cliente para que este se sinta confortável em compartilhar seus pensamentos e emoções.

2.3 Clínica-Escola

Clínica-Escola é uma instituição fundamental na formação prática de profissionais de saúde mental, servindo como uma ponte essencial entre teoria e prática. Conforme afirmado por Jacó-Vilela (2015), esse ambiente proporciona aos estudantes a oportunidade de desenvolver habilidades clínicas enquanto contribuem para o bem-estar dos pacientes. Essa experiência prática supervisionada é crucial para a preparação de futuros terapeutas, permitindo-lhes aplicar seus conhecimentos em um contexto real de atendimento, sob a orientação de profissionais experientes.

Além do desenvolvimento de competências clínicas, a Clínica-Escola também desempenha um papel importante na promoção da ética profissional na prática clínica. Esse ambiente propício ajuda os estudantes a compreenderem e internalizarem os princípios éticos da profissão. A ética não é apenas ensinada, mas vivenciada, preparando os futuros terapeutas para enfrentar os desafios éticos que podem surgir ao longo de suas carreiras.

A Clínica-Escola serve como um espaço onde a pesquisa e a prática clínica podem se entrelaçar, contribuindo para avanços significativos na compreensão e tratamento de problemas de saúde mental, como mencionado por Pérez-Alvarez (2017). A colaboração entre estudantes, supervisores e pesquisadores cria um ambiente ideal para investigações que podem resultar em melhores abordagens terapêuticas e uma compreensão mais profunda dos desafios clínicos.

A Clínica-Escola também desempenha um papel crucial na democratização do acesso a serviços de saúde mental. Como ressaltado por Santos e Silva (2019), ela muitas vezes oferece atendimento de qualidade a pacientes que não têm acesso a outros recursos de assistência médica. Isso não apenas beneficia a comunidade, mas também proporciona uma experiência enriquecedora para os estudantes, que aprendem a lidar com uma ampla gama de casos e desafios.

3 Softwares existentes

Através de pesquisa foi possível identificar softwares existentes no mercado com o objetivo de auxiliar clínicas de psicologia, no entanto estas soluções são genéricas, não

personalizáveis, e não atendem a todas as necessidades de uma clínica escola. Abaixo será apresentada a solução mais próxima com o objetivo do projeto.

- Psicomanager - É uma aplicação para a gestão administrativa, criada especificamente para ajudar psicólogos na gestão de seus negócios. Possui diversas funcionalidades como controle financeiro, aplicativo para pacientes, formulário para envio de documentos, com destaque para a agenda online e o controle de sessões. Na figura 1 está apresentada a tela de agendamento de atendimentos, listando os pacientes, horários, salas disponíveis e ocupadas.

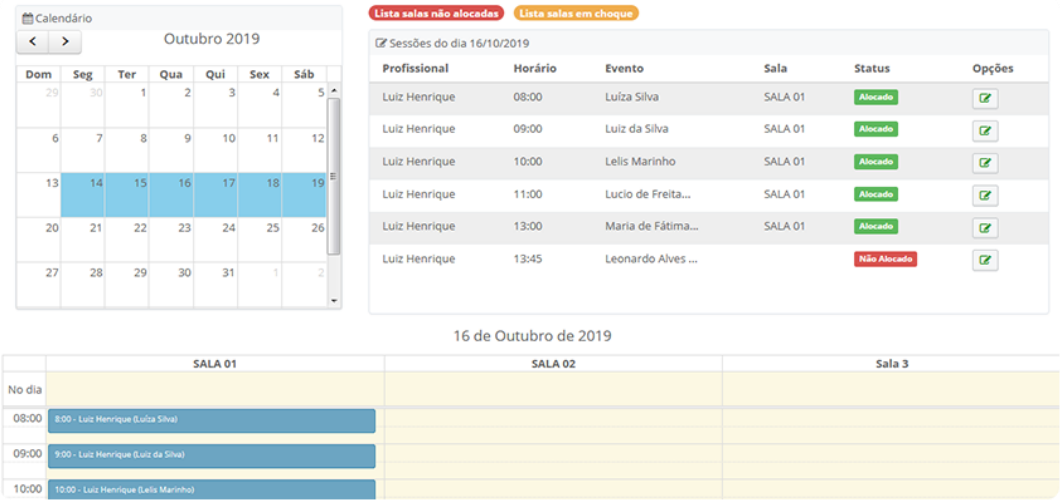

Figura 1 - Tela de agendamento de sessões

Fonte: (Psicomanager, 2023)

4 Engenharia de Software

Segundo Pressman (2014), a Engenharia de Software é uma área de atuação que se concentra na aplicação sistemática de princípios de engenharia no desenvolvimento de software. Abrangendo análise, projeto, implementação, testes e manutenção de sistemas de software para garantir que sejam entregues produtos de alta qualidade, dentro do prazo e do orçamento estabelecidos.

Durante o processo da engenharia, são gerados diversos artefatos, que são produtos documentais ou representações visuais que desempenham um papel essencial em todas as etapas do ciclo de vida do desenvolvimento de software. Esses artefatos servem como meios para capturar, expressar e comunicar informações essenciais relacionadas ao projeto.

4.1 Estrutura Analítica do Projeto

A Estrutura Analítica de Projeto (EAP) é uma valiosa ferramenta de gerenciamento de projetos que se constitui em um diagrama com finalidade de representar a decomposição do trabalho de um projeto em tarefas mais definidas e gerenciáveis, denominadas pacotes de trabalho. Esta abordagem cria uma hierarquia de tarefas que facilita a visualização e organização do escopo do projeto. No contexto da Engenharia de Software, a EAP pode ser empregada para a fragmentação do desenvolvimento do software em etapas e atividades específicas, simplificando, assim, o planejamento e controle do projeto.

4.2 Elicitação de requisitos

A elicitação de requisitos corresponde ao procedimento de coleta e documentação dos requisitos pertinentes a um sistema de software. Este processo envolve a interação com o *stakeholder*, ou seja, os usuários finais do software, com o intuito de compreender as suas necessidades e expectativas referentes ao software em desenvolvimento. É crucial ressaltar que a elicitação de requisitos desempenha um papel determinante no êxito de um projeto de engenharia de software, uma vez que proporciona a base necessária para a construção do software em conformidade com as especificações dos clientes e utilizadores.

4.3 Diagrama e documento de casos de uso

O diagrama e o documento de caso de uso constituem-se em instrumentos úteis na Engenharia de Software para a modelagem e comunicação do funcionamento do sistema sob a perspectiva do utilizador. O diagrama de caso de uso é uma representação gráfica que esboça as interações entre um sistema de software e usuários, viabilizando a identificação dos principais casos de uso e das suas relações no sistema. Já o documento de caso de uso compreende uma descrição textual mais minuciosa de cada caso de uso, englobando os seus passos, pré-condições e pós-condições.

4.4 Business Process Model and Notation

BPMN (Business Process Model and Notation) é uma notação gráfica padronizada utilizada para a modelagem de processos de negócios. Embora não esteja exclusivamente relacionada com a Engenharia de Software, assume relevância quando o software está sendo projetado com o propósito de apoiar processos de negócios. Este artefato possibilita a representação visual de fluxos de trabalho, atividades, eventos e decisões num processo de negócio, contribuindo para a compreensão e colaboração efetiva entre as equipes de desenvolvimento e os *stakeholders*.

5 Melhores práticas de desenvolvimento

Nessa sessão será abordado as melhores práticas de desenvolvimento de software que foram levadas em consideração para a realização deste projeto.

5.1 Metodologias Ágeis

As metodologias ágeis são frequentemente consideradas entre as melhores práticas no desenvolvimento de software. Elas são caracterizadas por um processo iterativo e incremental, que enfatiza a colaboração entre a equipe de desenvolvimento e o cliente. Segundo Beck et al. (2001), "O Manifesto Ágil é uma declaração de valores para o desenvolvimento de software ágil, com foco em indivíduos e interações, software funcionando, colaboração com o cliente e resposta às mudanças".

Um dos principais frameworks ágeis é o Scrum, criado por Schwaber e Sutherland (1995), que define papéis, artefatos e eventos que são usados para guiar o trabalho de uma equipe de desenvolvimento de software. Outro framework ágil muito utilizado é o Kanban, que foi criado no Japão pela Toyota nos anos 1950 e que foi adaptado para o desenvolvimento de software por David Anderson (2010).

Além desses frameworks, existem outras abordagens ágeis, como o *Extreme Programming* (XP), criado por Kent Beck (1999), que enfatiza a colaboração entre os membros da equipe e a entrega constante de software funcional; e o *Lean Software Development*, baseado nos princípios do *Lean Manufacturing* e que foi proposto por Mary Poppendieck e Tom Poppendieck (2003).

5.2 Testes Automatizados

Testes automatizados são considerados uma prática fundamental para o desenvolvimento de software de alta qualidade. Eles permitem que os desenvolvedores testem o código de forma rápida e precisa, reduzindo o tempo e o custo dos testes manuais, além de garantir segurança de manutenibilidade, ou seja, a capacidade do software de receber alterações de manutenção sem afetar negativamente as funcionalidades já existentes. De acordo com Bartié (2002, p. 196), testes automatizados são "a utilização de ferramentas de testes que possibilitem simular usuários ou atividades humanas de forma a não requerer procedimentos manuais no processo de execução dos testes".

Dentre as muitas formas de testes automatizados, os testes end-to-end (E2E) são uma técnica de teste de software que visa verificar se o sistema em questão funciona de forma esperada, do início ao fim do fluxo do usuário. Segundo Clerissi et al. (2017), os testes E2E são importantes porque são capazes de detectar problemas em todo o sistema, incluindo integrações com outros sistemas e dependências externas.

Os testes E2E podem ser realizados manualmente, mas com a crescente complexidade dos sistemas, tornou-se cada vez mais comum automatizá-los. A automação dos testes E2E pode ser uma tarefa complexa, mas oferece vantagens como a capacidade de executar testes repetitivos com precisão, a redução do tempo e custo dos testes e a capacidade de testar cenários complexos.

5.3 Controle de Versão

O controle de versão é outra prática imprescindível no desenvolvimento de software. Permite que os desenvolvedores gerenciem o código-fonte, acompanhem as alterações e colaborem com a equipe. De acordo com Spinellis (2012), "O controle de versão é essencial para garantir a integridade do código e para permitir que os desenvolvedores trabalhem juntos de forma eficaz".

5.4 Padrões de Projeto

Padrões de projeto são soluções comprovadas para problemas comuns de design de software. Eles ajudam os desenvolvedores a criar software de alta qualidade e fácil de manter.

Segundo Gamma et al. (1994), "Os padrões de projeto fornecem uma linguagem comum para os desenvolvedores discutirem problemas de design e soluções comprovadas para esses problemas".

6 Ferramentas e Métodos

Nesta seção serão apresentadas as ferramentas e as tecnologias utilizadas para o desenvolvimento do projeto.

6.1 Trello

O Trello é uma plataforma online para gerenciamento de projetos que tem como objetivo fornecer uma maneira fácil e intuitiva de gerenciar projetos em equipe. Na plataforma é possível criar quadros para representar projetos e listar para representar o fluxo de trabalho.

De acordo com Kruchten et al. (2019), a metodologia Kanban, que o Trello adota, foi desenvolvida pela Toyota nos anos 1950 como um sistema de produção just-in-time. A ideia básica é que o trabalho seja dividido em tarefas menores, que são então organizadas em um fluxo de trabalho visual, com o objetivo de aumentar a eficiência e a produtividade. O Trello usa essa abordagem para gerenciar projetos, permitindo que as equipes criem quadros, listas e cartões para organizar o trabalho e atribuir tarefas aos membros da equipe.

6.2 Typescript

TypeScript é uma linguagem de programação que foi desenvolvida pela Microsoft em 2012 como uma extensão da linguagem JavaScript. A principal característica do TypeScript é o uso de tipagem estática, o que significa que os tipos de dados são definidos explicitamente no código. Isso torna o desenvolvimento mais fácil e menos propenso a erros. O Typescript é recomendado em todos os tipos de projeto, desde os pequenos até os maiores.

6.3 Next.Js

Next.js é um framework web baseado em React que foi desenvolvido para ajudar os desenvolvedores a criar aplicativos web modernos de forma mais rápida e eficiente. O Next.js

oferece uma série de recursos avançados, como renderização do lado do servidor, geração de páginas estáticas e suporte para rotas dinâmicas, que ajudam a melhorar a experiência do usuário e o desempenho do aplicativo.

Segundo Han et al. (2020), o Next.js é uma escolha popular para desenvolvedores que desejam criar aplicativos web complexos e dinâmicos. A renderização do lado do servidor permite que o aplicativo seja carregado mais rapidamente e melhora a experiência do usuário, enquanto a geração de páginas estáticas ajuda a reduzir o tempo de carregamento e melhorar o desempenho do aplicativo. Além disso, o Next.js também oferece suporte para rotas dinâmicas, o que permite que os desenvolvedores criem páginas personalizadas e personalizadas para diferentes usuários.

6.4 VsCode

VsCode é um editor de código-fonte desenvolvido pela Microsoft. É uma ferramenta altamente popular entre desenvolvedores de software e é conhecido por sua extensibilidade, facilidade de uso e suporte para uma ampla variedade de linguagens de programação.

Segundo Maldonado e Zamora (2020), o VSCode é altamente personalizável e extensível. Ele oferece suporte para uma ampla variedade de extensões, que permitem que os desenvolvedores personalizem a interface do usuário, adicionem recursos avançados e integrem o editor com outras ferramentas de desenvolvimento. Isso torna o VSCode uma ferramenta altamente flexível e adaptável que pode ser usada para uma ampla variedade de projetos de software.

6.5 MySql

MySQL é um dos sistemas gerenciadores de banco de dados mais populares no mundo do desenvolvimento de software. É um sistema de banco de dados relacional de código aberto, que fornece uma ampla variedade de recursos para gerenciamento de dados e é usado em uma ampla variedade de aplicativos, desde pequenas aplicações até sistemas empresariais complexos.

Segundo Kofler e Kramer (2019), o MySQL é um dos sistemas de banco de dados mais usados no mundo. Ele oferece um ambiente seguro e escalável para o

armazenamento e gerenciamento de dados, e é compatível com uma ampla variedade de plataformas, incluindo Windows, Linux e MacOS. O MySQL é altamente personalizável e oferece uma ampla variedade de recursos para suportar a modelagem de dados, o gerenciamento de transações e a segurança de dados.

6.6 Prisma

Prisma é um ORM (Object-Relational Mapping) para bancos de dados que permite que desenvolvedores escrevam código para bancos de dados de forma fácil e intuitiva. Com o Prisma, os desenvolvedores podem escrever código em sua linguagem de programação favorita e usar modelos de dados para mapear automaticamente os dados do banco de dados para objetos no código.

Segundo a documentação oficial do Prisma (2021), a ferramenta oferece suporte a vários bancos de dados populares, como MySQL, PostgreSQL e MongoDB. Além disso, ele oferece uma camada de abstração de banco de dados que permite que os desenvolvedores trabalhem com bancos de dados sem ter que lidar diretamente com SQL. Isso torna o desenvolvimento de aplicativos mais rápido e fácil, permitindo que os desenvolvedores se concentrem no código do aplicativo em vez de se preocuparem com a complexidade do SQL.

6.7 Docker

Docker é uma plataforma de virtualização de contêineres que permite que os desenvolvedores empacotem e executem aplicativos em ambientes isolados. De acordo com a documentação oficial do Docker (2021), os contêineres do Docker fornecem uma maneira padronizada e portátil de empacotar e executar aplicativos em diferentes ambientes.

Os contêineres do Docker são baseados em imagens, que são arquivos de configuração que descrevem o ambiente de tempo de execução necessário para executar o aplicativo. Conforme descrito por Hamzei (2020), as imagens do Docker podem ser facilmente compartilhadas e implantadas em diferentes ambientes de desenvolvimento e produção.

6.8 Git e Gitlab

O Git é um sistema de controle de versão distribuído, criado por Linus Torvalds em 2005. Ele desempenha um papel fundamental no desenvolvimento de software, permitindo o rastreamento e a colaboração eficiente em projetos de código aberto e privados. Conforme observado por Torvalds (2005), "O Git é rápido, é eficiente e possui um modelo de dados extremamente sólido que oferece suporte a fluxos de trabalho de desenvolvimento não lineares".

GitLab, por outro lado, é uma plataforma abrangente de gerenciamento de ciclo de vida de DevOps que inclui recursos de hospedagem de código, integração contínua, entrega contínua e muito mais. Como destacado por James Smith (2018), "O GitLab oferece uma abordagem integrada e coesa para todo o ciclo de vida do desenvolvimento de software, proporcionando maior eficiência e visibilidade." Com sua interface amigável e recursos abrangentes, o GitLab se tornou uma escolha popular para equipes de desenvolvimento em todo o mundo.

A integração entre Git e GitLab é uma combinação poderosa. O Git é responsável pelo controle de versão do código-fonte, enquanto o GitLab oferece uma plataforma para hospedar, colaborar e automatizar o processo de desenvolvimento. Essa combinação é essencial para equipes que buscam gerenciar eficientemente seus projetos de software. Além disso, o GitLab oferece recursos avançados, como pipelines de CI/CD (Integração Contínua e Entrega Contínua), que permitem a automação de testes e implantações, aumentando a qualidade e a rapidez do desenvolvimento (Smith, 2018).

7 Desenvolvimento

A elaboração de um sistema envolve um processo sistemático de planejamento, implementação e manutenção de um conjunto de componentes inter-relacionados que trabalham juntos para atingir um objetivo específico. Esse processo requer uma abordagem cuidadosa e estruturada, levando em consideração vários fatores, como requisitos do sistema, recursos disponíveis e restrições técnicas.

7.1 Planejamento

É crucial que em todo projeto de software haja um planejamento minucioso do processo de desenvolvimento e acima de tudo o entendimento do propósito, objetivos e limites esperados para a solução. Com isso foi realizado um briefing com a coordenadora da clínica de psicologia para entender as necessidades e expectativas para o sistema.

Após os autores realizarem a viabilidade do sistema, foi criado um quadro na plataforma Trello, representado pela Figura 2, para organização das tarefas e acompanhamento da produtividade. No quadro é possível acompanhar todas as tarefas pendentes (Backlog); as tarefas pendentes para a sprint atual (To do); as tarefas em andamento (Doing); as que foram finalizadas mas dependem de aprovação terceira (Resolved); duas colunas para tarefas de desenvolvimento de software, as tarefas que estão pendentes de análise do código (Code Review) e as tarefas que estão em teste (Testing) e as tarefas finalizadas (Done).

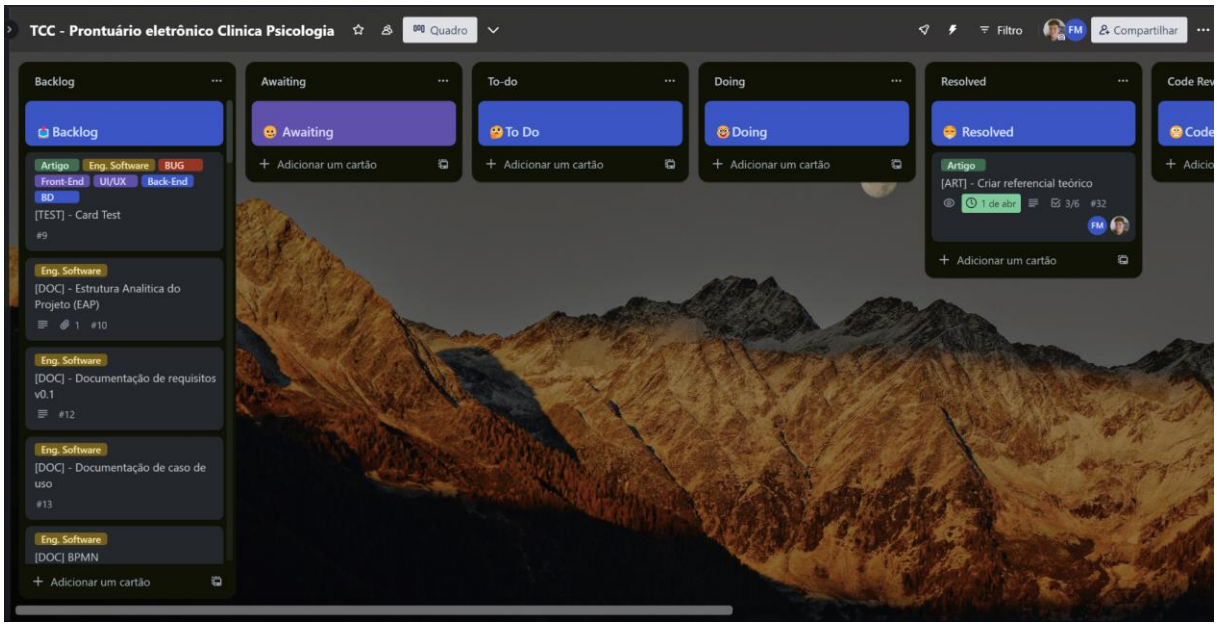

Figura 2 - Trello usado para o desenvolvimento do projeto

Fonte: Elaborado pelos autores

7.2 EAP

Foi realizado um diagrama de Estrutura Analítica do Projeto (EAP), ilustrado na Figura 3, para que todos os integrantes do projeto consigam compreender o escopo do projeto de maneira clara, estruturada e visual. No nível mais alto está definido o nome do projeto, os

níveis intermediários incluem as principais entregas do projeto, já os níveis das extremidades definem as etapas iniciais e finais do projeto.

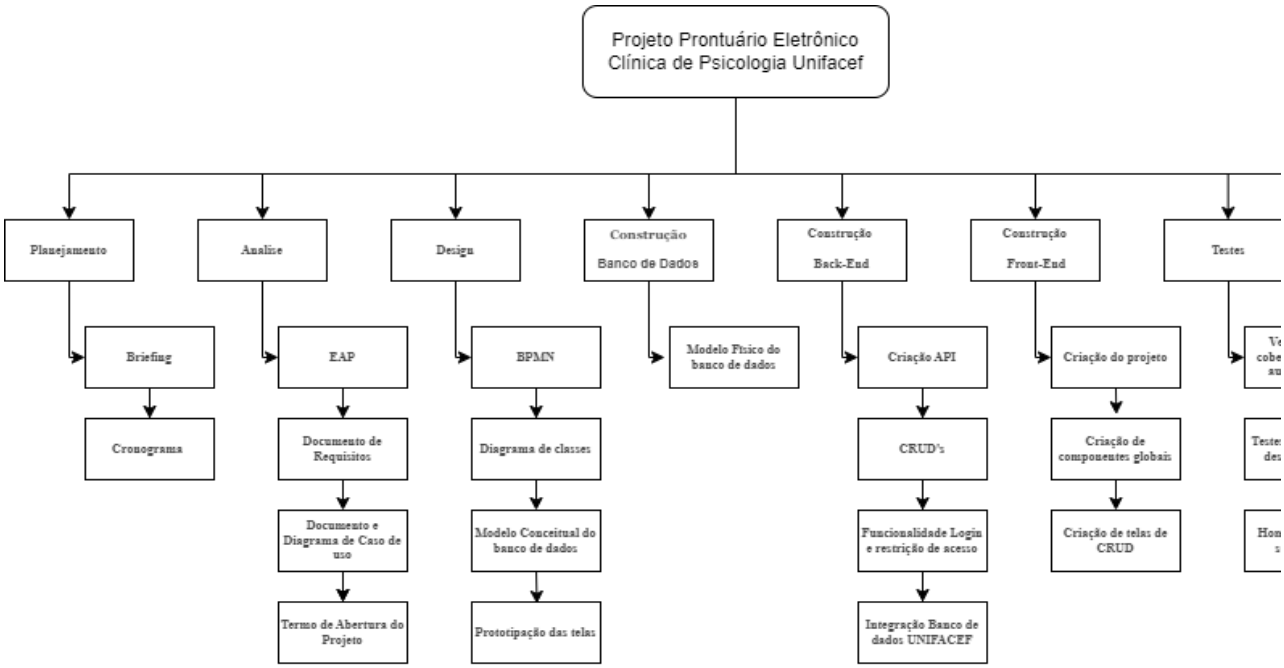

Figura 3 - Estrutura Analítica do Projeto

Fonte: Autoria própria

7.3 Elicitação de Requisitos

Com os processos de execução do projeto definido, foram realizadas reuniões com a coordenadora da clínica de psicologia para o levantamento dos requisitos do sistema. Onde, usando da técnica de entrevista, foi documentado as funcionalidades, restrições e expectativas que devem ser atendidas para satisfazer as necessidades do usuário final.

Em seguida, foi formalizado um Documento de Requisitos que divide as necessidades em Requisitos Funcionais, que se refere a aquelas funcionalidades e comportamentos que o sistema deve garantir, e Requisitos Não Funcionais, que são restrições e atributos de qualidade que o sistema deve contemplar como usabilidade e segurança.

Na Figura 4 estão representados alguns dos requisitos funcionais identificados durante o processo de licitação de requisitos.

[RF001] — Cadastrar abordagem — O sistema deve conter uma tela de cadastro da abordagem inicial. O cadastro deve ser preenchido apenas com o dado descrição. O cadastro da abordagem é restrito, portanto, somente a coordenação deve executar essa ação.

[RF002] — Cadastrar professor — O sistema deve conter uma tela de cadastro e listagem de professores, O cadastro deve ser preenchido apenas com o nome do professor. O cadastro dos professores é restrito, portanto, somente a coordenação deve executar essa ação.

[RF003] — Cadastrar estágio — O sistema deve conter uma tela de cadastro e listagem dos supervisores. O cadastro deve ser preenchido com os dados: nome do estágio, professor supervisor. Caso o estágio selecionado seja "clínica", o usuário deve informar qual a abordagem.

[RF004] — Associar aluno ao estágio — O sistema deve conter uma tela de listagem dos programas de estágio e uma tela para associar um ou mais alunos a um estágio. A associação é restrita, portanto, somente a coordenação deve executar essa ação.

[RF005] — Cadastrar usuários — O sistema deve conter uma tela de listagem dos programas de estágio e uma tela para associar um ou mais alunos a um estágio. A associação é restrita, portanto, somente a coordenação deve executar essa ação.

[RF006] — Pré-cadastro de pacientes — O sistema deve conter uma tela de cadastro de usuário para aqueles que não possuem usuário vinculado à UNI-FACEF. O cadastro será preenchido com os seguintes dados: nome, usuário, senha e nível de acesso. A criação é restrita, portanto, somente a coordenação deve executar essa ação.

[RF007] — Realizar triagem — O sistema deve conter uma tela de listagem com os possíveis pacientes com maior prioridade, juntamente com seu número de contato e um registro com as tentativas de contato. Após a segunda tentativa de contato, o sistema deverá permitir que o estagiário possa realizar o desligamento do paciente com uma justificativa. Caso o possível paciente seja menor de idade, é necessário que no cadastro haja anexo a autorização do responsável, caso contrário não será possível realizar o atendimento. Somente os estagiários podem realizar esse processo.

[RF008] — Ficha de triagem — O sistema deve conter uma tela para abertura da ficha de triagem. A abertura da ficha de triagem deverá conter os campos presente no arquivo "Modelo ficha de triagem". Caso o paciente seja menor de idade, é necessário que no cadastro haja anexo a autorização do responsável, caso contrário não será possível realizar o atendimento.

[RF009] — Lista de espera pós triagem — O sistema deve conter uma tela para que a coordenação possa visualizar os documentos de triagem e a opção que possibilita atribuir um paciente a um programa. Um paciente só pode estar associado a um tipo de programa. Somente a coordenação deverá ter acesso a essa tela

Figura 4 - Requisitos Funcionais do Sistema

Fonte: Autoria própria

7.4 BPMN

Com o intuito de identificar e esclarecer o fluxo dos processos de trabalho e restrições de acesso, que sistematicamente são representados por regras de negócio, dos

colaboradores da clínica foi realizado o BPMN, representado parcialmente pela Figura 5. Onde foi definido todos os possíveis fluxos de trabalho separados pelos papéis dos colaboradores da clínica, sendo eles administração, supervisores, estagiários e secretaria.

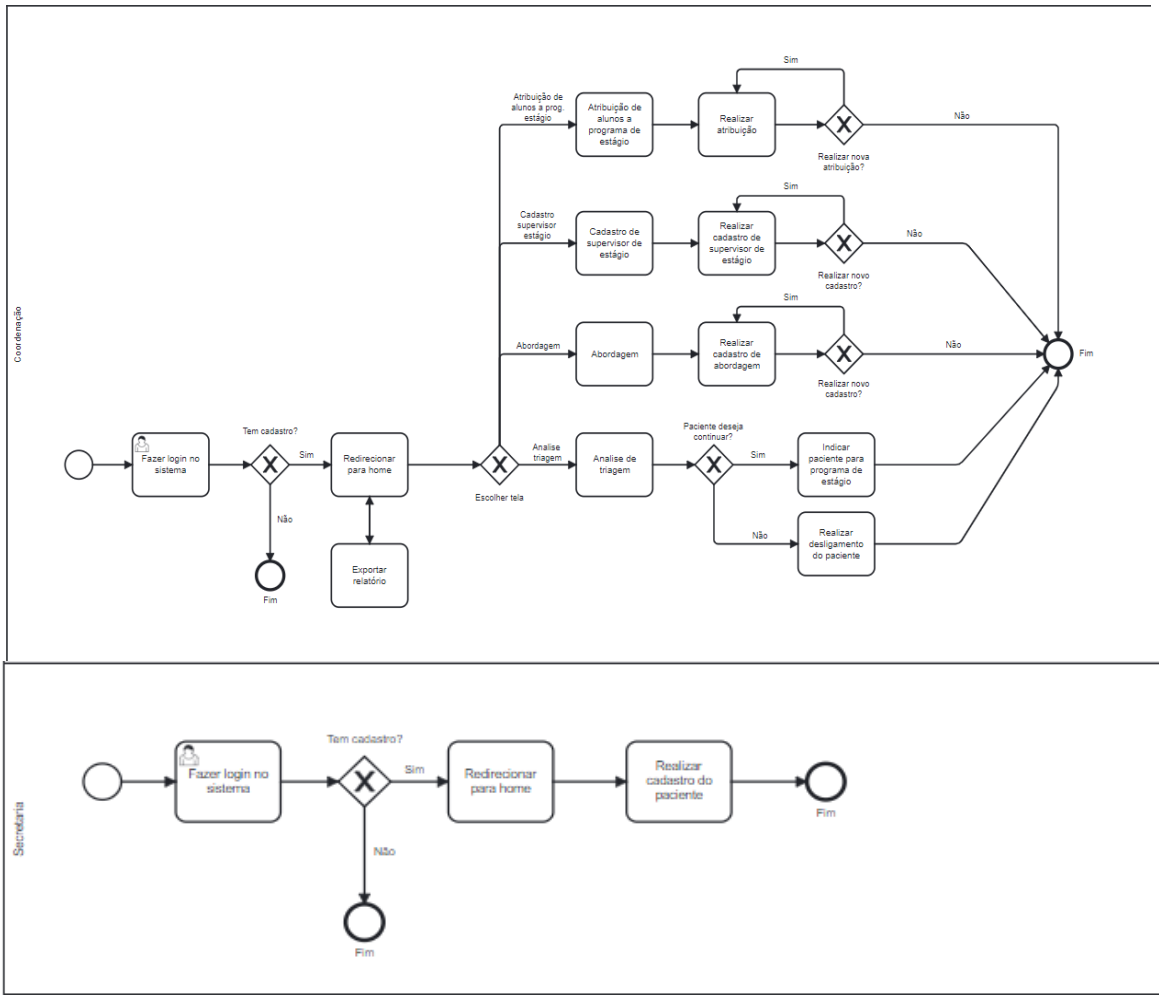

Figura 5- BPMN

Fonte: Autoria própria

7.5 Diagrama de Caso de Uso

Com os requisitos, regras de negócio e fluxos de trabalho definidos, foi realizada a representação visual dos fluxos que os usuários poderão realizar dentro do sistema de acordo com seu nível de acesso, como representa a Figura 6. Neste diagrama é possível identificar, pelas setas, como é possível o usuário acessar cada funcionalidade do sistema, esses representados pelas elipses.

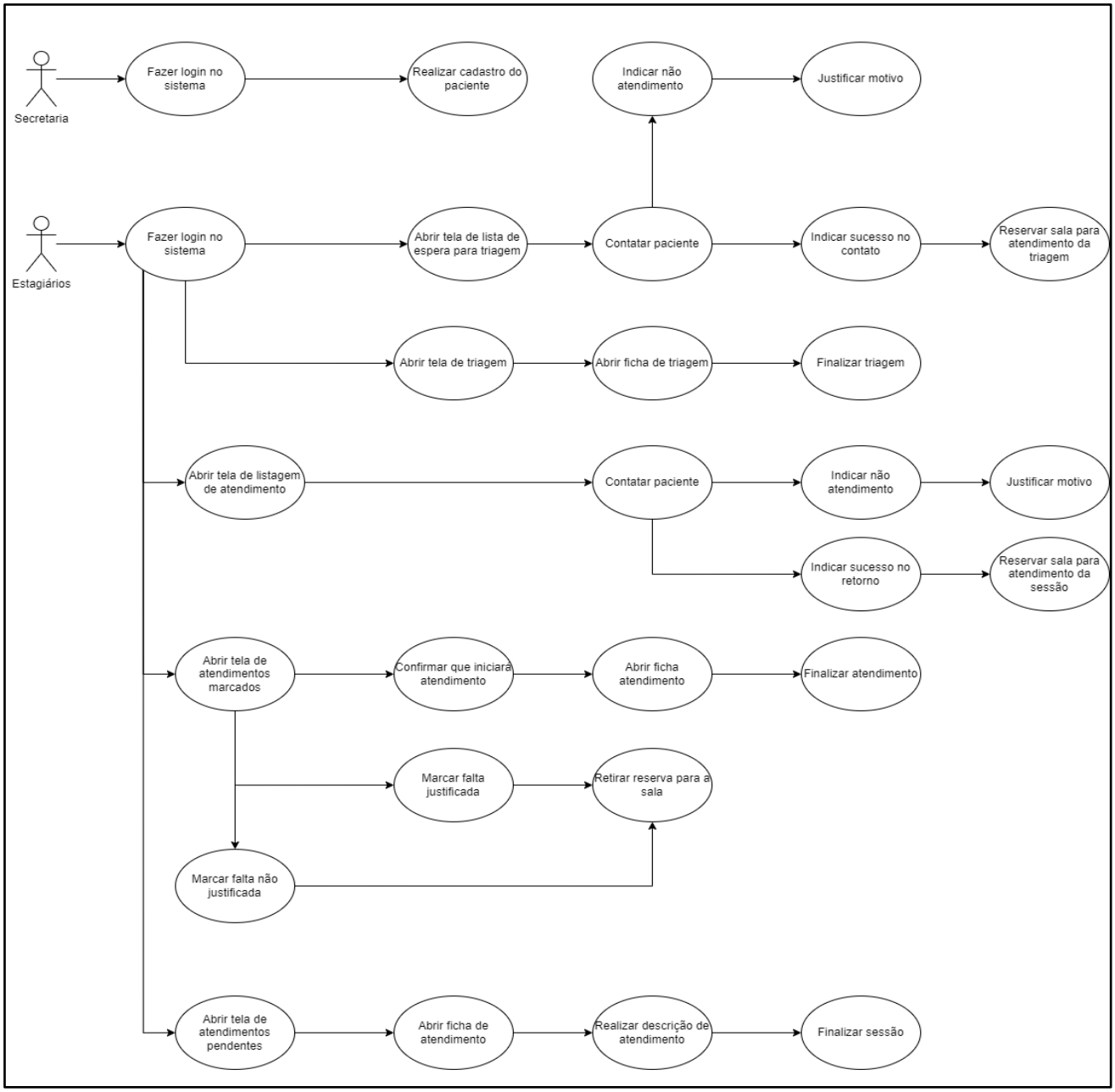

Figura 6 - Diagrama de Caso de Uso do Sistema

Fonte: Autoria própria

7.6 Documento de Caso de Uso

Após a validação do último artefato, foi realizada a formalização por texto do diagrama, o documento de requisitos, representado pela figura 7. Onde de forma detalhada as

funcionalidades são descritas juntamente com os atores que têm permissão de realizar determinada ação, cenários principais e alternativos.

Caso de uso – Iniciar atendimento de triagem	
ID	UC 009
Descrição	Este caso tem por objetivo realizar a triagem do paciente
Ator Primário	Estagiário
Pré-condição	Ter realizado o login no sistema como estagiário, ter reservado sala para atendimento de triagem
Cenário Principal	1. O use case se inicia após o estagiário confirmar que irá iniciar o atendimento de triagem. 2. Após o processo de triagem, o estagiário deverá finalizar a triagem.
Pós-condição	Não possui
Cenário Alternativo	*a – Em qualquer momento o aluno pode sair do sistema 1ª – O estagiário pode esquecer de finalizar a triagem
Inclusão	
Extensão	UC010 – Enviar Dúvida

Figura 7 - Documento de caso de uso

Fonte: Autoria própria

7.7 Diagrama de Classes

O primeiro passo para a construção do diagrama consistiu na identificação das classes relevantes para o sistema de acordo com a documentação existente, que foram as classes que representavam as funcionalidades de perfil de acesso, tipos de estágio, a realização do agendamento de sala para atendimento, entre outras. Para cada classe identificada, foram definidos e detalhados os atributos (nome e tipo de dado) e métodos correspondentes. Isso proporcionou uma compreensão abrangente das características e comportamentos de cada classe.

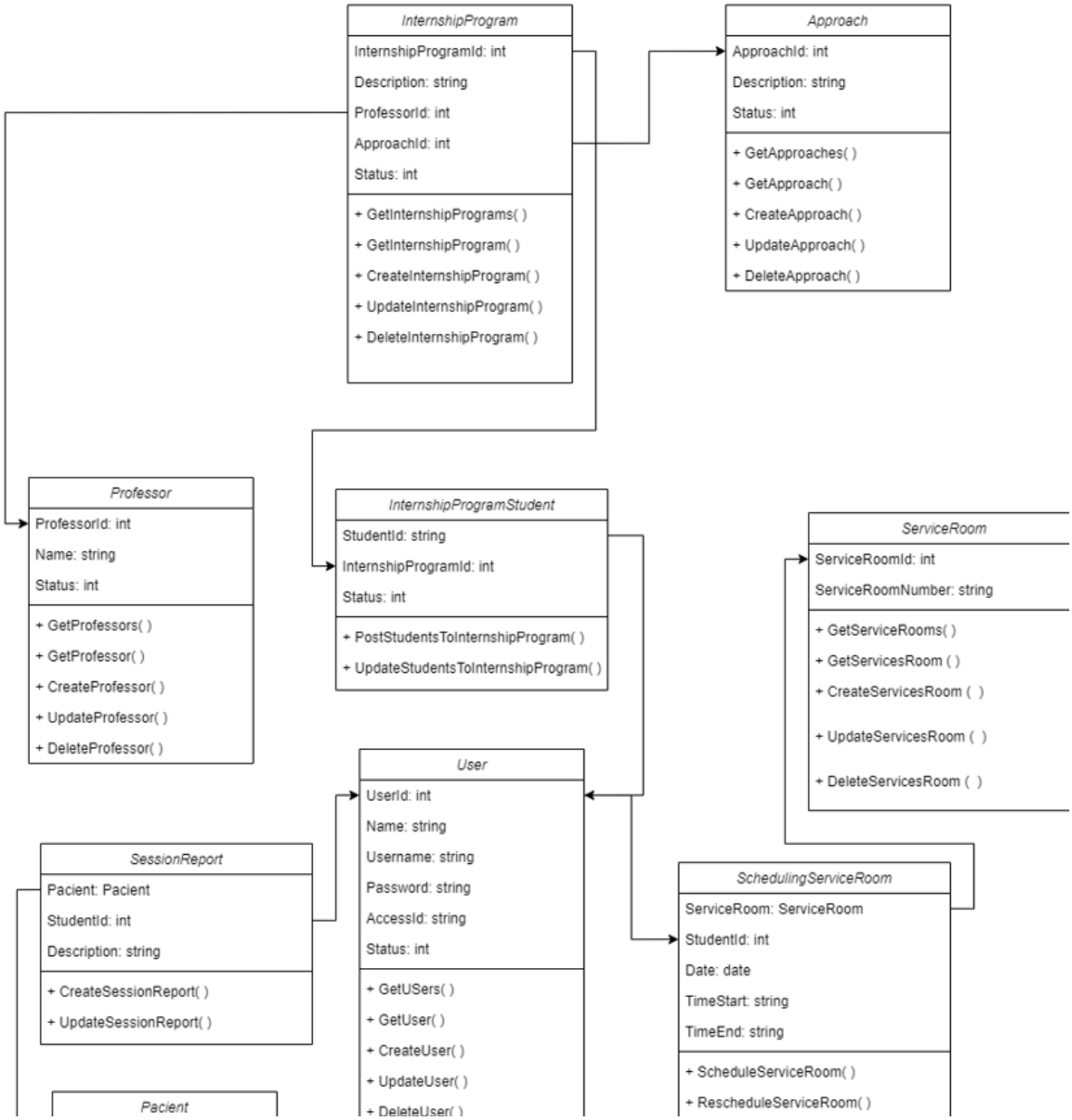

Figura 8 - Diagrama de classes

Fonte: Autoria própria

7.8 Modelo Conceitual do Banco de Dados

Com o diagrama de classes pronto, foi possível realizar o mapeamento das classes para as tabelas do banco de dados. As classes que foram transformadas em tabela tiveram seus atributos transformados em colunas e tiveram alteração da nomenclatura de seus nomes, de *CamelCase* para *Snake Case*, além de terem sido pluralizadas.

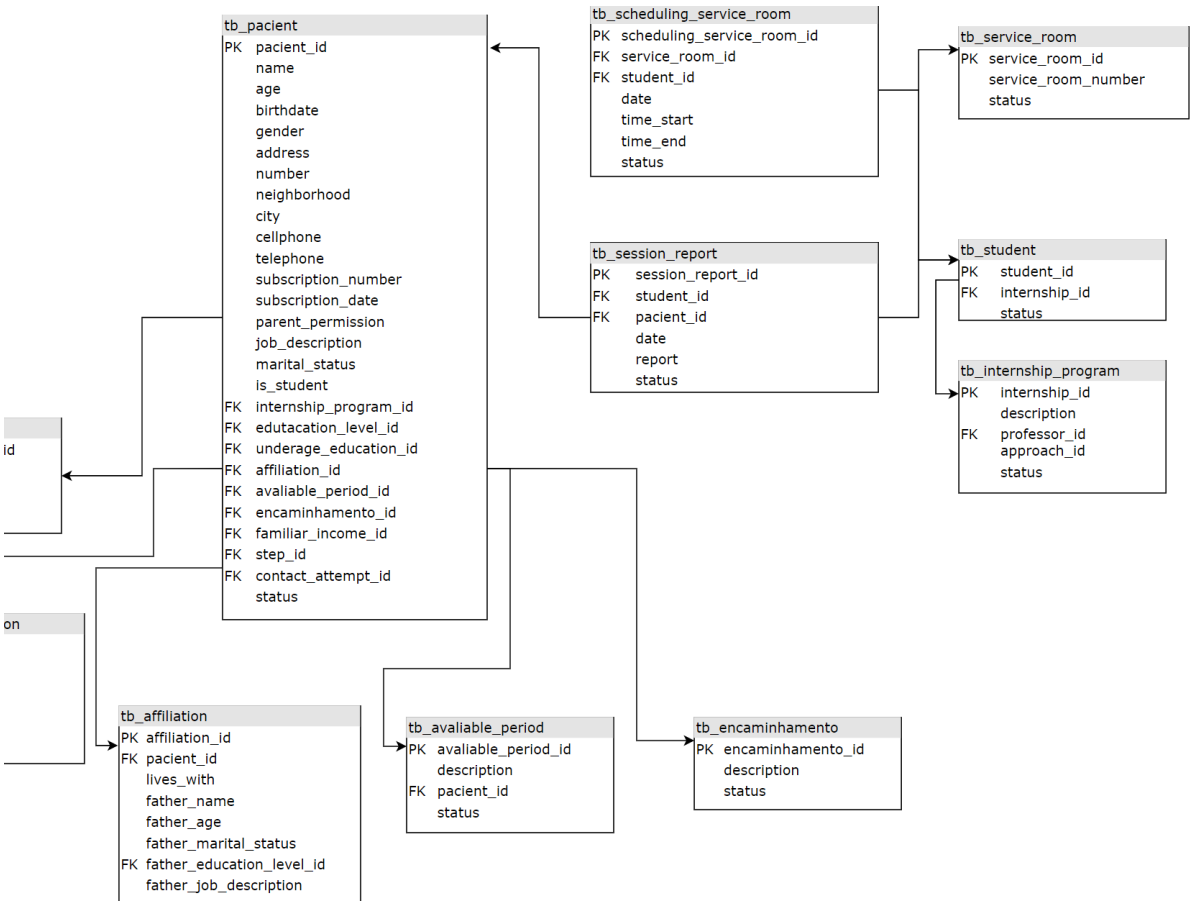

Figura 9 - Modelo conceitual do banco de dados

Fonte: Autoria própria

7.9 Prototipação das telas

Antes de iniciar a prototipação, foi fundamental estabelecer claramente os objetivos e metas do sistema em termos de usabilidade, design e funcionalidade. Foi definido que os objetivos do sistema precisam ser claros o suficiente para que não seja necessário um longo treinamento com os novos usuários, visto que a cada ano surge uma nova turma de estagiários, além de ter sido definido que o sistema deve se remeter às identidades visuais da faculdade e da clínica, que se mantivesse minimalista e contemplasse todas as funcionalidades definidas pelo documento de requisitos.

Para esse projeto foi criado somente um modelo de média fidelidade chamado *wireframe*, representado pela Figura 10. Que consiste num protótipo que mostra quais campos devem conter no sistema, telas e funcionalidades, mas que não contém a identidade visual,

tipografia e todas as proporções corretas que têm como propósito receber a validação do stakeholder de que essa solução contempla as expectativas, além de coletar feedbacks de usabilidade. Nesse primeiro protótipo foi relatado pela coordenadora da clínica que os ícones do menu de navegação não eram comuns, podendo causar confusão para novos usuários.

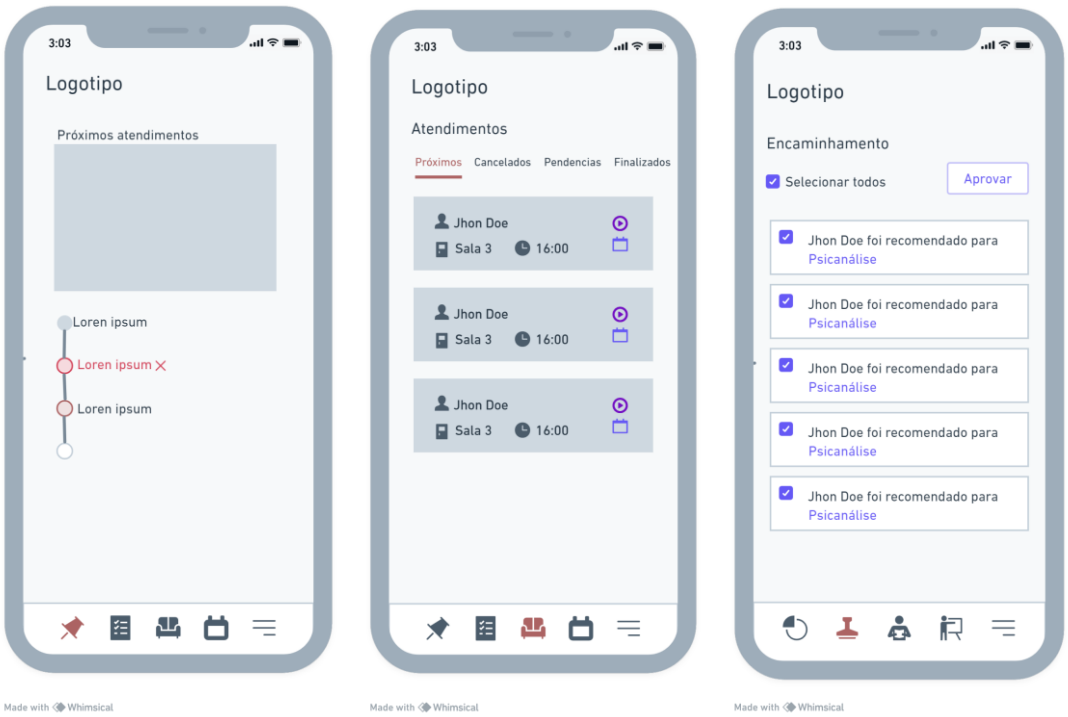

Figura 10 - Wireframe

Fonte: Autoria própria

7.10 Back-End

Para a criação da API foi necessária a escolha das tecnologias que seriam utilizadas levando em consideração os objetivos do back-end, uma API efetiva, que garanta a segurança dos dados, que não apresentasse bugs, que fosse construída de tal forma que fosse fácil receber manutenção e que caso necessário seja possível trocar a conexão com o banco de dados.

Com isso foi escolhido o Prisma para ser o responsável pela criação e conexão com o banco de dados. Com ele foi possível criar um *schema*, conforme mostra a Figura 11, que representa todas as tabelas do banco de dados além de definir qual Sistema de gerenciamento de banco de dados (SGBD), portanto caso seja necessário alterar o MySQL para

algum outro SGBD seria necessário somente alterar a parametrização do provedor no arquivo de configuração do Prisma, a URL do banco de dados e solicitar que o Prisma crie um banco de dados.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

model Approach {
  id String @id @default(uuid())
  description String
  status Int? @default(1)
}

model User {
  id String @id @default(uuid())
  name String
  username String
  password String
  access_level_id String
  status Int? @default(1)

  acessLevel AcessLevel @relation(fields: [access_level_id], references: [id])
}

model AcessLevel {
  id String @id @default(uuid())
  description String
  status Int? @default(1)

  users User[]
}
```

Figura 11 - Arquivo de configuração do Prisma

Fonte: Autoria própria

Com o banco de dados criado, o próximo passo foi definir a estrutura na qual a API seria construída (Figura 12). Foi definido que as requisições seriam recebidas mediante protocolo HTTP, passaria por uma validação para garantir que o usuário estava logado e com a permissão necessária para realizar a ação solicitada, uma validação do corpo da requisição para verificar se os dados obrigatórios tinham sido informados pelo *client* e só depois aconteceria o registro no banco de dados ou o retorno das informações solicitadas.

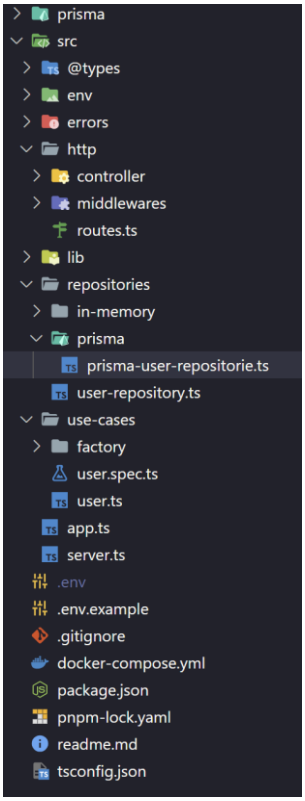

Figura 12 - Estrutura do Back-End

Fonte: Autoria própria

Para assegurar que as funcionalidades suprem os requisitos sem a presença de *bugs* e para garantir o aspecto de segurança para futuras implementações foram criados testes unitários, representado pela Figura 13, e de ponta a ponta usando a biblioteca Vitest. Onde de forma automatiza a biblioteca percorre pelos testes é válida se o retorno é o mesmo que o esperado.

```
describe('User use case tests', ()=>{
    beforeEach(() => {
        usersRepository = new InMemoryUserREpository()
        registerUseCase = new CreateUser(usersRepository)
      })
    it('should hash user password upon registration', async ()=>{
        const { user } = await registerUseCase.execute({
            email: 'jhondoe@example.com',
            username:'jhonGameplays',
            password: '1234567'
        })

        const isPasswordHashCorrectly = await compare('1234567', user.password)

        expect(isPasswordHashCorrectly).toBe(true)
    })
```

Figura 13 - Trecho de código de teste unitário.

Fonte: Autoria própria

7.11 Front-End

Também foi preciso definir as tecnologias que seriam usadas na construção do frontend e sua estrutura, conforme mostra a Figura 14.

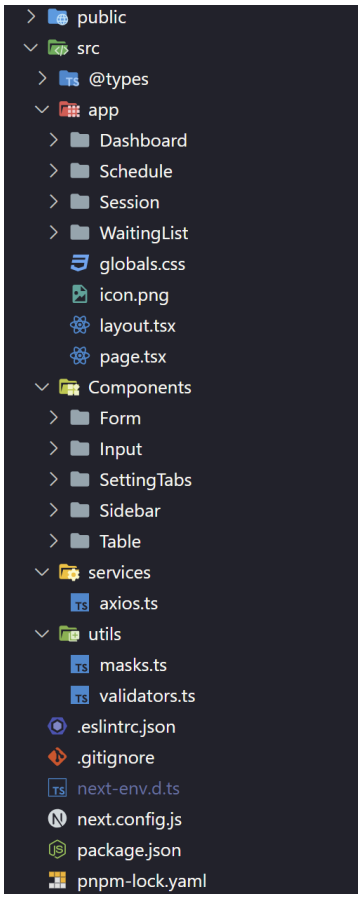

Figura 14 - Estrutura do Front-End

Fonte: Autoria própria

Levando em consideração que o produto final teria que um sistema WEB responsivo, ou seja, tem como prioridade a interface mobile para que de fato possa acompanhar os estagiários e supervisores durante todo o processo de atendimento, também seria necessário que o site se adaptasse em telas maiores, de computadores, para que fosse mais fácil realizar tarefas que demandam muita digitação como relatório pós atendimento foi usado a biblioteca Tailwind para estilizar o site já que esta traz consigo mecanismos que funcionam a implementação de responsividade, conforme mostra a Figura 15 onde é informado para que o componente de menu mobile deixe de existir quando a resolução *large*, “lg”, seja atingida.

```
<div className="flex lg:hidden ml-auto items-center">
    <Collapsible.Root className="z-20" onOpenChange={setIsNavBarOpen}>
        <Collapsible.Trigger asChild>
```

Figura 14 - Trecho de código que representa regra de responsividade

Fonte: Autoria própria

Com as tecnologias definidas, foi implementado os componentes globais da aplicação, como barra de navegação, barra lateral, botões, tabelas, entre outros, juntamente com as telas levando o wireframe criado na sessão de Prototipação das telas e adicionando identidade visual, tipografia correta e a funcionalidade de *tooltip* para suprir a problema de identificação de ícones não convencionais, como exemplifica a Figura 15 e Figura 16

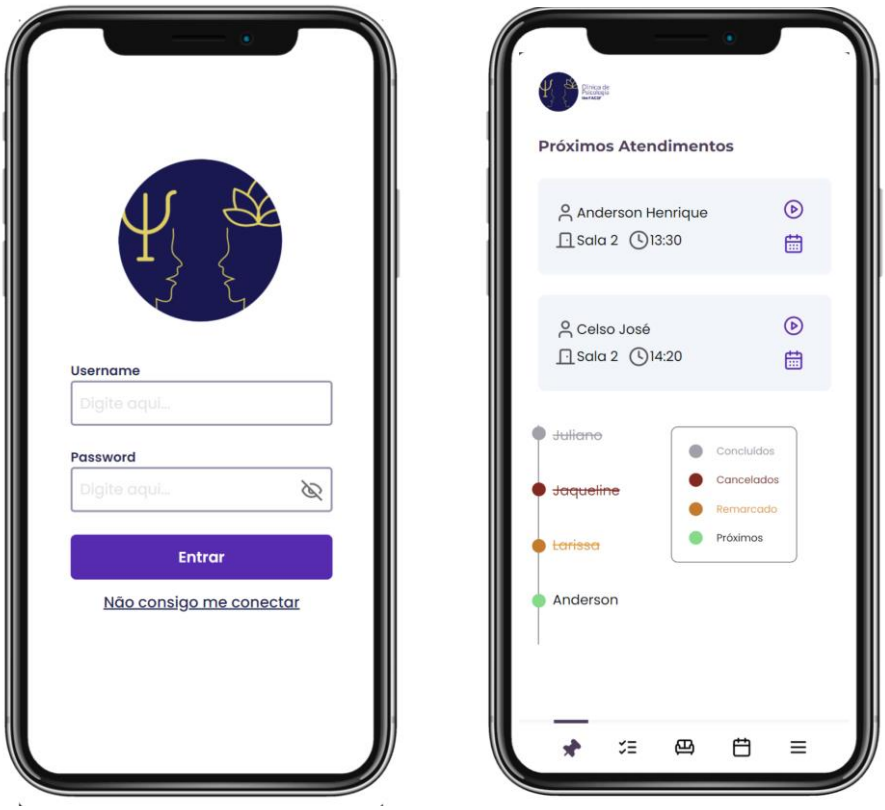

Figura 15 - Telas de Login e Home

Fonte: Autoria própria

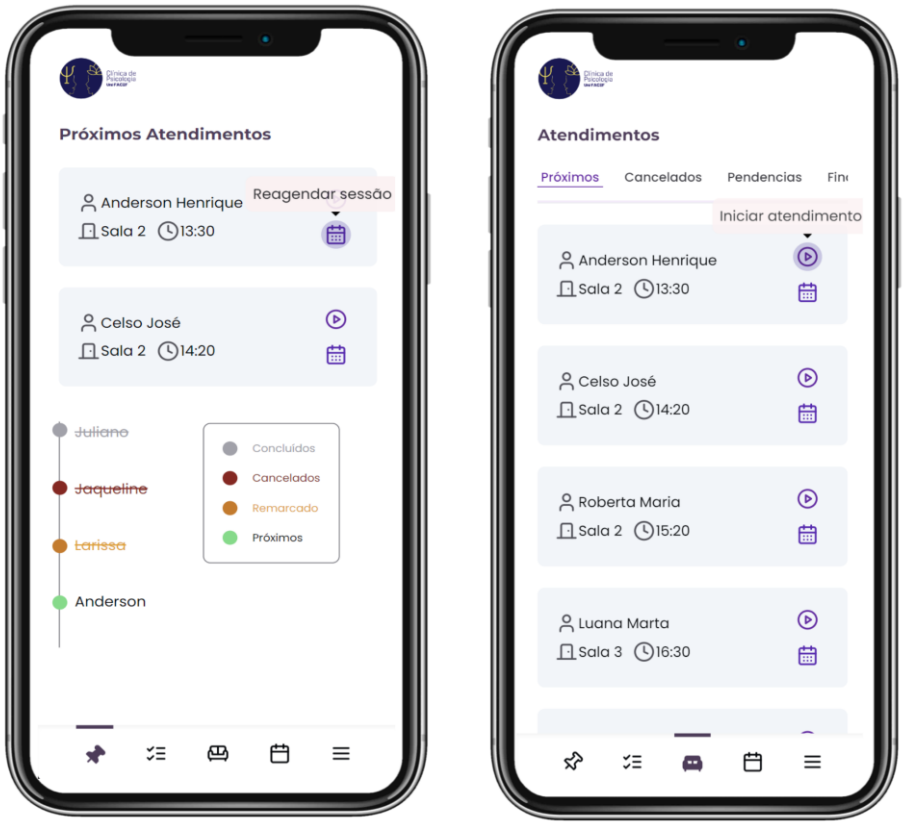

Figura 16 - Funcionalidade de tooltip tela de Próximos Atendimentos e tela de Atendimentos

Fonte: Autoria própria

8 Considerações finais

A idealização deste projeto surgiu após a identificação das dores da clínica de psicologia Uni-FACEF que não tinha uma solução sistêmica que os auxiliasse durante os processos pré-consulta, precisando de realizar os cadastros dos pacientes em planilhas, o agendamento das salas de forma manual, o que muitas vezes causava confusão na clínica. Com o aprofundamento das reuniões de elicitação de requisitos percebemos que além de problemas administrativos a clínica não possuía nenhum sistema para os relatórios de consulta, pois esse processo era feito em papel, o que dificulta a comunicação entre estagiário e supervisor, além de não garantir a segurança dos dados dos pacientes.

Com isso surge um novo objetivo, que foi a criação de uma solução totalmente focada nos processos e particularidades de uma clínica-escola, já que os softwares existentes no mercado não estão preparados para lidar com as demandas específicas desse tipo de clínica.

Apesar de a implementação de um novo sistema seja acompanhada de desafios, acredita-se que este software possa transformar a rotina dos colaboradores da clínica, e acima de tudo dinamizar os processos pré consulta e melhorar a comunicação entre supervisor e estagiário com ao garantir a disponibilidade dos prontuários.

Apesar das funcionalidades principais do sistema terem sido construídas seria necessário a implementação da funcionalidade de login social com as credenciais da Uni-FACEF além do acesso de leitura sobre os alunos e professores de psicologia para que o projeto se tornasse viável, já que muitas funcionalidades principais dependem da parametrização do aluno com um programa de atendimento, além da grande quantidade de colaboradores na clínica, também agrega-se o fato de que todos os anos os alunos formados devem perder acesso ao sistema enquanto os novos universitários devem ser cadastrados.

Estima-se que no futuro, com a implementação das funcionalidades de login e acesso à base de dados, seja implementada uma funcionalidade de “primeiros passos” para que os novos usuários possam se ambientar com o sistema sem necessidade de um programa de treinamento e que o sistema possa ser utilizado como oficial da universidade.

Referências

BANDURA, A. (1997) Self-efficacy: The exercise of control. New York: W.H. Freeman and Company.

BARTIÉ, A. (2002), Garantia da qualidade de software: adquirindo maturidade organizacional. Campus.

BECK, J. S. (2011) Terapia cognitivo-comportamental: teoria e prática. Artmed Editora.

CLERISSI, D. et al. (2017) Towards the generation of end-to-end web test scripts from requirements specifications. IEEE.

COON, D.; MITTERER, J. O. (2010) Introdução à psicologia: Uma abordagem científica. São Paulo: Cengage Learning.

COREY, G. (2017) Theory and Practice of Counseling and Psychotherapy. Cengage Learning.

DOCKER DOCUMENTATION (2021) Docker Overview.

FREUD, S. (1915) A Interpretação dos Sonhos. Editora Imago.

GAMMA et al. (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional

GITLAB DOCUMENTATION (2021) GitLab Overview.

HAMZEI, M. (2020) Docker Image: What It Is and How to Use It. DigitalOcean.

HAN, S.; PARK, J.; KIM, J. (2020) Server-Side Rendering with Next.js: Build and Optimize Your React Apps for SEO, Speed, and User Experience. Birmingham: Packt Publishing. Language User Guide. Addison-Wesley.

HARPER, T. L. (2002). The Systems Bible: The Beginner's Guide to Systems Large and Small. General Systemantics Press.

JACÓ-VILELA. (2015) História da Psicologia no Brasil: uma narrativa por meio de seu ensino.

KOFLER, M.; KRAMER, S. (2019) MySQL: Das umfassende Handbuch. Rheinwerk Computing.

KRUCHTEN, P. et al. (2019) The Agile Manifesto. Agile Alliance.

MALDONADO, R.; ZAMORA, J. (2020) The Top 10 Visual Studio Code Extensions for Web Developers. Sitepoint.

MISCHEL, W. (2014) The Marshmallow Test: Mastering Self-Control. Little, Brown and Company.

PÉREZ ÁLVAREZ, M. (2017) Ciencia y pseudociencia en psicología y psiquiatría. Alianza Editorial

PSICOMANAGER. Sistema de Gestão para Psicólogos. Disponível em: https://www.psicomanager.com.br/. Acesso em: 12 de agosto de 2023.

PRISMA DOCUMENTATION (2021) Prisma: Database Tools for Modern Application Development.

PRESSMAN, R. S. (2014) Engenharia de Software: Uma abordagem profissional. McGraw-Hill.

Project Management Institute (PMI). (2017) Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK®) - Sexta Edição. PMI Publications.

SANTOS, A. B.; SILVA, M. R. (2019) Clínica-Escola: Promovendo a democratização do acesso a serviços de saúde mental. Psicologia em Foco, 13(2), 109-124.

SCHWABER e Sutherland (1995).

SMITH, J. (2018) GitLab: Unificando o Desenvolvimento de Software com DevOps. Revista DevOps, 3(2), 45-56.

SPINELLIS, D. (2012). Version Control Systems. Addison-Wesley.

TORVALDS, L. (2005) Git - Sistema de Controle de Versão Distribuído. Disponível em: https://git-scm.com/.