



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO
GRANDE DO NORTE
TECNOLOGIA EM SISTEMAS PARA INTERNET

JOÃO VITOR OTHON SILVA
KAYKE DE PÁDUA DA SILVA SOUZA
THALES ADRIEL SOARES DE ARAÚJO
WELSON ROSENDO RODRIGUES

Safe Mind

Saúde Mental e Tecnologia

CURRAIS NOVOS / RN
2024



JOÃO VITOR OTHON SILVA
KAYKE DE PÁDUA DA SILVA SOUZA
THALES ADRIEL SOARES DE ARAÚJO
WELSON ROSENDO RODRIGUES

Safe Mind

Saúde Mental e Tecnologia

Este documento refere-se ao desenvolvimento de um site realizado na disciplina de Desenvolvimento Web Back-end, sob orientação do professor Jorge Chrystiann Guimarães da Cunha Nunes, como parte do curso de Tecnologia em Sistemas para Internet, oferecido pelo Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN).

CURRAIS NOVOS / RN

2024

INTRODUÇÃO

Durante os últimos anos, onde a facilidade de acesso às tecnologias e a informação se tornou cada vez mais fácil e evidente, trouxe consigo uma certa dependência entre pessoas e suas tecnologias, que se tornou mais fácil de se obter, e graças ao incidente pandêmico entre os anos de 2019 e 2021, o isolamento social envolvendo toda a população, para fins de evitar um agravamento de mortes pelo mundo, acabou trazendo consigo um mal perigoso, casos de pessoas com ansiedade e depressão se agravou de forma agressiva.

Este projeto tem como objetivo a criação de uma plataforma de auto ajuda para pessoas que se sentem afetadas por esses sintomas, suportando clínicas psicológicas e terapêuticas, de forma online e acessível, destacando a integração estratégica de tecnologias-chave como **HTML**, **CSS**, **Javascript**, **Node.js**, **PostgresSQL** e **MongoDB**.

A combinação de **HTML** e **CSS**, usados como base de construção de uma interface visualmente confortável, atrativa e intuitiva. Utilizando o backend com o auxílio do Node.js, que proporciona estabilidade na construção das camadas de funcionalidades mais técnicas, utilizando juntamente da linguagem de programação Javascript, no qual permite realizar configurações complexas que ajudarão com a funcionalidade das principais funções da plataforma.

Além disso, o sistema utilizará duas estruturas de banco de dados distintas para o armazenamento de informações. O **PostgreSQL**, um banco de dados relacional, será responsável pela gestão dos dados estruturados, como o cadastro de usuários e perfis. Já o **MongoDB**, um banco de dados não relacional, será empregado para armazenar logs de atividades e outros dados menos estruturados.

OBJETIVOS

O principal objetivo deste projeto é desenvolver uma plataforma de autoajuda focada em saúde mental, com ênfase em ansiedade e depressão. A plataforma visa conectar usuários a conteúdos relevantes, como vídeos e artigos sobre saúde mental, e oferecer funcionalidades interativas, como o preenchimento de questionários para autoavaliação e o acesso a profissionais de saúde mental, tudo em um ambiente online acessível e de fácil uso.

Objetivos específicos incluem:

- Proporcionar uma plataforma para agendamentos e acompanhamento psicológico de forma online e segura.
- Criar uma interface intuitiva e amigável para os usuários realizarem autoavaliações.
- Armazenar os dados dos usuários de forma segura, utilizando banco de dados relacional e não relacional.
- Garantir que as interações entre usuários e a plataforma sejam registradas e auditáveis por meio de logs.

JUSTIFICATIVA

A necessidade deste projeto surgiu como uma resposta aos impactos da pandemia de COVID-19, onde o isolamento social resultou no agravamento de casos de ansiedade e depressão em grande parte da população. Muitas pessoas enfrentaram dificuldades em acessar suporte psicológico devido às restrições de mobilidade e às limitações de atendimento presencial.

O projeto visa preencher essa lacuna, oferecendo suporte e orientação através de uma plataforma que combina tecnologia com assistência psicológica. Além disso, a crescente adoção de ferramentas digitais para diversos serviços, incluindo saúde mental, reforça a relevância da proposta. A plataforma oferece uma abordagem prática, combinando o uso de tecnologias Web com a gestão eficiente de dados.

METODOLOGIA

Abordagem Geral

A metodologia aplicada ao desenvolvimento deste projeto seguiu um modelo iterativo e incremental, com foco em desenvolvimento ágil. A escolha dessa abordagem visou garantir flexibilidade na implementação das funcionalidades, permitindo que ajustes e melhorias fossem realizados ao longo do processo, conforme surgissem novas necessidades ou desafios.

Etapas do Desenvolvimento

1. Levantamento de Requisitos

O projeto foi iniciado com uma fase de levantamento de requisitos, onde foram definidas as principais funcionalidades da plataforma, como:

- Cadastro e login de usuários (pacientes e profissionais).
- Seção de vídeos e artigos de autoajuda.
- Área especial para a campanha Setembro Amarelo.
- Agendamento de consultas.

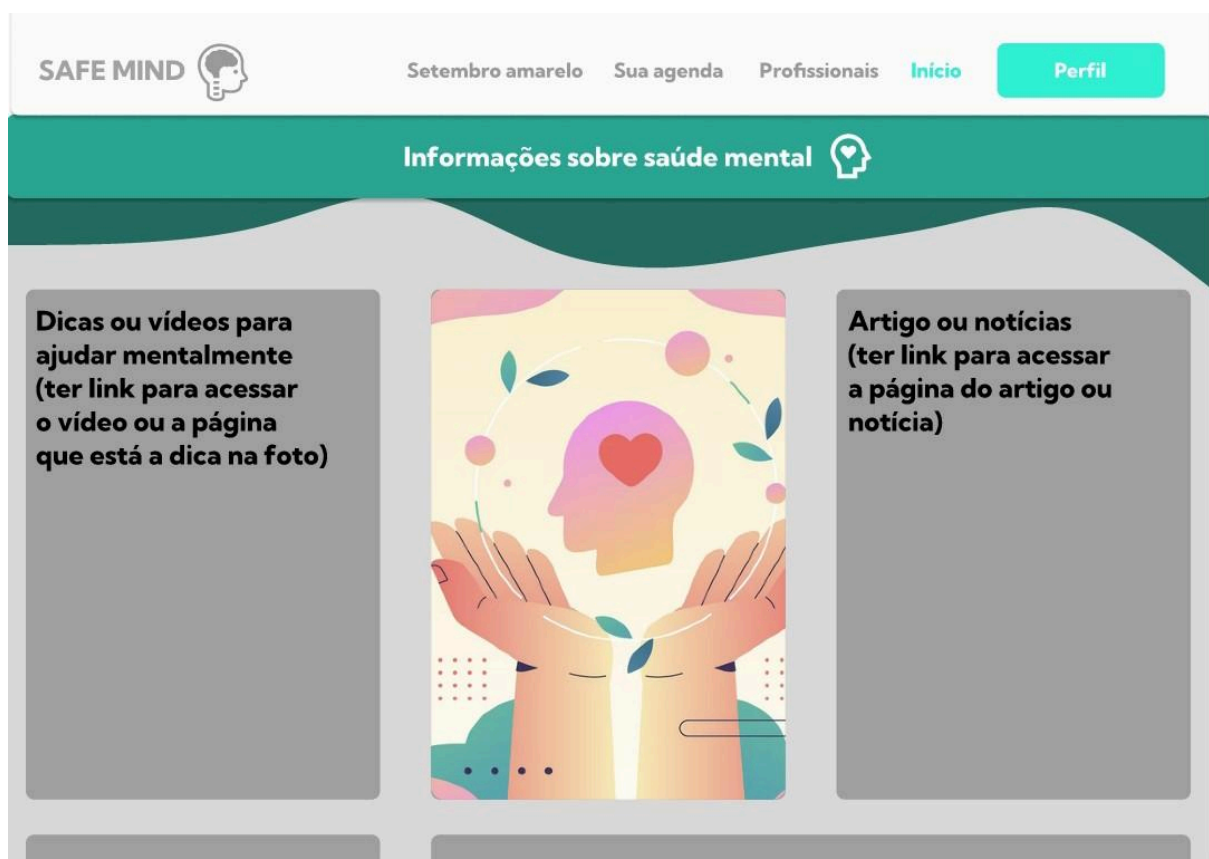
Esses requisitos foram organizados e priorizados para orientar o desenvolvimento das funcionalidades essenciais, criando um escopo claro para o projeto.

2. Prototipação da Interface


Com os requisitos em mãos, o próximo passo foi a criação de protótipos para as interfaces de usuário (UI). Utilizamos ferramentas de design visual, como o **Figma**, para esboçar as telas de cadastro, login, perfis e as seções de conteúdo de autoajuda. A prototipação nos permitiu validar o layout e a usabilidade das páginas antes da codificação, garantindo que a interface fosse acessível e fácil de usar.

Visualização dos Protótipos de Interface:

Home Page



Tela de Cadastro



Cadastro

Nome

Email

Senha


Confirmar Senha

☐ Quero me cadastrar como paciente.


☐ Quero me cadastrar como profissional.

[Continuar](#)

Já tem uma conta? [Login](#)



Tela de Login



Login


Email

Senha


☐ Lembrar login

[Login](#)

Não tem conta ainda? [cadastre-se](#)



Cadastro Profissional



Querido profissional


Se descreva brevemente

Qual função você faz de melhor? Escreva sobre

Qual sua especialidade? Escreva sobre

Como você lida com seus pacientes? Escreva sobre

Cadastrar



Cadastro Paciente



Querido Paciente

Se descreva brevemente

Qual você considera o seu maior problema? Escreva sobre

Quais seus hobbies? Escreva sobre

Quais seus vícios? Escreva sobre

Cadastrar



Perfil Profissional

SAFE MIND

Setembro amareloSua agendaPacientesInícioPerfil

Agendar

Profissional


Descrição breve do paciente

Funções


Especialidades

Comportamento profissional

Perfil Paciente

SAFE MIND

Setembro amareloSua agendaProfissionaisInícioPerfil

Paciente

Descrição breve do paciente

Maior problema

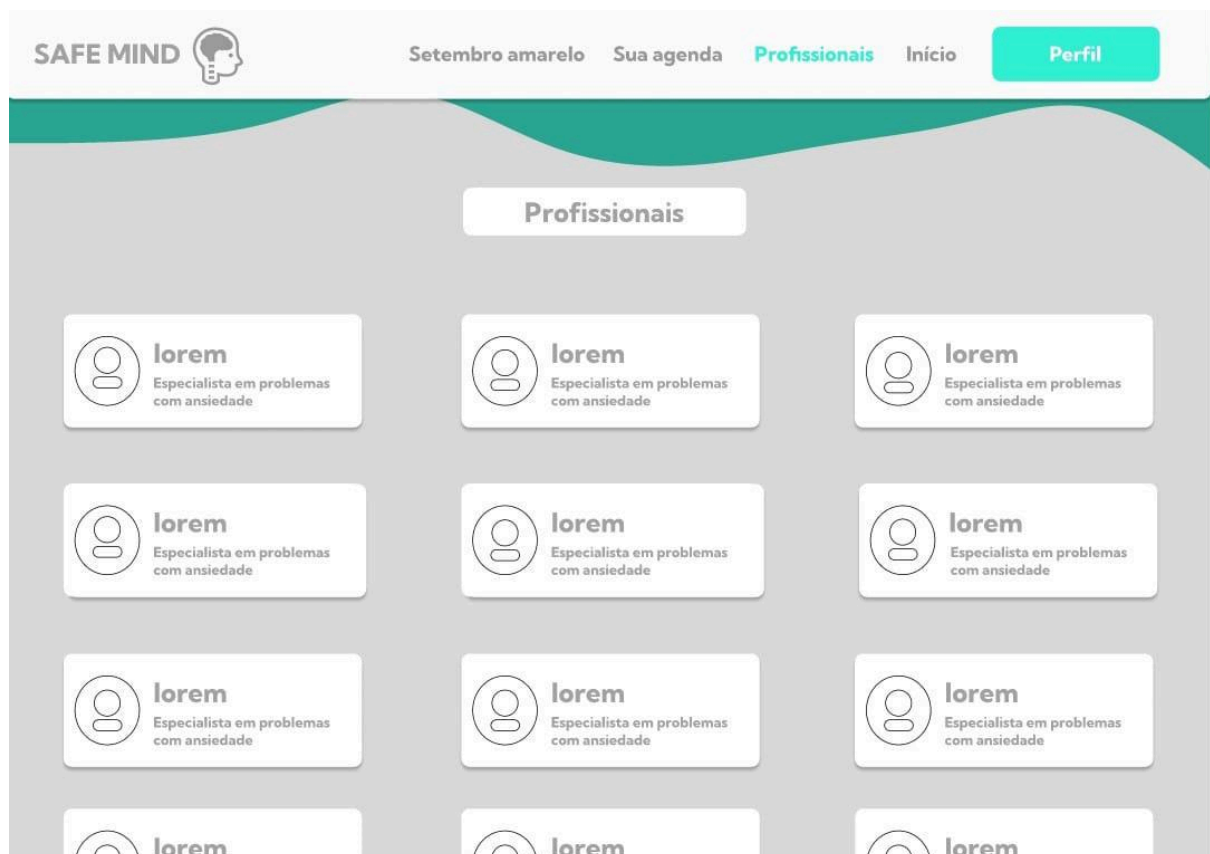
Hobbies

Vícios

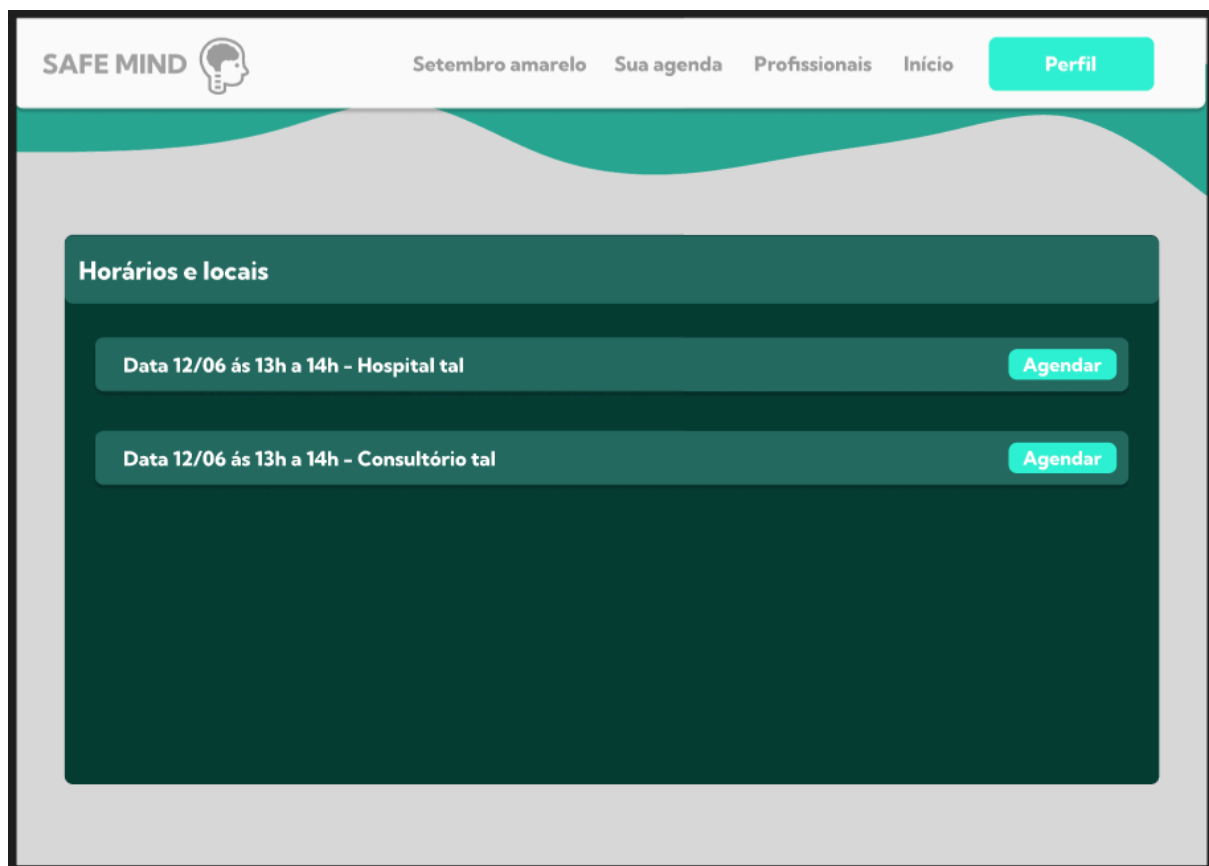
Tela de Setembro Amarelo



Aba de profissionais



Aba que o paciente agenda com o profissional



3. Desenvolvimento do Frontend

O desenvolvimento da interface foi feito utilizando **HTML**, **CSS** e **JavaScript**, assegurando que o frontend fosse responsivo e agradável ao usuário. Os elementos foram organizados semanticamente e visualmente, seguindo boas práticas de acessibilidade e design responsivo, permitindo uma experiência coesa em desktops.

4. Desenvolvimento do Backend

Para o backend, foi escolhido o **Node.js** com o framework **Express.js**, permitindo uma estrutura modular e escalável. O backend foi responsável por:

- Gerenciar as rotas e as APIs necessárias para o frontend se comunicar com os bancos de dados.
- Implementar a lógica de negócios, como autenticação de usuários/profissionais.
- Agendar horário, local e data com banco de dados não relacional.

5. Integração com o PostgreSQL

A plataforma utilizou o **PostgreSQL** como banco de dados relacional para armazenar informações estruturadas dos usuários (dados de login, perfis, etc.). Para facilitar essa integração, instalamos o pacote **pg** no ambiente Node.js, permitindo a conexão direta ao PostgreSQL. Foram criadas funções para realizar inserções, consultas e atualizações no banco de dados, garantindo a persistência e integridade dos dados.

6. Integração com o MongoDB

Além do PostgreSQL, também utilizamos o **MongoDB** como banco de dados não relacional para armazenar logs de atividades do usuário, como ações de login e interações com a plataforma. Usamos o **Mongoose**, uma biblioteca **ODM** (Object Data Modeling) que facilita a interação com o **MongoDB**, definindo esquemas flexíveis para a coleta de dados de logs.

7. Testes e Validação

Após a implementação das principais funcionalidades, foram realizados testes para validar o funcionamento correto do sistema. Testes manuais e automatizados garantiram:

- A correta integração entre o frontend e backend.
- A integridade dos dados estão sendo armazenados no **PostgreSQL** e **MongoDB**.
- A segurança dos dados, com a validação de inputs e prevenção de ataques como SQL Injection e XSS (Cross-site Scripting).

8. Documentação e Revisão

Durante o processo de desenvolvimento, toda a lógica e as tecnologias utilizadas foram devidamente documentadas, permitindo fácil manutenção e expansão futura. Essa documentação inclui o mapeamento dos modelos de dados, as rotas da **API**, as interações do usuário com a plataforma, e o fluxo de desenvolvimento técnico.

Ferramentas Utilizadas

- **Visual Studio Code:** Ambiente de desenvolvimento integrado (IDE) utilizado para codificação.
- **PostgreSQL:** Sistema de gerenciamento de banco de dados relacional.
- **MongoDB:** Banco de dados NoSQL para logs e dados não estruturados.
- **Node.js:** Plataforma para o desenvolvimento do backend.
- **Express.js:** Framework para gerenciar rotas e controle do backend.
- **pg:** Biblioteca utilizada para conectar o Node.js ao PostgreSQL.
- **Mongoose:** Biblioteca ODM para interação com o MongoDB.

Controle de Versão

Utilizamos o **Git** e o **GitHub** para controle de versão, permitindo rastreamento das modificações e colaboração na codificação, além de garantir a integridade do código com histórico de alterações claras

CRONOLOGIA / PLANEJAMENTO

O planejamento foi estruturado em fases, organizando o tempo de forma que cada funcionalidade fosse desenvolvida de maneira progressiva:

1. Fase de Planejamento e Definição dos Requisitos:

- Reunião com os membros do grupo no Discord.
- Definição das funcionalidades principais e arquitetura do sistema.

2. Fase de Desenvolvimento:

- Desenvolvimento do frontend (**HTML**, **CSS** e **JavaScript**).
- Desenvolvimento do backend (**Node.js**).
- Implementação do banco de dados relacional (**PostgreSQL**) e não relacional (**MongoDB**).

3. Fase de Integração e Testes:

- Integração das diferentes partes do sistema.
- Testes de funcionalidades e de segurança.

4. Fase de Documentação e Finalização:

- Finalização da documentação.
- Ajustes finais conforme os feedbacks que tivemos nas reuniões.

RECURSOS E FERRAMENTAS

Para o desenvolvimento do projeto, foram utilizadas as seguintes ferramentas e recursos:

Linguagens e Frameworks:

- **HTML/CSS/JavaScript:** Para a construção do frontend e da interface do usuário.
- **Node.js:** Utilizado no backend para gerenciar as rotas, lógica de negócios e integração com os bancos de dados.
- **Express.js:** Framework utilizado no Node.js para facilitar a criação das rotas e da estrutura da aplicação.

Banco de Dados

- **PostgreSQL:** Banco de dados relacional utilizado para armazenar as informações dos usuários, como cadastro e login.
- **MongoDB:** Banco de dados não relacional utilizado para registrar logs e dados não estruturados.

Outras Ferramentas

- **pgAdmin:** Ferramenta de gerenciamento e administração do banco de dados PostgreSQL.
- **Mongoose:** Biblioteca que simplificou a integração com o MongoDB no Node.js.
- **Visual Studio Code (VSCode):** Editor de código utilizado para o desenvolvimento da aplicação.

DESENVOLVIMENTO

Estrutura Front-End:

O frontend foi desenvolvido utilizando as linguagens **HTML**, **CSS** e **JavaScript**, visando proporcionar uma interface de usuário simples, acessível e responsiva.

- **HTML/CSS:** As páginas de login, cadastro, perfil de usuário e seções de autoajuda (vídeos e artigos) foram construídas utilizando uma estrutura semântica em HTML e estilizadas com CSS, para garantir uma experiência visual agradável e funcional em desktop.
- **JavaScript:** Funções em JavaScript foram implementadas no frontend para adicionar interatividade, como a validação de formulários (email e senha), controle de exibição de mensagens de erro e manipulação de elementos DOM. O JavaScript também foi usado para realizar requisições assíncronas ao backend, garantindo que certas operações, como login, sejam feitas sem a necessidade de recarregar a página.

Backend e Integração com o PostgreSQL:

O backend foi desenvolvido com **Node.js** usando o framework **Express.js**, que oferece uma maneira rápida e eficiente de gerenciar as rotas e controladores do sistema.

1. Criação do Servidor com Node.js:

O servidor foi configurado para receber requisições **HTTP** dos clientes. Rotas foram definidas para operações básicas de usuários, como:

- Cadastro de usuário (**/cadastro**): Recebe as informações do formulário e cria um novo usuário no banco de dados **PostgreSQL**.
- Login (**/login**): Autentica o usuário comparando as credenciais fornecidas com as informações armazenadas no banco.
- Perfil de usuário (**/perfil**): Retorna os dados do usuário autenticado.
- Página Inicial (**/home**): Apresenta as opções de navegação no sistema, como conteúdos de autoajuda, consultas, e a campanha **Setembro Amarelo**.
- Área do Paciente (**/paciente**): Acesso a todas as funcionalidades destinadas aos pacientes, como consultas e autoajuda.

- Perfil do Paciente (`/perfilPaciente`): Exibe as informações detalhadas do paciente, incluindo histórico de consultas e conteúdos de autoajuda acessados.
- Perfil do Profissional (`/perfilProfissional`): Mostra as informações do profissional, incluindo os conteúdos criados e consultas realizadas.
- Área do Profissional (`/profissional`): Centraliza as funcionalidades específicas para os profissionais de saúde mental, como gerenciamento de consultas e criação de conteúdo.
- Página de Profissional (`/paginaProfissional`): Exibe a lista de profissionais disponíveis para agendamento de consultas pelos pacientes.
- Setembro Amarelo (`/setembroAmarelo`): Exibe conteúdos e informações relacionadas à campanha de prevenção ao suicídio, incluindo vídeos e artigos relevantes.

A estrutura de rotas e middlewares foi planejada para garantir uma separação clara entre diferentes funcionalidades.

2. Instalação do **pg** para Conexão com PostgreSQL:

Para integrar o **PostgreSQL** ao **Node.js**, foi utilizado o pacote **pg** (node-postgres), que permite realizar consultas **SQL** diretamente no banco de dados a partir do backend. O arquivo de configuração do **PostgreSQL** foi criado para gerenciar a conexão com o banco. Foi configurado um cliente para enviar consultas **SQL**, como criação de novos usuários e consulta de dados armazenados.

Com essa configuração, conseguimos fazer consultas **SQL** ao banco de dados **PostgreSQL**, como inserção e consulta de dados, garantindo a persistência das informações dos usuários na plataforma.

3. Funções de Manipulação de Dados:

As operações básicas de manipulação de dados no **PostgreSQL** foram encapsuladas em funções específicas, como:

- **Inserção de usuários:** Função responsável por inserir os dados de um novo usuário no banco.

- **Consulta de usuários:** Função que realiza uma busca pelo email do usuário, retornando seus dados.

MongoDB para Armazenamento de Logs:

Para o armazenamento de dados não estruturados e logs de ações dos usuários, como registros de login e interações com a plataforma, optamos pelo uso do **MongoDB**, um banco de dados não relacional. O **MongoDB** é especialmente útil para armazenar documentos JSON flexíveis, o que facilita o gerenciamento de logs, sem a rigidez dos esquemas de dados relacionais.

1. Instalação do Mongoose:

O pacote **mongoose** foi instalado para facilitar a conexão entre o **MongoDB** e o **Node.js**.

2. Criação do Modelo de Logs:

Foi definido um esquema para o armazenamento dos logs no **MongoDB**, incluindo campos como **userId**, **email**, **timestamp** e **IP** do usuário, a fim de registrar as principais atividades na plataforma.

3. Registro de Logs:

Sempre que um usuário realiza uma ação importante, como o login, um registro é criado e salvo no **MongoDB** para fins de auditoria e monitoramento do sistema.

Integração e Testes:

Após a implementação das funcionalidades backend e frontend, foi realizada a integração de todos os componentes. Os testes incluíram:

- Verificação do fluxo de cadastro e login.
- Validação da conexão e persistência dos dados nos dois bancos (**PostgreSQL** para dados de usuários e **MongoDB** para logs).
- Testes de segurança, como validação de entradas de usuários e proteção contra injeção de SQL e XSS.

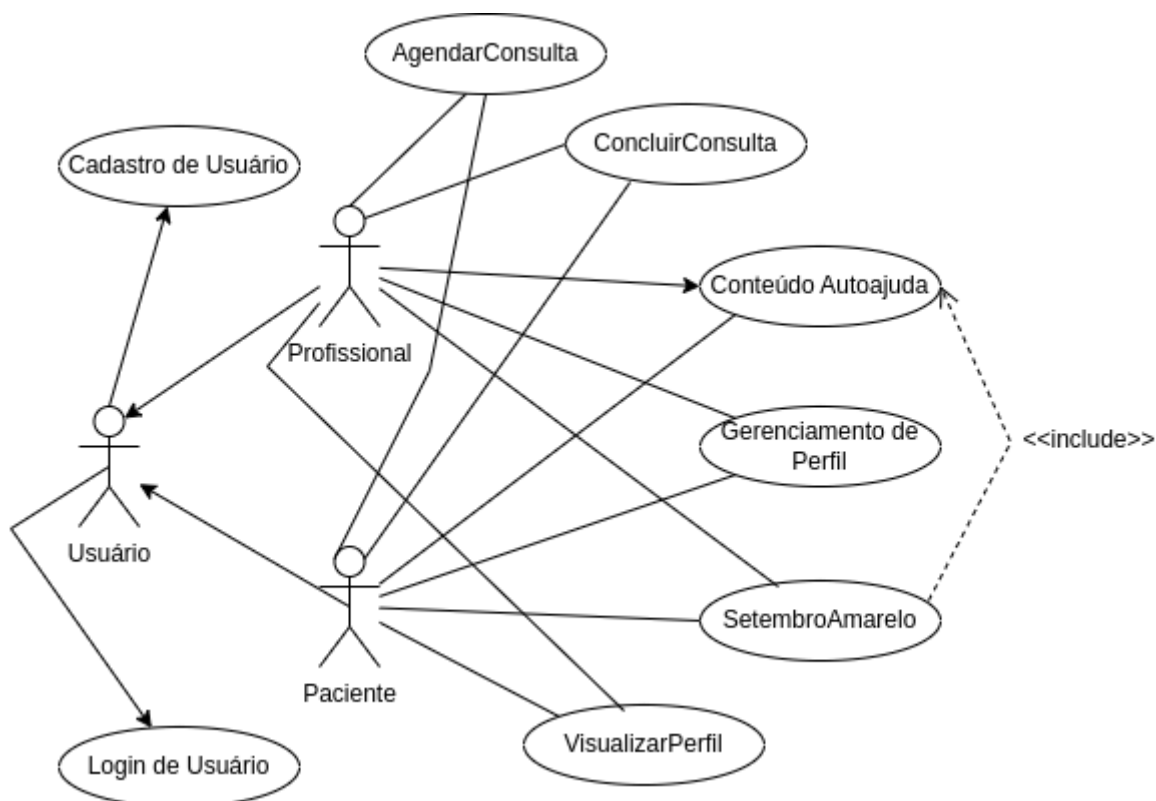
Diagrama de Casos de Uso:

O **Diagrama de Casos de Uso** demonstra as interações entre os diferentes atores do sistema e as principais funcionalidades oferecidas. Nele, identificamos dois atores principais: o **Paciente** e o **Profissional**, que são representações das duas categorias de usuários.

- O **Paciente** pode realizar ações como visualizar o conteúdo de autoajuda, gerenciar seu perfil e participar de campanhas do **Setembro Amarelo**.
- O **Profissional**, além de criar e gerenciar o conteúdo de autoajuda, pode agendar e concluir consultas com os pacientes.

Além disso, o **Usuário** é responsável pelo processo de cadastro e login no sistema. As funcionalidades como **Agendar Consulta** e **Concluir Consulta** são relacionadas diretamente ao profissional, enquanto o paciente interage com as funcionalidades de visualização e autoajuda.

Por fim, o diagrama utiliza uma relação de inclusão (<<include>>) para representar que a funcionalidade do **Setembro Amarelo** está diretamente relacionada à seção de autoajuda, mas com foco específico na campanha de prevenção ao suicídio.



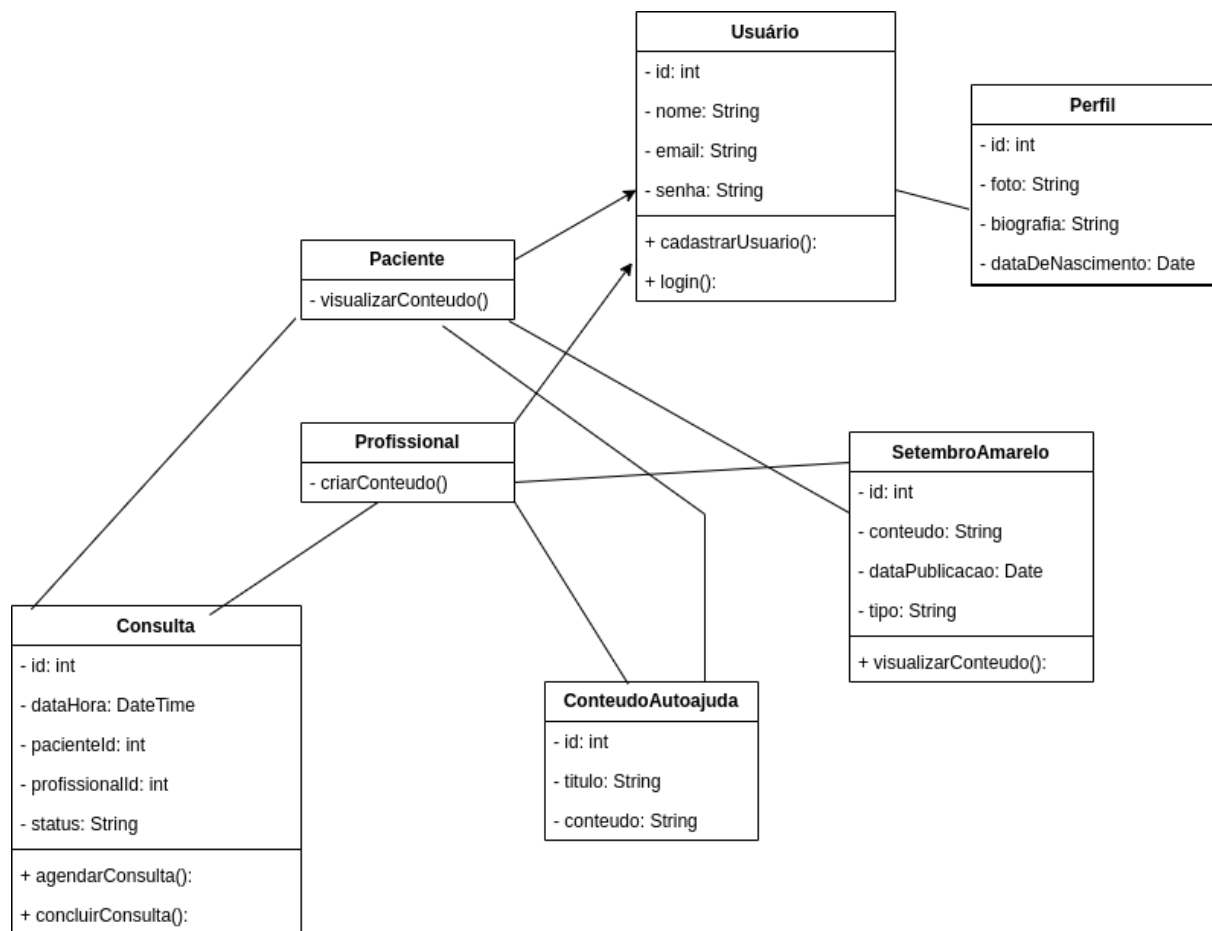
[Diagrama de Casos de Uso]

Diagrama de Classes:

O **Diagrama de Classes** define a estrutura estática do sistema, apresentando as classes principais e seus atributos e métodos. As principais classes modeladas são:

- **Usuário:** Representa tanto pacientes quanto profissionais, contendo informações como nome, email e senha. A classe também possui métodos para cadastro e login de usuários.
- **Paciente e Profissional:** Estas classes herdam da classe **Usuário**, adicionando funcionalidades específicas, como o método `visualizarConteudo()` para o paciente e `criarConteudo()` para o profissional.
- **ConteudoAutoajuda:** Modela os conteúdos criados pelos profissionais e acessados pelos pacientes. Contém atributos como título e conteúdo, além do método `visualizarConteudo()` para que o paciente acesse o material.
- **Consulta:** Representa as consultas entre pacientes e profissionais, com atributos como data e hora, status da consulta, além dos métodos `agendarConsulta()` e `concluirConsulta()` para controle da interação.
- **SetembroAmarelo:** Classe que modela os conteúdos relacionados à campanha de conscientização, com atributos como data de publicação e tipo de conteúdo (vídeo, artigo). Também contém o método `visualizarConteudo()` que permite a interação dos pacientes com os materiais da campanha

Este diagrama é essencial para entender como as entidades do sistema estão organizadas e se relacionam, além de evidenciar os métodos e atributos principais que sustentam a lógica de funcionamento do sistema. Ele mostra as interações entre as classes de maneira clara, permitindo uma visão geral da estrutura de software.



[Diagrama de Classes]

RESULTADOS E DESAFIOS

Resultados

O projeto foi concluído com sucesso, atendendo às especificações iniciais. As principais funcionalidades, como cadastro, login e autoajuda através de vídeos e artigos, foram implementadas com êxito. A integração entre **PostgreSQL** e **MongoDB** permitiu o armazenamento eficiente de dados estruturados e não estruturados, facilitando tanto o gerenciamento de informações dos usuários quanto a auditoria das ações no sistema.

Desafios

- **Integração de dois bancos de dados diferentes:** A utilização de **PostgreSQL** e **MongoDB** simultaneamente foi um dos maiores desafios, exigindo uma coordenação cuidadosa para que ambos os bancos operassem de maneira integrada.

- **Segurança:** A implementação de criptografia para senhas e a proteção contra vulnerabilidades, como injeção de SQL, foram desafios superados através do uso de boas práticas no desenvolvimento.
- **Depreciação de funcionalidades no MongoDB:** Alguns parâmetros, como `useNewUrlParser` e `useUnifiedTopology`, se tornaram obsoletos, o que exigiu ajustes no código para utilizar versões mais recentes das bibliotecas.

CONCLUSÃO

A plataforma de autoajuda desenvolvida representa uma importante ferramenta para aqueles que buscam apoio psicológico, especialmente em um contexto onde o acesso à saúde mental de forma online tem se tornado cada vez mais necessário. A combinação de tecnologias modernas permitiu a criação de uma aplicação robusta e segura, que atende às necessidades de usuários em busca de suporte para problemas como ansiedade e depressão.

O uso de **Node.js**, **PostgreSQL** e **MongoDB** mostrou-se eficaz para lidar com diferentes tipos de dados, garantindo tanto a integridade quanto a flexibilidade no armazenamento de informações. O projeto também destaca a importância de uma interface bem projetada, permitindo que o usuário navegue facilmente pelas funcionalidades da plataforma.

REFERÊNCIAS

FIGMA. *Figma: Web-based interface design tool*. Disponível em: <https://www.figma.com/>. Acesso em: 31 ago. 2024.

PGADMIN. *PostgreSQL Tools*. Disponível em: <https://www.pgadmin.org/>. Acesso em: 07 set. 2024.

MONGODB INC. *MongoDB: The Developer Data Platform*. Disponível em: <https://www.mongodb.com/>. Acesso em: 11 set. 2024.

MICROSOFT. *Visual Studio Code: Code Editing. Redefined*. Disponível em: <https://code.visualstudio.com/>. Acesso em: 31 ago. 2024.

OPENJS FOUNDATION. *Node.js: JavaScript runtime built on Chrome's V8 JavaScript engine*. Disponível em: <https://nodejs.org/>. Acesso em: 31 ago. 2024.

OPENJS FOUNDATION. *Express: Fast, unopinionated, minimalist web framework for Node.js*. Disponível em: <https://expressjs.com/>. Acesso em: 31 ago. 2024.