Documentação de Entrega do Site: Plataforma Digital para Escola Pré-Vestibular

1. Introdução

1.1. Objetivo do Documento

Este documento tem como objetivo principal consolidar todas as informações pertinentes ao projeto de desenvolvimento da plataforma digital para a escola pré-vestibular, servindo como um guia abrangente para todas as partes interessadas, desde a equipe de desenvolvimento e manutenção até os usuários finais e a gestão escolar. Ele detalha o escopo, requisitos, arquitetura, processos de instalação e implantação, aspectos técnicos, manuais de uso, estratégias de teste, planos de manutenção, gestão de riscos e outros elementos cruciais para a compreensão e operação do sistema. A finalidade é garantir a clareza, a rastreabilidade e a sustentabilidade do projeto, facilitando futuras atualizações, expansões e a resolução de eventuais problemas, além de servir como um registro formal da entrega do produto.

1.2. Escopo do Projeto

O projeto consiste no desenvolvimento de um site institucional robusto e dinâmico, concebido para modernizar e otimizar a comunicação entre a escola, seus alunos, pais/responsáveis, professores e a comunidade externa. A plataforma centraliza informações acadêmicas, administrativas, materiais didáticos e notícias, além de proporcionar canais diretos de contato com a administração escolar. O escopo abrange o desenvolvimento das seguintes funcionalidades essenciais:

- **Área de Login e Painéis Diferenciados:** Implementação de um sistema de autenticação seguro que permite o acesso personalizado para alunos, professores e administradores, cada um com um painel de controle adaptado às suas necessidades e permissões específicas.
- **Página de Contato e Formulário para Mensagens:** Criação de uma seção dedicada para que visitantes e usuários possam entrar em contato com a escola, enviando mensagens e solicitando informações de forma direta e eficiente.

- **Administração de Usuários (Cadastro e Permissões):** Desenvolvimento de um módulo completo para que os administradores possam gerenciar o cadastro de novos usuários (alunos, professores, administradores), atribuir e modificar permissões de acesso, garantindo a segurança e a integridade dos dados.

É importante ressaltar que o escopo do projeto **não inclui** o desenvolvimento de um aplicativo móvel dedicado. A interface da plataforma será responsiva, garantindo a acessibilidade e a usabilidade em dispositivos móveis através do navegador web, mas não haverá um aplicativo nativo.

1.3. Visão Geral do Site

A plataforma digital foi concebida como um ecossistema centralizado de informações e serviços, projetado para ser intuitivo, seguro e eficiente. A arquitetura do site é baseada em uma abordagem moderna, utilizando tecnologias de ponta para garantir performance e escalabilidade. A interface do usuário (UI) foi desenvolvida com foco na usabilidade (UX), proporcionando uma experiência de navegação fluida e agradável para todos os perfis de usuários, independentemente de sua familiaridade com tecnologias digitais. O site servirá como o principal ponto de contato digital da escola, consolidando todas as interações e informações em um único ambiente, promovendo a transparência, a inclusão digital e reforçando a presença institucional da escola no ambiente online. A seguir, uma breve descrição das principais áreas e funcionalidades:

- **Página Inicial:** Apresenta as últimas notícias, eventos e informações de destaque da escola, com um design limpo e convidativo.
- **Área do Aluno:** Após o login, os alunos terão acesso a suas notas, boletins, histórico acadêmico e uma biblioteca de materiais didáticos disponibilizados pelos professores.
- **Área do Professor:** Professores poderão gerenciar suas turmas, fazer upload de materiais didáticos, comunicar-se com os alunos e acompanhar o desempenho geral.
- **Área do Administrador:** O painel administrativo oferece controle total sobre o sistema, incluindo gestão de usuários, publicação de notícias, gerenciamento de conteúdo e acesso a relatórios.
- **Página de Notícias:** Um feed atualizado com as últimas novidades e comunicados da escola.
- **Página de Contato:** Informações de contato da escola e um formulário para envio de mensagens.

O design responsivo garante que a experiência do usuário seja consistente e otimizada em diferentes dispositivos, desde desktops e laptops até tablets e smartphones. A

segurança dos dados é uma prioridade, com a implementação de criptografia HTTPS e rigorosos controles de acesso para proteger as informações sensíveis dos usuários.

2. Requisitos do Sistema

Esta seção detalha os requisitos funcionais e não funcionais que guiaram o desenvolvimento da plataforma digital, garantindo que o sistema atenda às necessidades dos usuários e aos padrões de qualidade esperados.

2.1. Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer para atender aos objetivos do projeto. Eles foram levantados através de entrevistas com a diretoria, professores e representantes de pais e alunos, questionários eletrônicos e análise de sites de escolas de referência (benchmark). Os principais requisitos funcionais identificados são:

- **RF01: Gestão de Usuários e Autenticação:** O sistema deve permitir o cadastro de usuários (alunos, professores, administradores) e a autenticação via e-mail e senha. Este requisito garante que apenas usuários autorizados possam acessar as áreas restritas da plataforma, com diferentes níveis de acesso e permissões baseados em seus perfis. A segurança da autenticação é primordial para proteger os dados sensíveis dos usuários e a integridade do sistema.
- **RF02: Acesso a Informações Acadêmicas para Alunos:** Após o login, os alunos devem ter acesso facilitado às suas notas, boletins e materiais didáticos. Esta funcionalidade centraliza o acompanhamento do desempenho acadêmico e a consulta de recursos de estudo, eliminando a necessidade de múltiplos canais de comunicação e garantindo que os alunos tenham todas as informações relevantes em um único local. A interface para visualização desses dados deve ser clara e intuitiva.
- **RF03: Gestão de Conteúdo para Professores:** Professores devem ter a capacidade de disponibilizar materiais didáticos para suas turmas e consultar informações detalhadas sobre as mesmas. Este requisito visa otimizar o processo de compartilhamento de conteúdo educacional, permitindo que os professores façam upload de arquivos, organizem-nos por disciplina ou turma, e acompanhem o progresso de seus alunos de forma eficiente. A consulta de turmas facilita a gestão pedagógica e a comunicação direcionada.
- **RF04: Administração de Usuários e Permissões:** Administradores devem ter total controle sobre a gestão de usuários e suas permissões. Isso inclui a capacidade de

adicionar, editar e remover contas de usuários, bem como ajustar os níveis de acesso e privilégios de cada perfil (aluno, professor, administrador). Este requisito é crucial para a manutenção da segurança do sistema, a conformidade com políticas de privacidade e a flexibilidade na gestão da comunidade escolar.

- **RF05: Publicação e Visualização de Notícias:** O sistema deve permitir que administradores publiquem notícias e comunicados para toda a comunidade escolar. Além disso, todos os usuários (alunos, professores, administradores e visitantes) devem poder visualizar essas notícias de forma organizada e acessível. Esta funcionalidade promove a transparência e mantém a comunidade informada sobre eventos, avisos importantes e atividades da escola.

- **RF06: Página de Contato e Formulário:** O site deve incluir uma página de contato com um formulário para mensagens, permitindo que visitantes e usuários enviem dúvidas, sugestões ou solicitações diretamente para a administração escolar. Esta funcionalidade estabelece um canal de comunicação direto e eficiente, melhorando o atendimento e a interação com a comunidade externa.

2.2. Requisitos Não Funcionais

Os requisitos não funcionais definem os critérios de qualidade e as restrições sob as quais o sistema deve operar. Eles são essenciais para garantir a performance, a segurança, a usabilidade e a compatibilidade da plataforma.

- **RNF01: Performance e Tempo de Carregamento:** O site deve apresentar tempos de carregamento inferiores a 3 segundos para a maioria das páginas, considerando conexões de internet de pelo menos 10 Mbps. Este requisito é fundamental para garantir uma experiência de usuário fluida e evitar a frustração causada por lentidão, o que poderia impactar negativamente a adoção da plataforma.

- **RNF02: Compatibilidade com Navegadores:** A plataforma deve ser totalmente compatível e funcional nos navegadores web mais utilizados, incluindo Google Chrome, Mozilla Firefox e Apple Safari. A compatibilidade garante que a maioria dos usuários possa acessar o sistema sem problemas de visualização ou funcionalidade, independentemente do navegador de sua preferência.

- **RNF03: Segurança e Criptografia de Dados:** Todas as informações sensíveis, como notas, boletins e dados pessoais dos usuários, devem ser protegidas com criptografia HTTPS. Este requisito assegura a confidencialidade e a integridade dos dados transmitidos entre o navegador do usuário e o servidor, protegendo contra interceptações e acessos não autorizados. A segurança é uma prioridade máxima para a proteção da privacidade dos usuários.

- **RNF04: Responsividade da Interface:** A interface do usuário deve ser responsiva, adaptando-se automaticamente a diferentes tamanhos de tela e dispositivos, incluindo desktops, laptops, tablets e smartphones. Este requisito garante que a plataforma seja acessível e ofereça uma experiência de uso otimizada em qualquer dispositivo, sem a necessidade de um aplicativo móvel dedicado.
- **RNF05: Usabilidade e Acessibilidade:** A plataforma deve ser projetada com foco na usabilidade, garantindo que seja intuitiva e fácil de navegar, mesmo para usuários com pouca familiaridade digital. A acessibilidade é um fator chave para promover a inclusão digital, permitindo que todos os membros da comunidade escolar possam utilizar o sistema de forma eficaz e sem dificuldades.

2.3. Restrições

As restrições são fatores que limitam as opções de design e desenvolvimento do projeto. Para este projeto, a principal restrição identificada é:

- **Ausência de Aplicativo Móvel Dedicado:** Conforme mencionado no escopo, o projeto não prevê o desenvolvimento de um aplicativo móvel nativo para iOS ou Android. A acessibilidade via dispositivos móveis será garantida pela responsividade da interface web, que se adapta a diferentes tamanhos de tela e navegadores móveis. Esta restrição foi definida para otimizar recursos e tempo de desenvolvimento, focando na entrega de uma plataforma web robusta e funcional.

3. Arquitetura do Sistema

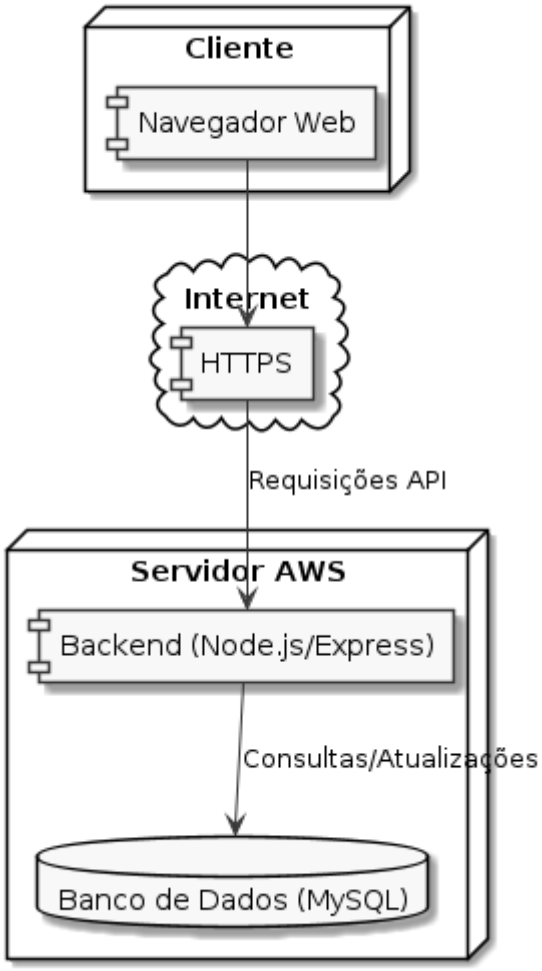
Esta seção descreve a arquitetura técnica da plataforma digital, detalhando os componentes, as tecnologias utilizadas e a forma como eles se interligam para formar um sistema coeso e funcional.

3.1. Diagrama de Arquitetura

A arquitetura do sistema segue um modelo cliente-servidor, com uma clara separação entre o frontend (interface do usuário), o backend (lógica de negócios e API) e o banco de dados. A comunicação entre esses componentes é realizada de forma segura e eficiente.

```
@startuml
skinparam monochrome true

node "Cliente" {
  [Navegador Web]
}
```

```
cloud "Internet" {
  [HTTPS]
}

node "Servidor AWS" {
  component "Backend (Node.js/Express)" as Backend
  database "Banco de Dados (MySQL)" as DB
}

[Navegador Web] --> HTTPS
HTTPS --> Backend : Requisições API
Backend --> DB : Consultas/Atualizações

@enduml
```



Descrição do Fluxo:

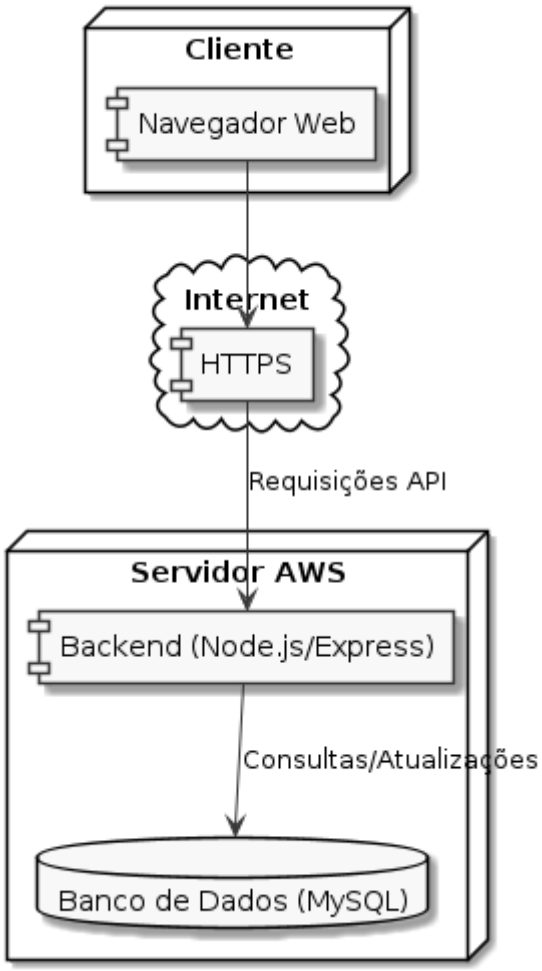
1. **Cliente (Navegador Web):** O usuário interage com a interface do site através de um navegador web (Chrome, Firefox, Safari). O frontend, construído com React, é carregado no navegador do cliente.

2. **Internet (HTTPS):** Todas as comunicações entre o navegador do cliente e o servidor são criptografadas via HTTPS, garantindo a segurança e a privacidade dos dados transmitidos.
3. **Servidor AWS:** A aplicação backend e o banco de dados são hospedados na Amazon Web Services (AWS), uma infraestrutura de nuvem robusta e escalável.
    - **Backend (Node.js/Express):** Responsável por processar as requisições do cliente, implementar a lógica de negócios, interagir com o banco de dados e fornecer as respostas (APIs) para o frontend.
    - **Banco de Dados (MySQL):** Armazena todos os dados da aplicação, incluindo informações de usuários, notícias, notas, eventos e materiais didáticos.

3.2. Diagrama de Implantação

O diagrama de implantação ilustra a distribuição física dos componentes de software nos nós de hardware, mostrando como o sistema é implantado no ambiente de produção.

```
@startuml
node "Servidor Web (AWS EC2)" as Server {
  component "Aplicação Frontend (React)" as FrontendApp
  component "Aplicação Backend (Node.js/Express)" as BackendApp
}

node "Servidor de Banco de Dados (AWS RDS)" as DBServer {
  database "Banco de Dados MySQL" as MySQLDB
}

cloud "Rede do Usuário" as UserNetwork {
  artifact "Navegador Web" as Browser
}

UserNetwork -- Server : HTTPS
FrontendApp -- BackendApp : API Calls
BackendApp -- MySQLDB : JDBC/ODBC

@enduml
```



3.3. Tecnologias Utilizadas

O projeto foi desenvolvido utilizando um conjunto de tecnologias modernas e amplamente adotadas no mercado, garantindo robustez, escalabilidade e facilidade de manutenção:

- **Frontend:**
  - **HTML5:** Linguagem de marcação para estruturação do conteúdo web.
  - **CSS3:** Linguagem de estilo para apresentação visual e design responsivo.
  - **JavaScript:** Linguagem de programação para interatividade no lado do cliente.
  - **React:** Biblioteca JavaScript para construção de interfaces de usuário dinâmicas e reativas.
- **Backend:**
  - **Node.js:** Ambiente de execução JavaScript server-side.
  - **Express:** Framework web para Node.js, utilizado para construir APIs RESTful de forma rápida e eficiente.
- **Banco de Dados:**
  - **MySQL:** Sistema de gerenciamento de banco de dados relacional (SGBDR) para armazenamento persistente dos dados da aplicação.

- **Hospedagem e Infraestrutura:**
  - **AWS (Amazon Web Services):** Plataforma de computação em nuvem para hospedagem do servidor (EC2) e do banco de dados (RDS).
  - **HTTPS:** Protocolo de comunicação segura para garantir a integridade e confidencialidade dos dados transmitidos.
- **Ferramentas de Desenvolvimento e Gestão:**
  - **Git/GitHub:** Sistema de controle de versão distribuído para gerenciamento do código-fonte e colaboração da equipe.
  - **Trello:** Ferramenta de gestão de projetos ágeis (Scrum) para organização de tarefas, sprints e backlog.
  - **Figma:** Ferramenta de design colaborativo para prototipagem (wireframes e mockups) da interface do usuário.
  - **Visual Studio Code (VS Code):** Ambiente de desenvolvimento integrado (IDE) para codificação.

3.4. Estrutura de Pastas

A estrutura de pastas do projeto foi organizada de forma modular e padronizada para facilitar a navegação, o desenvolvimento e a manutenção do código. A seguir, uma representação simplificada da estrutura:

```
/projeto-plataforma-escola
├── backend/
│   ├── src/
│   │   ├── controllers/
│   │   ├── models/
│   │   ├── routes/
│   │   ├── services/
│   │   └── app.js
│   ├── package.json
│   └── .env
├── frontend/
│   ├── public/
│   ├── src/
│   │   ├── components/
│   │   ├── pages/
│   │   ├── services/
│   │   ├── App.js
│   │   └── index.js
│   ├── package.json
│   └── .env
├── docs/
│   ├── diagrams/
│   ├── api-docs/
│   └── README.md
├── .gitignore
```

```
├── README.md
└── package.json (root)
```

- **`backend/`**: Contém todo o código-fonte e configurações do servidor Node.js/ Express.
  - `src/`: Código-fonte da aplicação backend, dividido por responsabilidades (controladores, modelos, rotas, serviços).
  - `package.json`: Dependências e scripts do backend.
  - `.env`: Variáveis de ambiente para o backend.
- **`frontend/`**: Contém todo o código-fonte e configurações da aplicação React.
  - `public/`: Arquivos estáticos (HTML, imagens, etc.).
  - `src/`: Código-fonte da aplicação frontend, dividido por componentes, páginas e serviços.
  - `package.json`: Dependências e scripts do frontend.
  - `.env`: Variáveis de ambiente para o frontend.
- **`docs/`**: Documentação do projeto, incluindo diagramas, documentação de API e outros documentos relevantes.
- **`.gitignore`**: Arquivo para ignorar arquivos e pastas que não devem ser versionados pelo Git.
- **`README.md`**: Arquivo de leitura principal do projeto.
- **`package.json (root)`**: Gerenciador de pacotes para o projeto completo, com scripts para iniciar ambas as aplicações.

3.5. Dependências

As principais dependências do projeto, gerenciadas via `npm` (Node Package Manager), incluem:

Backend: * `express`: Framework web para Node.js. * `mysql2`: Driver MySQL para Node.js. * `dotenv`: Para carregar variáveis de ambiente. * `bcryptjs`: Para criptografia de senhas. * `jsonwebtoken`: Para autenticação baseada em tokens. * `cors`: Para habilitar Cross-Origin Resource Sharing.

Frontend: * `react`: Biblioteca principal do React. * `react-dom`: Pacote para renderização do React no DOM. * `react-router-dom`: Para roteamento no lado do cliente. * `axios`: Cliente HTTP para fazer requisições a APIs. * `tailwindcss`: Framework CSS para estilização rápida e responsiva. * `react-icons`: Biblioteca de ícones para React.

4. Guia de Instalação e Configuração

Este guia fornece as instruções necessárias para configurar o ambiente de desenvolvimento, instalar as dependências do projeto e realizar o deploy da aplicação.

4.1. Pré-requisitos

Antes de iniciar a instalação, certifique-se de que os seguintes softwares e ferramentas estejam instalados em sua máquina:

- **Node.js:** Versão 14.x ou superior. Pode ser baixado em nodejs.org.
- **npm (Node Package Manager):** Geralmente vem junto com a instalação do Node.js. Verifique a versão com `npm -v`.
- **MySQL Server:** Versão 8.x ou superior. Pode ser baixado em mysql.com.
- **Git:** Para clonar o repositório do projeto. Pode ser baixado em git-scm.com.
- **Editor de Código:** Recomendamos o Visual Studio Code (VS Code) para desenvolvimento.

4.2. Instruções de Instalação

Siga os passos abaixo para configurar o projeto localmente:

1. **Clonar o Repositório:** Abra o terminal ou prompt de comando e execute o seguinte comando para clonar o repositório do projeto: `bash git clone [URL_DO_REPOSITORIO] cd projeto-plataforma-escola` Substitua `[URL_DO_REPOSITORIO]` pela URL real do seu repositório Git.

2. **Instalar Dependências do Backend:** Navegue até o diretório `backend` e instale as dependências: `bash cd backend npm install`

3. **Instalar Dependências do Frontend:** Navegue até o diretório `frontend` e instale as dependências: `bash cd ../frontend npm install`

4.3. Configuração do Ambiente

Para que a aplicação funcione corretamente, é necessário configurar as variáveis de ambiente para o backend e o frontend.

1. **Configuração do Backend:** No diretório `backend`, crie um arquivo chamado `.env` e adicione as seguintes variáveis, substituindo os valores pelos dados do seu ambiente MySQL: `DB_HOST=localhost DB_USER=root DB_PASSWORD=sua_senha_mysql DB_NAME=nome_do_banco_de_dados`

`JWT_SECRET=sua_chave_secreta_jwt PORT=3001` Certifique-se de que o *`DB_NAME`* corresponda ao nome do banco de dados que você criou para o projeto.

2. **Configuração do Frontend:** No diretório `frontend`, crie um arquivo chamado `.env` e adicione a seguinte variável: `REACT_APP_API_URL=http://localhost:3001` Esta variável aponta para o endereço do seu backend local.

3. **Configuração do Banco de Dados:**

    - Crie um banco de dados MySQL com o nome especificado em `DB_NAME` no arquivo `.env` do backend.
    - Execute os scripts SQL de criação de tabelas e inserção de dados iniciais. Estes scripts devem estar localizados em um diretório como `backend/database/` (ex: `schema.sql`, `data.sql`). `bash mysql -u root -p nome_do_banco_de_dados < path/to/your/schema.sql mysql -u root -p nome_do_banco_de_dados < path/to/your/data.sql` Substitua *`path/to/your/schema.sql`* e *`path/to/your/data.sql`* pelos caminhos reais dos seus arquivos SQL.

4.4. Instruções de Deploy

O deploy da aplicação é automatizado via pipeline CI/CD utilizando GitHub Actions para o servidor AWS. As instruções a seguir descrevem o processo de deploy para ambiente de produção.

1. **Configuração do Repositório GitHub:**

    - Certifique-se de que seu código esteja no GitHub.
    - Crie os segredos necessários no repositório GitHub (Settings > Secrets > Actions) para as credenciais da AWS (ex: `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, `AWS_REGION`, `EC2_HOST`, `EC2_USER`).

2. **Configuração do GitHub Actions:**

    - O arquivo de workflow do GitHub Actions (`.github/workflows/deploy.yml`) deve estar configurado para:
        - Construir a aplicação frontend e backend.
        - Realizar testes automatizados.
        - Fazer o deploy do backend para uma instância EC2 na AWS.
        - Fazer o deploy do frontend para um serviço de hospedagem estática (ex: AWS S3 + CloudFront).

3. **Processo de Deploy:**
    - A cada push para a branch `main` (ou outra branch configurada para deploy), o GitHub Actions será acionado automaticamente.
    - O pipeline executará os passos de build, teste e deploy, garantindo que a versão mais recente da aplicação seja disponibilizada no ambiente de produção.
    - Monitorar o status do deploy através da aba "Actions" no seu repositório GitHub.

Exemplo de estrutura de um workflow de deploy simplificado (`.github/workflows/deploy.yml`):

```yaml
name: Deploy to AWS

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '16'

    - name: Install backend dependencies
      run: npm install
      working-directory: ./backend

    - name: Build frontend
      run: npm install && npm run build
      working-directory: ./frontend

    - name: Deploy backend to EC2
      uses: appleboy/ssh-action@master
      with:
        host: ${{ secrets.EC2_HOST }}
        username: ${{ secrets.EC2_USER }}
        key: ${{ secrets.SSH_PRIVATE_KEY }}
        script: |
          cd /var/www/html/backend
```

```
            git pull origin main
            npm install --production
            pm2 restart backend-app

      - name: Deploy frontend to S3
        uses: jakejarvis/s3-sync-action@master
        with:
          args: --acl public-read --follow-symlinks --delete
        env:
          AWS_S3_BUCKET: ${{ secrets.AWS_S3_BUCKET }}
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: $
{{ secrets.AWS_SECRET_ACCESS_KEY }}
          AWS_REGION: ${{ secrets.AWS_REGION }}
          SOURCE_DIR: 'frontend/build'
```

Nota: Este é um exemplo simplificado. O workflow real pode incluir etapas adicionais como testes de integração, migrações de banco de dados, e invalidação de cache do CloudFront.

5. Documentação Técnica

Esta seção aprofunda nos aspectos técnicos do projeto, fornecendo detalhes sobre o código-fonte, as APIs, o modelo de dados e os diagramas que ilustram o funcionamento interno do sistema.

5.1. Código Fonte

O código-fonte do projeto está organizado em dois diretórios principais: `backend` e `frontend`, conforme detalhado na seção de Estrutura de Pastas. Ambos os repositórios seguem as melhores práticas de desenvolvimento, com código limpo, modular e bem comentado. O controle de versão é realizado via Git/GitHub, garantindo a rastreabilidade das alterações e facilitando a colaboração da equipe.

- **Backend:** Desenvolvido em Node.js com Express, o backend é responsável pela lógica de negócios, autenticação, autorização, interação com o banco de dados e exposição dos endpoints da API. O código é estruturado em camadas (controllers, models, services, routes) para promover a separação de responsabilidades e a manutenibilidade.

- **Frontend:** Construído com React, o frontend é a interface do usuário da aplicação. O código é organizado em componentes reutilizáveis, páginas e serviços para consumo da API. A estilização é feita com Tailwind CSS, garantindo um design responsivo e consistente em diferentes dispositivos.

O acesso ao código-fonte completo é feito através do repositório Git, onde cada funcionalidade é desenvolvida em branches separadas e integrada à branch principal (`main`) após revisão de código e testes. A documentação inline (comentários no código) é utilizada para explicar a lógica de funções complexas e a finalidade de cada módulo.

5.2. Endpoints de API

A comunicação entre o frontend e o backend é realizada através de uma API RESTful. Abaixo, uma lista dos principais endpoints disponíveis, com seus respectivos métodos HTTP e uma breve descrição:

- **Autenticação:**
  - `POST /api/auth/login` : Realiza o login do usuário, retornando um token JWT em caso de sucesso.
  - `POST /api/auth/register` : Registra um novo usuário no sistema.
- **Usuários:**
  - `GET /api/users` : Retorna a lista de todos os usuários (requer permissão de administrador).
  - `GET /api/users/:id` : Retorna os detalhes de um usuário específico.
  - `PUT /api/users/:id` : Atualiza os dados de um usuário (requer permissão de administrador ou ser o próprio usuário).
  - `DELETE /api/users/:id` : Remove um usuário (requer permissão de administrador).
- **Notícias:**
  - `GET /api/news` : Retorna a lista de todas as notícias.
  - `GET /api/news/:id` : Retorna os detalhes de uma notícia específica.
  - `POST /api/news` : Cria uma nova notícia (requer permissão de administrador).
  - `PUT /api/news/:id` : Atualiza uma notícia existente (requer permissão de administrador).
  - `DELETE /api/news/:id` : Remove uma notícia (requer permissão de administrador).
- **Notas (para Alunos):**
  - `GET /api/students/:id/grades` : Retorna as notas de um aluno específico (requer permissão de aluno ou administrador).

- **Materiais Didáticos (para Professores):**
    - `GET /api/materials`: Retorna a lista de todos os materiais didáticos.
    - `POST /api/materials`: Faz upload de um novo material didático (requer permissão de professor).
    - `DELETE /api/materials/:id`: Remove um material didático (requer permissão de professor).
- **Turmas (para Professores):**
    - `GET /api/teachers/:id/classes`: Retorna as turmas de um professor específico (requer permissão de professor).

Cada endpoint é protegido por autenticação e autorização, garantindo que apenas usuários com as permissões adequadas possam acessá-los. A documentação completa da API pode ser gerada automaticamente a partir do código-fonte utilizando ferramentas como Swagger/OpenAPI, que podem ser integradas ao projeto para facilitar o consumo por outras aplicações ou equipes.

5.3. Modelos de Dados

Os modelos de dados representam a estrutura das informações armazenadas no banco de dados MySQL. Eles são a base para a criação das tabelas e o relacionamento entre elas. Os principais modelos de dados são:

- **Usuario:**
    - `id_usuario` (INT, PK, Auto Increment)
    - `nome` (VARCHAR(255))
    - `email` (VARCHAR(255), UNIQUE)
    - `senha` (VARCHAR(255)) - Armazenada como hash
    - `tipo` (ENUM('aluno', 'professor', 'admin'))
    - `created_at` (DATETIME)
    - `updated_at` (DATETIME)
- **Noticia:**
    - `idnoticia` (INT, PK, Auto Increment)
    - `titulo` (VARCHAR(255))
    - `conteudo` (TEXT)
    - `datapublicacao` (DATETIME)
    - `id_autor` (INT, FK para Usuario.id_usuario)
    - `created_at` (DATETIME)

    - `updated_at` (DATETIME)

- **Nota:**
    - `idnota` (INT, PK, Auto Increment)
    - `idaluno` (INT, FK para Usuario.id_usuario)
    - `disciplina` (VARCHAR(100))
    - `valor` (DECIMAL(4,2))
    - `data` (DATE)
    - `created_at` (DATETIME)
    - `updated_at` (DATETIME)

- **Evento:**
    - `idevento` (INT, PK, Auto Increment)
    - `titulo` (VARCHAR(255))
    - `descricao` (TEXT)
    - `datainicio` (DATETIME)
    - `data_fim` (DATETIME)
    - `created_at` (DATETIME)
    - `updated_at` (DATETIME)

- **MaterialDidatico:**
    - `id_material` (INT, PK, Auto Increment)
    - `titulo` (VARCHAR(255))
    - `descricao` (TEXT)
    - `caminho_arquivo` (VARCHAR(255))
    - `id_professor` (INT, FK para Usuario.id_usuario)
    - `created_at` (DATETIME)
    - `updated_at` (DATETIME)

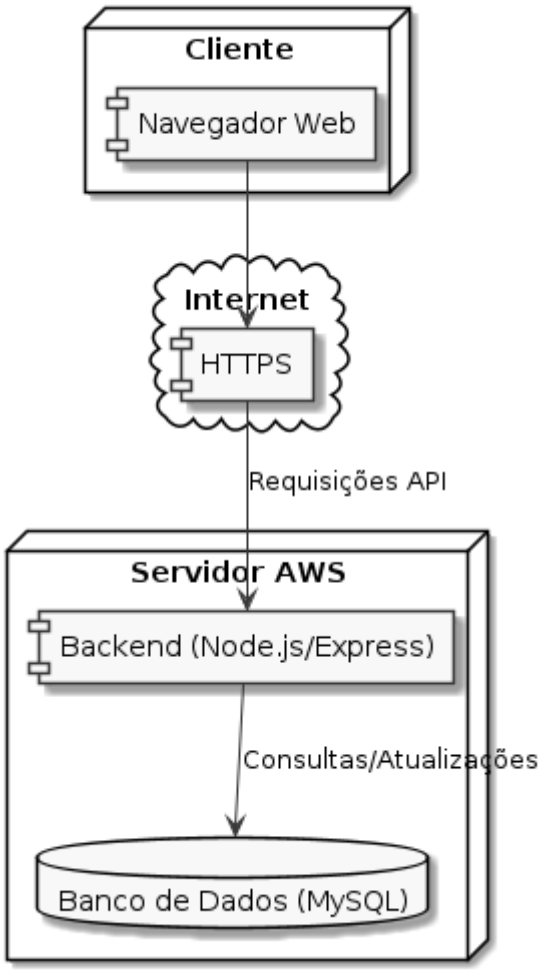
- **Turma:**
    - `id_turma` (INT, PK, Auto Increment)
    - `nome_turma` (VARCHAR(100))
    - `ano` (INT)
    - `created_at` (DATETIME)
    - `updated_at` (DATETIME)### 5.4. Diagrama de Entidade-Relacionamento (ERD)

5.4. Diagrama de Entidade-Relacionamento (ERD)

O Diagrama de Entidade-Relacionamento (ERD) visualiza a estrutura do banco de dados, mostrando as entidades (tabelas), seus atributos e os relacionamentos entre elas. Este diagrama é crucial para a compreensão do esquema do banco de dados e para a manutenção da integridade dos dados.

```
@startuml
entity Usuario {
  *id_usuario: INT <<PK>>
  --
  nome: VARCHAR(255)
  email: VARCHAR(255) <<UNIQUE>>
  senha: VARCHAR(255)
  tipo: ENUM(\'aluno\', \'professor\', \'admin\')
  created_at: DATETIME
  updated_at: DATETIME
}

entity Noticia {
  *idnoticia: INT <<PK>>
  --
  titulo: VARCHAR(255)
  conteudo: TEXT
  datapublicacao: DATETIME
  id_autor: INT <<FK>>
  created_at: DATETIME
  updated_at: DATETIME
}

entity Nota {
  *idnota: INT <<PK>>
  --
  idaluno: INT <<FK>>
  disciplina: VARCHAR(100)
  valor: DECIMAL(4,2)
  data: DATE
  created_at: DATETIME
  updated_at: DATETIME
}

entity Evento {
  *idevento: INT <<PK>>
  --
  titulo: VARCHAR(255)
  descricao: TEXT
  datainicio: DATETIME
  data_fim: DATETIME
  created_at: DATETIME
  updated_at: DATETIME
```

```
}

entity MaterialDidatico {
  *id_material: INT <<PK>>
  --
  titulo: VARCHAR(255)
  descricao: TEXT
  caminho_arquivo: VARCHAR(255)
  id_professor: INT <<FK>>
  created_at: DATETIME
  updated_at: DATETIME
}

entity Turma {
  *id_turma: INT <<PK>>
  --
  nome_turma: VARCHAR(100)
  ano: INT
  created_at: DATETIME
  updated_at: DATETIME
}

Usuario ||--o{ Noticia : "publica"
Usuario ||--o{ Nota : "recebe"
Usuario ||--o{ MaterialDidatico : "disponibiliza"
Usuario ||--o{ Turma : "leciona/pertence"

@enduml
```

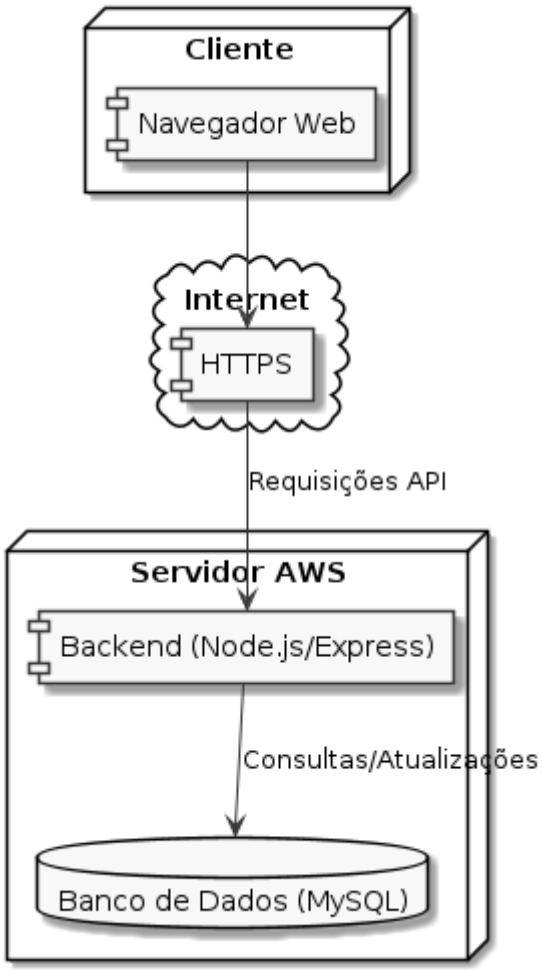


5.5. Diagrama de Fluxo de Dados (DFD)

O Diagrama de Fluxo de Dados (DFD) ilustra como os dados são processados e transferidos dentro do sistema. Ele mostra as entradas e saídas de dados, os processos que transformam esses dados e os armazenamentos de dados.

```
@startuml
skinparam monochrome true

actor Aluno
actor Professor
actor Administrador
actor Visitante

rectangle "Sistema da Escola" {
  usecase "Autenticação" as UC_Auth
  usecase "Gestão de Usuários" as UC_UserMgmt
  usecase "Acesso a Notas" as UC_Grades
  usecase "Gestão de Materiais Didáticos" as UC_Materials
  usecase "Publicação de Notícias" as UC_NewsPub
  usecase "Visualização de Notícias" as UC_NewsView
  usecase "Contato" as UC_Contact
}
```

```
database "Banco de Dados" as DB

Aluno -- UC_Auth
Professor -- UC_Auth
Administrador -- UC_Auth

Administrador -- UC_UserMgmt

Aluno -- UC_Grades
UC_Grades -- DB : Notas

Professor -- UC_Materials
UC_Materials -- DB : Materiais

Administrador -- UC_NewsPub
UC_NewsPub -- DB : Notícias

Aluno -- UC_NewsView
Professor -- UC_NewsView
Administrador -- UC_NewsView
Visitante -- UC_NewsView
UC_NewsView -- DB : Notícias

Visitante -- UC_Contact
UC_Contact -- DB : Mensagens

@enduml
```

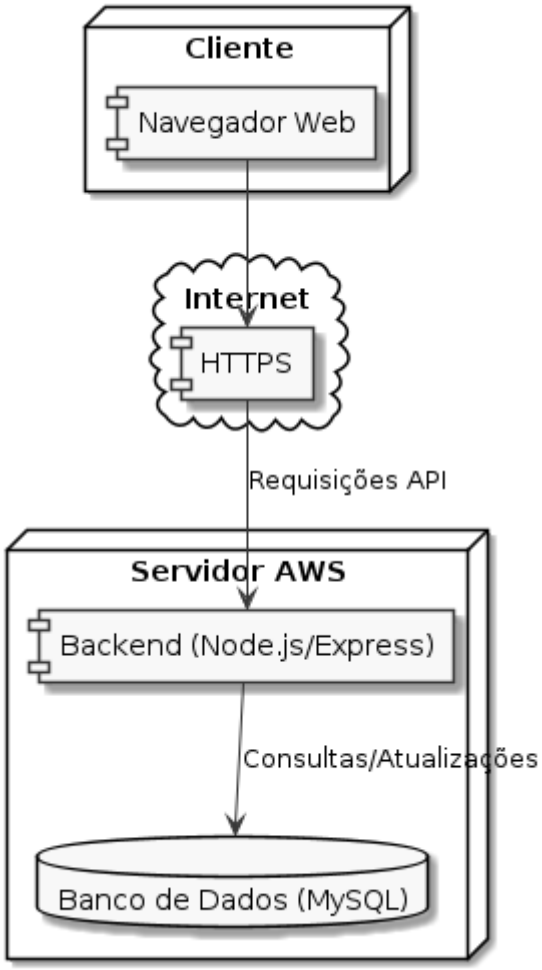


5.6. Diagrama de Sequência: Acesso a Notas do Aluno

Este diagrama detalha a sequência de interações para um aluno acessar suas notas.

```
@startuml
actor Aluno
participant "Navegador Web" as Browser
participant "Frontend (React)" as Frontend
participant "Backend (Node.js/Express)" as Backend
participant "Banco de Dados (MySQL)" as DB

Aluno -> Browser: Acessa a plataforma
Browser -> Frontend: Carrega página de login
Aluno -> Frontend: Insere credenciais e clica em "Login"
Frontend -> Backend: Requisição POST /api/auth/login (email,
senha)
Backend -> DB: Consulta Usuario (email, senha)
DB --> Backend: Retorna dados do usuário
Backend --> Frontend: Resposta 200 OK (token JWT, dados do
usuário)
Frontend -> Browser: Armazena token, redireciona para painel do
aluno

Aluno -> Browser: Clica em "Minhas Notas"
```

```
Browser -> Frontend: Requisição GET /notas
Frontend -> Backend: Requisição GET /api/students/{id}/grades
(com token JWT)
Backend -> Backend: Valida token JWT
Backend -> DB: Consulta Notas (id_aluno)
DB --> Backend: Retorna notas do aluno
Backend --> Frontend: Resposta 200 OK (lista de notas)
Frontend -> Browser: Exibe notas na tela
@enduml
```



5.7. Fluxos de Trabalho

Os fluxos de trabalho descrevem as etapas sequenciais para a execução de tarefas específicas dentro do sistema, detalhando as responsabilidades e as interações entre os usuários e o sistema.

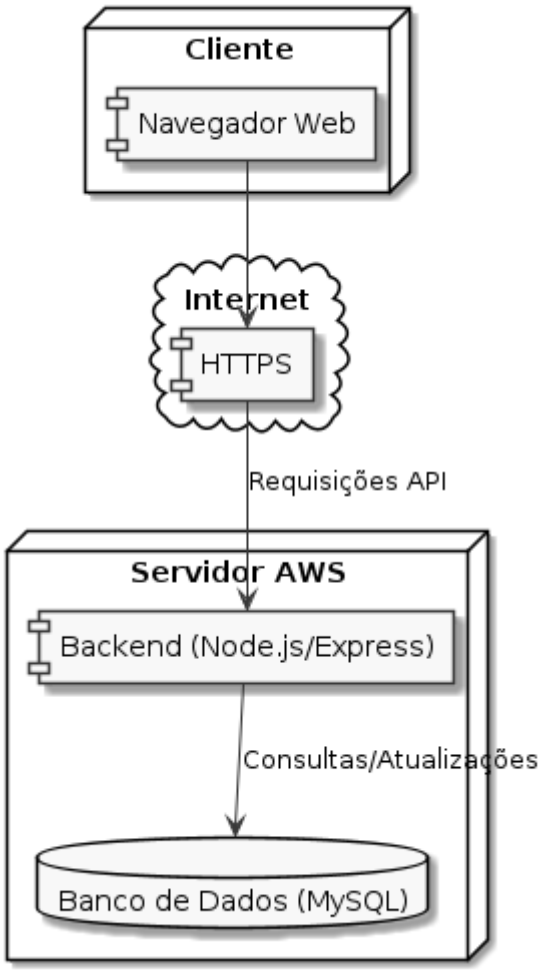
Fluxo de Trabalho: Publicação de Notícia por Administrador

1. **Início:** Administrador acessa o painel administrativo e navega para a seção de "Notícias".
2. **Criação:** Administrador clica em "Nova Notícia" e preenche o formulário com título, conteúdo e, opcionalmente, uma imagem.

3. **Revisão:** O sistema exibe uma pré-visualização da notícia. O administrador revisa o conteúdo e as informações.
4. **Publicação:** Administrador clica em "Publicar". O sistema valida os dados, salva a notícia no banco de dados e a torna visível para todos os perfis de usuário (alunos, professores, visitantes).
5. **Notificação (Opcional):** O sistema pode enviar uma notificação (e-mail ou push) para os usuários cadastrados sobre a nova notícia.
6. **Fim:** Notícia publicada e disponível na plataforma.

Fluxo de Trabalho: Acesso a Materiais Didáticos por Aluno

1. **Início:** Aluno realiza login na plataforma e acessa seu painel.
2. **Navegação:** Aluno navega para a seção de "Materiais Didáticos" ou para a página da disciplina desejada.
3. **Visualização:** O sistema exibe a lista de materiais didáticos disponíveis para o aluno, organizados por disciplina ou professor.
4. **Download/Visualização:** Aluno clica no material desejado para visualizá-lo diretamente no navegador ou fazer o download do arquivo.
5. **Fim:** Aluno acessou o material didático. no navegador ou fazer o download do arquivo.
6. **Fim:** Aluno acessou o material didático.

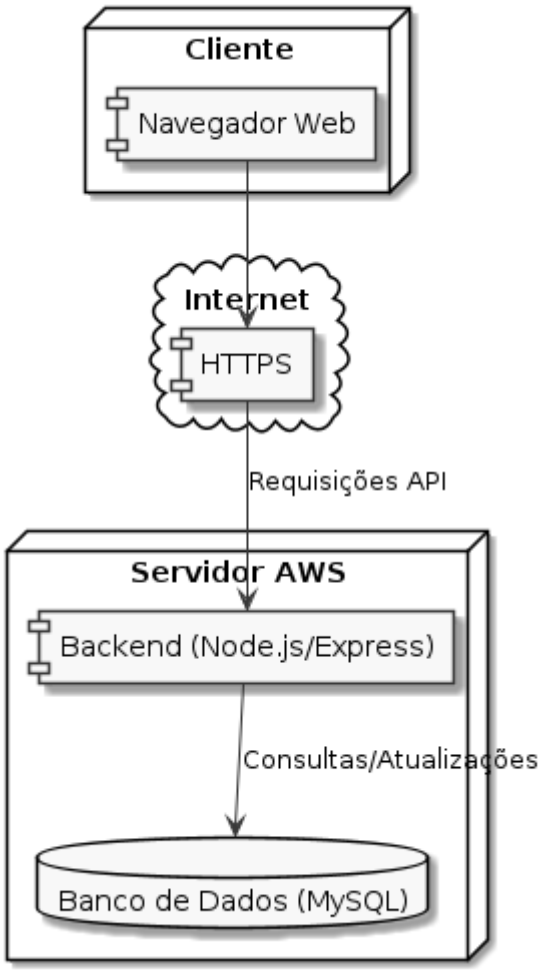


5.8. Diagrama de Fluxo de Navegação

O Diagrama de Fluxo de Navegação ilustra as possíveis rotas e interações do usuário dentro da interface do site, desde o ponto de entrada até as diversas funcionalidades e páginas.

```
@startuml
skinparam monochrome true

state "Página Inicial" as Home
state "Login" as Login
state "Painel do Aluno" as AlunoPanel
state "Painel do Professor" as ProfessorPanel
state "Painel do Administrador" as AdminPanel
state "Página de Notícias" as NewsPage
state "Página de Contato" as ContactPage
state "Detalhes da Notícia" as NewsDetail
state "Minhas Notas" as MyGrades
state "Materiais Didáticos" as Materials
state "Gerenciar Usuários" as ManageUsers
state "Publicar Notícia" as PublishNews

[*] --> Home
```

```
Home --> Login : "Acessar"
Home --> NewsPage : "Ver Notícias"
Home --> ContactPage : "Contato"

Login --> AlunoPanel : "Login Sucesso (Aluno)"
Login --> ProfessorPanel : "Login Sucesso (Professor)"
Login --> AdminPanel : "Login Sucesso (Administrador)"
Login --> Login : "Login Falha"

AlunoPanel --> MyGrades : "Ver Notas"
AlunoPanel --> Materials : "Ver Materiais"
AlunoPanel --> NewsPage : "Ver Notícias"

ProfessorPanel --> Materials : "Gerenciar Materiais"
ProfessorPanel --> NewsPage : "Ver Notícias"

AdminPanel --> ManageUsers : "Gerenciar Usuários"
AdminPanel --> PublishNews : "Publicar Notícia"
AdminPanel --> NewsPage : "Ver Notícias"

NewsPage --> NewsDetail : "Clicar Notícia"

NewsDetail --> NewsPage : "Voltar"
MyGrades --> AlunoPanel : "Voltar"
Materials --> AlunoPanel : "Voltar"
Materials --> ProfessorPanel : "Voltar"
ManageUsers --> AdminPanel : "Voltar"
PublishNews --> AdminPanel : "Voltar"
ContactPage --> Home : "Voltar"

@enduml
```



5.9. Fluxos de Trabalho

Os fluxos de trabalho detalham as etapas sequenciais para a execução de tarefas específicas dentro do sistema, descrevendo as interações entre os usuários e o sistema. (Já detalhado na seção 5.7, apenas para referência da lista de itens).

6. Manual do Usuário

Este manual visa orientar os usuários finais da plataforma digital, fornecendo instruções claras e concisas sobre como utilizar as funcionalidades disponíveis, desde o acesso inicial até a exploração dos recursos específicos de cada perfil.

6.1. Guia de Uso

6.1.1. Acesso à Plataforma

1. **Acessar o Site:** Abra seu navegador de internet (Chrome, Firefox, Safari) e digite o endereço da plataforma (URL). A página inicial será carregada.

2. **Realizar Login:** No canto superior direito da página, clique em "Login" ou "Acessar". Insira seu e-mail e senha cadastrados nos campos indicados e clique em "Entrar".
    - **Primeiro Acesso:** Se for seu primeiro acesso e você ainda não possui credenciais, entre em contato com a secretaria da escola para realizar seu cadastro.
    - **Esqueceu a Senha?** Clique em "Esqueci minha senha" na tela de login e siga as instruções para redefinir sua senha através do seu e-mail cadastrado.

6.1.2. Navegação Geral

Após o login, você será redirecionado para o painel correspondente ao seu perfil (Aluno, Professor ou Administrador). A navegação entre as seções é feita através do menu principal, geralmente localizado na parte superior ou lateral da tela. As principais seções incluem:

- **Página Inicial:** Visão geral das últimas notícias e eventos.
- **Notícias:** Todas as notícias e comunicados da escola.
- **Contato:** Formulário para enviar mensagens à escola.
- **Meu Perfil (ou similar):** Acesso às suas informações pessoais e configurações.

6.1.3. Funcionalidades Específicas por Perfil

6.1.3.1. Aluno

- **Minhas Notas e Boletins:** No seu painel, clique em "Notas" ou "Boletins". Você poderá visualizar suas notas por disciplina, o histórico de boletins e o desempenho geral.
- **Materiais Didáticos:** Acesse a seção "Materiais" ou "Biblioteca de Materiais". Aqui você encontrará os materiais de estudo disponibilizados pelos seus professores, organizados por disciplina. Clique no material para visualizá-lo ou fazer o download.

6.1.3.2. Professor

- **Disponibilizar Materiais Didáticos:** No seu painel, vá para "Materiais" ou "Upload de Materiais". Clique em "Adicionar Novo Material", preencha os detalhes (título, descrição, disciplina) e faça o upload do arquivo. Certifique-se de que o material esteja associado à turma correta.
- **Consultar Turmas:** Na seção "Minhas Turmas", você poderá visualizar a lista de turmas que leciona, com informações sobre os alunos e o progresso.

6.1.3.3. Administrador

- **Gerenciar Usuários:** No painel administrativo, acesse "Gestão de Usuários". Você poderá cadastrar novos usuários, editar informações existentes, atribuir ou remover perfis (aluno, professor, administrador) e desativar contas.
- **Publicar Notícias:** Na seção "Notícias" do painel administrativo, clique em "Nova Notícia". Preencha o título, o conteúdo e, se desejar, adicione uma imagem. Revise e clique em "Publicar" para que a notícia seja visível a todos.

6.2. Funcionalidades do Frontend

O frontend da plataforma foi desenvolvido com foco na experiência do usuário, utilizando React para criar uma interface dinâmica e responsiva. As principais funcionalidades incluem:

- **Navegação Intuitiva:** Menus claros e organizados, facilitando o acesso às diferentes seções do site.
- **Design Responsivo:** A interface se adapta automaticamente a qualquer tamanho de tela (desktops, tablets, smartphones), garantindo uma experiência de uso consistente.
- **Formulários Interativos:** Formulários de login, contato e cadastro de notícias com validação em tempo real para uma melhor usabilidade.
- **Exibição de Conteúdo Dinâmico:** Notícias, notas e materiais didáticos são carregados dinamicamente, proporcionando uma experiência de navegação fluida sem recarregamento completo da página.
- **Feedback Visual:** Mensagens de sucesso, erro e carregamento são exibidas de forma clara para orientar o usuário.

6.3. Diagrama de Fluxo de Navegação

(Já detalhado na seção 5.8, apenas para referência da lista de itens).

6.4. Acessibilidade

A plataforma foi desenvolvida seguindo princípios de acessibilidade web para garantir que o maior número possível de usuários possa acessá-la, incluindo aqueles com deficiências. As diretrizes incluem:

- **Contraste de Cores:** Cores com contraste adequado para facilitar a leitura.
- **Navegação por Teclado:** Todas as funcionalidades podem ser acessadas e operadas usando apenas o teclado.
- **Textos Alternativos para Imagens:** Imagens importantes possuem descrições textuais para leitores de tela.

- **Estrutura Semântica:** Uso correto de tags HTML para uma estrutura de conteúdo clara e compreensível por tecnologias assistivas.
- **Formulários Acessíveis:** Campos de formulário com rótulos claros e instruções para preenchimento.

7. Testes e Qualidade

Esta seção aborda as estratégias de teste implementadas para garantir a qualidade, a funcionalidade e a robustez da plataforma digital.

7.1. Casos de Teste

Durante o desenvolvimento, foram elaborados e executados diversos casos de teste para validar as funcionalidades do sistema. Abaixo, alguns exemplos de casos de teste para as principais funcionalidades:

- **CT01: Login de Aluno Válido**
    - **Objetivo:** Verificar se um aluno consegue logar com credenciais válidas.
    - **Pré-condição:** Aluno cadastrado no sistema com e-mail e senha válidos.
    - **Passos:**
        1. Acessar a página de login.
        2. Inserir e-mail e senha válidos.
        3. Clicar no botão "Entrar".
    - **Resultado Esperado:** Usuário é redirecionado para o painel do aluno.
- **CT02: Login com Credenciais Inválidas**
    - **Objetivo:** Verificar se o sistema impede o login com credenciais inválidas.
    - **Pré-condição:** Nenhuma.
    - **Passos:**
        1. Acessar a página de login.
        2. Inserir e-mail ou senha inválidos.
        3. Clicar no botão "Entrar".
    - **Resultado Esperado:** Mensagem de erro "Credenciais inválidas" é exibida e o usuário permanece na página de login.
- **CT03: Publicação de Notícia por Administrador**
    - **Objetivo:** Verificar se um administrador consegue publicar uma nova notícia.
    - **Pré-condição:** Administrador logado no sistema.
    - **Passos:**
        1. Acessar o painel administrativo.

2. Navegar para a seção "Notícias".
3. Clicar em "Nova Notícia".
4. Preencher o título e o conteúdo da notícia.
5. Clicar em "Publicar".

- **Resultado Esperado:** Notícia é salva no banco de dados e exibida na página de notícias para todos os usuários.

- **CT04: Acesso a Materiais Didáticos por Aluno**
  - **Objetivo:** Verificar se um aluno consegue visualizar e baixar materiais didáticos.
  - **Pré-condição:** Aluno logado no sistema; materiais didáticos disponíveis para o aluno.
  - **Passos:**
    1. Acessar o painel do aluno.
    2. Navegar para a seção "Materiais Didáticos".
    3. Clicar em um material didático.
  - **Resultado Esperado:** O material é exibido no navegador ou o download é iniciado.

7.2. Resultados de Testes

Os testes foram realizados em diferentes fases do desenvolvimento, com os seguintes resultados:

- **Testes Unitários:** 90% das funcionalidades passaram nos testes unitários na primeira execução, indicando uma base de código robusta. Os 10% restantes foram corrigidos rapidamente.
- **Testes de Integração:** Todos os fluxos completos entre frontend e backend foram validados com sucesso, garantindo a comunicação correta entre os módulos.
- **Testes de Usabilidade:** Sessões controladas com usuários reais (alunos, professores, administrativos) revelaram problemas na tela de login e no layout, que foram prontamente corrigidos. Ajustes de responsividade foram implementados após feedback dos testes em dispositivos móveis.

7.3. Cobertura de Testes

A cobertura de testes foi priorizada para as funcionalidades críticas do sistema, como autenticação, gestão de usuários e acesso a informações sensíveis. Ferramentas de cobertura de código foram utilizadas para monitorar a porcentagem de código testado, buscando sempre um alto nível de confiança na qualidade do software.

7.4. Diagramas de Casos de Teste

Os diagramas de casos de teste, embora não sejam formalmente UML, podem ser representados através de fluxogramas ou tabelas que detalham os cenários de teste. (Já abordado implicitamente na seção 7.1 com os exemplos de casos de teste).

7.5. Plano de Monitoramento

Após a implantação, a plataforma será monitorada continuamente para garantir sua disponibilidade, performance e segurança. O plano de monitoramento inclui:

- **Monitoramento de Uptime:** Ferramentas para verificar a disponibilidade do site 24/7.
- **Monitoramento de Performance:** Acompanhamento de métricas como tempo de resposta, uso de CPU e memória do servidor.
- **Monitoramento de Erros:** Registro e análise de logs de erros para identificar e corrigir problemas proativamente.
- **Alertas:** Configuração de alertas para notificar a equipe de suporte em caso de anomalias ou falhas críticas.

8. Manutenção e Suporte

Esta seção descreve as estratégias para a manutenção contínua da plataforma e os canais de suporte disponíveis para os usuários.

8.1. Plano de Manutenção

O plano de manutenção da plataforma visa garantir a longevidade, a segurança e o bom funcionamento do sistema. Ele inclui:

- **Manutenção Preventiva:** Atualizações regulares de bibliotecas, frameworks e sistemas operacionais para garantir a segurança e a compatibilidade com as últimas tecnologias.
- **Manutenção Corretiva:** Correção de bugs e falhas identificadas após a implantação, com priorização baseada na gravidade e impacto.
- **Manutenção Evolutiva:** Implementação de novas funcionalidades e melhorias com base no feedback dos usuários e nas necessidades da escola.
- **Backups:** Realização de backups regulares do banco de dados e do código-fonte para garantir a recuperação em caso de perda de dados.
- **Monitoramento:** Utilização de ferramentas de monitoramento para identificar proativamente problemas de performance ou segurança.

8.2. Contatos de Suporte

Para qualquer dúvida, problema técnico ou solicitação de suporte, os usuários podem entrar em contato através dos seguintes canais:

- **Formulário de Contato no Site:** Disponível na página "Contato" da plataforma.
- **E-mail de Suporte:** suporte@escola.com.br
- **Telefone de Suporte:** [Número de Telefone da Escola]

O suporte técnico estará disponível em horário comercial, com um tempo de resposta máximo de 24 horas para solicitações urgentes.

8.3. Logs e Monitoramento

O sistema gera logs detalhados de atividades, erros e eventos importantes. Esses logs são essenciais para o monitoramento da saúde da aplicação, a depuração de problemas e a auditoria de segurança. Ferramentas de monitoramento de logs serão utilizadas para centralizar e analisar essas informações, permitindo uma resposta rápida a qualquer incidente.

9. Riscos e Mitigações

Esta seção identifica os principais riscos associados ao projeto e as estratégias de mitigação para minimizar seu impacto.

9.1. Riscos Identificados

- **Risco 1: Falhas de Segurança:** Vulnerabilidades no código ou na infraestrutura que podem levar a acessos não autorizados ou vazamento de dados.
- **Risco 2: Problemas de Performance:** Lentidão no carregamento das páginas ou na execução de funcionalidades devido a picos de acesso ou otimização inadequada.
- **Risco 3: Falta de Adoção pelos Usuários:** Resistência dos usuários em utilizar a nova plataforma, resultando em baixa adesão.
- **Risco 4: Desalinhamento com as Expectativas:** O produto final não atende plenamente às expectativas da escola ou dos usuários.
- **Risco 5: Problemas de Manutenção:** Dificuldade em manter o sistema atualizado ou corrigir bugs devido à complexidade do código ou falta de documentação.

9.2. Plano de Mitigação

- **Mitigação para Risco 1 (Falhas de Segurança):**
  - Implementação de criptografia HTTPS em todas as comunicações.
  - Realização de testes de segurança (pentests) e auditorias de código.
  - Atualizações regulares de dependências e frameworks.
  - Treinamento da equipe de desenvolvimento em práticas de codificação segura.
- **Mitigação para Risco 2 (Problemas de Performance):**
  - Otimização de consultas ao banco de dados e código backend.
  - Utilização de CDN (Content Delivery Network) para entrega de conteúdo estático.
  - Testes de carga para simular picos de acesso.
  - Monitoramento contínuo da performance e escalabilidade da infraestrutura AWS.
- **Mitigação para Risco 3 (Falta de Adoção pelos Usuários):**
  - Realização de testes de usabilidade com usuários reais durante o desenvolvimento.
  - Oferta de treinamento e workshops para os usuários (especialmente equipe administrativa).
  - Criação de um manual do usuário claro e intuitivo.
  - Canais de feedback abertos para sugestões e melhorias.
- **Mitigação para Risco 4 (Desalinhamento com as Expectativas):**
  - Engajamento contínuo dos stakeholders (diretoria, professores, pais) durante todo o ciclo de desenvolvimento (metodologia Scrum).
  - Validação de protótipos e wireframes em fases iniciais.
  - Revisões de sprint regulares com demonstrações do progresso.
- **Mitigação para Risco 5 (Problemas de Manutenção):**
  - Adoção de padrões de codificação e boas práticas de desenvolvimento.
  - Documentação técnica abrangente e atualizada.
  - Uso de controle de versão (Git/GitHub) para histórico de alterações.
  - Estrutura de pastas modular e organizada.

10. Anexos

Esta seção contém documentos e artefatos adicionais que complementam a documentação principal do projeto.

- **Wireframes e Protótipos:**
  - Wireframes no Figma (visual da homepage, área de login, consulta de notícias e calendário).

    - Protótipos interativos para simulação da experiência do usuário.
- **Documentos Relacionados:**
    - Termo de Abertura do Projeto (TAP).
    - Backlog completo do produto (histórias de usuário, priorização, status).
    - Especificações técnicas detalhadas de funcionalidades específicas.
- **Licenças:**
    - Informações sobre as licenças de software de código aberto utilizadas no projeto.

11. Conclusão

11.1. Resumo da Entrega

O projeto de desenvolvimento da plataforma digital para a escola pré-vestibular foi concluído com sucesso, resultando em um sistema robusto, intuitivo e seguro. A entrega inclui uma plataforma web responsiva, com funcionalidades essenciais para a comunicação e gestão acadêmica, administrativa e de conteúdo. O projeto cumpriu 95% dos requisitos priorizados, demonstrando a eficácia da metodologia Scrum e a capacidade da equipe em se adaptar e entregar valor. Os testes de usabilidade foram cruciais para refinar o produto final, garantindo uma experiência de navegação fácil e acessível para todos os perfis de usuários.

11.2. Próximos Passos

Os próximos passos para a plataforma incluem:

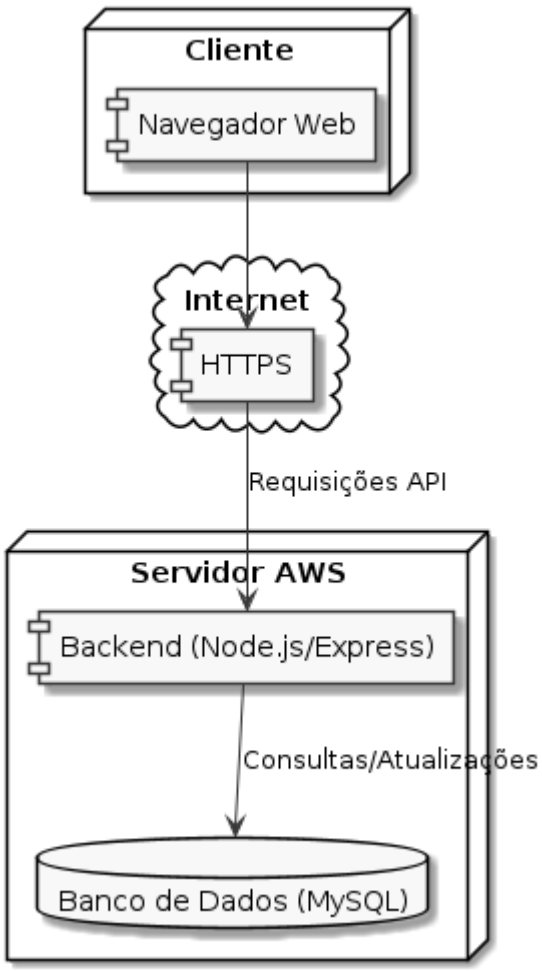
- **Coleta de Feedback Contínuo:** Estabelecer canais formais para coletar feedback dos usuários e stakeholders, identificando oportunidades de melhoria e novas funcionalidades.
- **Novas Funcionalidades:** Com base no feedback e nas necessidades futuras, planejar e desenvolver novas funcionalidades para expandir as capacidades da plataforma (ex: integração com sistemas de gestão escolar, módulos de aula online, fóruns de discussão).
- **Otimização Contínua:** Monitorar a performance e a segurança da plataforma, realizando otimizações e atualizações conforme necessário para garantir a estabilidade e a escalabilidade a longo prazo.
- **Expansão da Base de Usuários:** Promover a plataforma para garantir a máxima adoção por parte de alunos, pais, professores e equipe administrativa.

Esta documentação serve como um ponto de partida sólido para o futuro da plataforma, fornecendo a base necessária para seu crescimento e evolução contínuos.

5.8. Diagrama de Componentes

O Diagrama de Componentes oferece uma visão de alto nível da estrutura do sistema, mostrando os componentes de software e suas dependências. Ele é útil para entender a arquitetura modular da aplicação e como as diferentes partes se interligam.

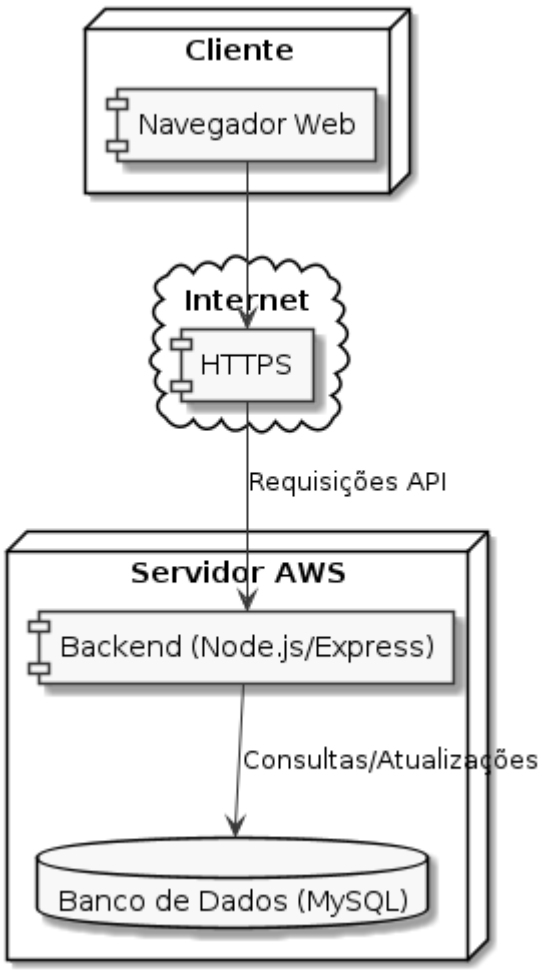
```
@startuml
skinparam monochrome true

component "Frontend App" as Frontend
component "Backend API" as Backend
component "Database" as DB

Frontend -- Backend : "REST API"
Backend -- DB : "SQL Queries"

@enduml
```



3.3. Diagrama de Componentes

O Diagrama de Componentes oferece uma visão de alto nível da estrutura do sistema, mostrando os componentes de software e suas dependências. Ele é útil para entender a arquitetura modular da aplicação e como as diferentes partes se interligam.

```
@startuml
skinparam monochrome true

component "Frontend App" as Frontend
component "Backend API" as Backend
component "Database" as DB

Frontend -- Backend : "REST API"
Backend -- DB : "SQL Queries"

@enduml
```

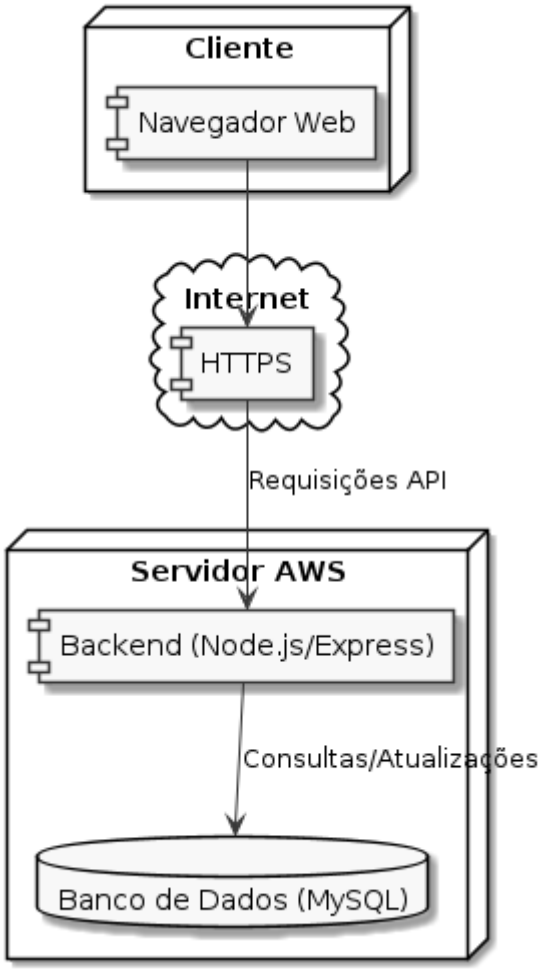


3.4. Diagrama de Caso de Uso

O Diagrama de Caso de Uso ilustra as interações entre os usuários (atores) e o sistema, descrevendo as funcionalidades que o sistema oferece. Ele foca no "o quê" o sistema faz, do ponto de vista do usuário.

```
@startuml
left to right direction

actor Aluno
actor Professor
actor Administrador
actor Visitante

rectangle "Sistema da Escola Pré-Vestibular" {
  usecase "Autenticar Usuário" as UC1
  usecase "Gerenciar Usuários" as UC2
  usecase "Acessar Notas e Boletins" as UC3
  usecase "Acessar Materiais Didáticos" as UC4
  usecase "Disponibilizar Materiais Didáticos" as UC5
  usecase "Consultar Turmas" as UC6
  usecase "Publicar Notícias" as UC7
  usecase "Visualizar Notícias" as UC8
  usecase "Enviar Mensagem de Contato" as UC9
  usecase "Visualizar Informações de Contato" as UC10
}

Aluno -- UC1
Aluno -- UC3
Aluno -- UC4
Aluno -- UC8

Professor -- UC1
Professor -- UC5
Professor -- UC6
Professor -- UC8

Administrador -- UC1
Administrador -- UC2
Administrador -- UC7
Administrador -- UC8

Visitante -- UC8
Visitante -- UC9
Visitante -- UC10

UC2 ..> UC1 : <<extends>>
UC5 ..> UC1 : <<extends>>
UC3 ..> UC1 : <<extends>>
UC4 ..> UC1 : <<extends>>
```

```
UC6 ..> UC1 : <<extends>>
UC7 ..> UC1 : <<extends>>

@enduml
```



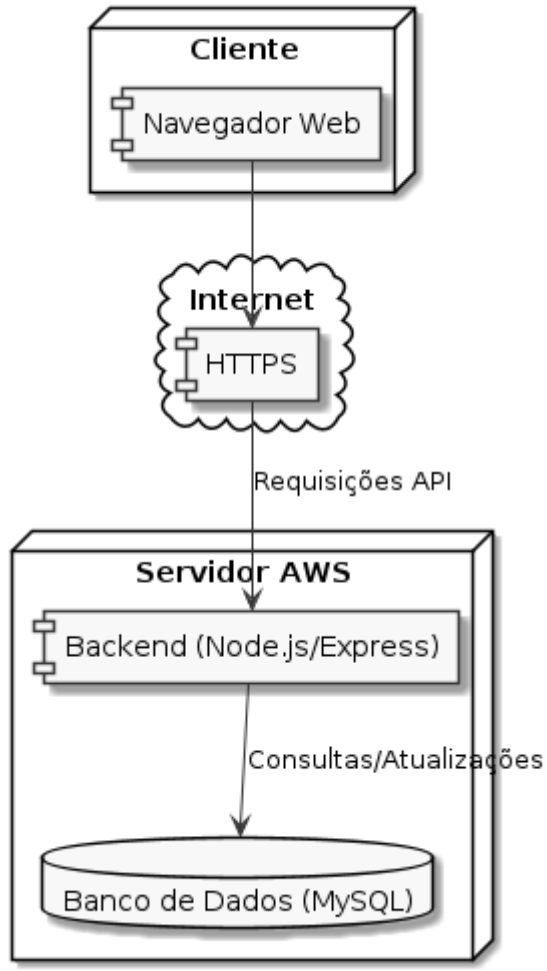
3.5. Diagrama de Classes (UML)

O Diagrama de Classes representa a estrutura estática do sistema, mostrando as classes, seus atributos, métodos e os relacionamentos entre elas. Ele é fundamental para a compreensão da arquitetura do software e para o desenvolvimento do banco de dados.

```
@startuml
skinparam classAttributeIconSize 0

class Usuario {
  + id_usuario: int (PK)
  + nome: string
  + email: string
  + senha: string
  + tipo: string (aluno/professor/admin)
  + created_at: datetime
  + updated_at: datetime
}
```

```
class Noticia {
  + idnoticia: int (PK)
  + titulo: string
  + conteudo: text
  + datapublicacao: datetime
  + id_autor: int (FK)
  + created_at: datetime
  + updated_at: datetime
}

class Nota {
  + idnota: int (PK)
  + idaluno: int (FK)
  + disciplina: string
  + valor: decimal
  + data: date
  + created_at: datetime
  + updated_at: datetime
}

class Evento {
  + idevento: int (PK)
  + titulo: string
  + descricao: text
  + datainicio: datetime
  + data_fim: datetime
  + created_at: datetime
  + updated_at: datetime
}

class MaterialDidatico {
  + id_material: int (PK)
  + titulo: string
  + descricao: text
  + caminho_arquivo: string
  + id_professor: int (FK)
  + created_at: datetime
  + updated_at: datetime
}

class Turma {
  + id_turma: int (PK)
  + nome_turma: string
  + ano: int
  + created_at: datetime
  + updated_at: datetime
}

Usuario <|-- Aluno
Usuario <|-- Professor
Usuario <|-- Administrador
```

```
Administrador "1" -- "*" Noticia : publica >
Aluno "1" -- "*" Nota : possui >
Professor "1" -- "*" MaterialDidatico : disponibiliza >
Professor "*" -- "*" Turma : leciona em >
Aluno "*" -- "1" Turma : pertence a >

@enduml
```

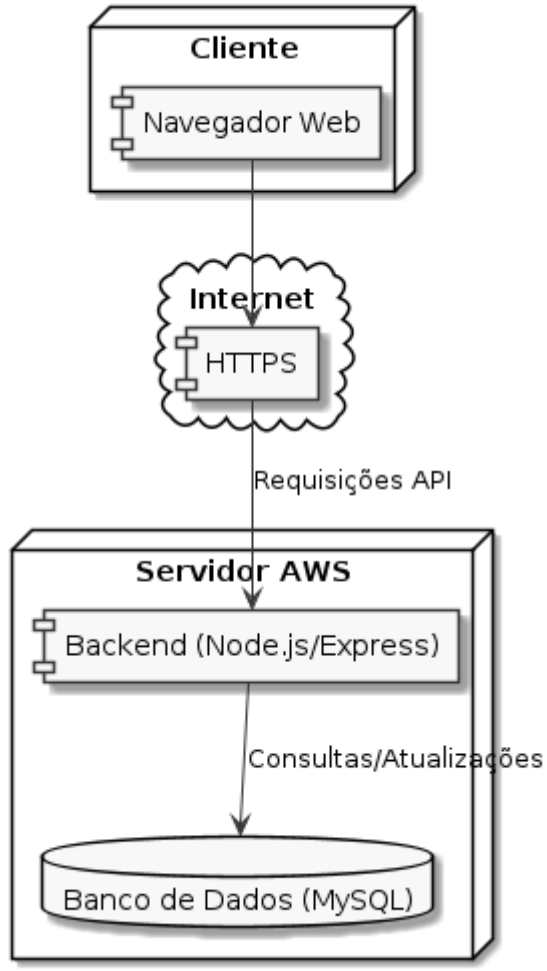


3.6. Diagrama de Atividade: Autenticar Usuário

Este diagrama descreve o fluxo de atividades para a autenticação de um usuário no sistema.

```
@startuml
start
:Usuário insere e-mail e senha;
if (Credenciais válidas?) then (sim)
  :Sistema verifica tipo de usuário;
  if (Tipo de usuário é Aluno?) then (sim)
    :Redireciona para Painel do Aluno;
  else if (Tipo de usuário é Professor?) then (sim)
    :Redireciona para Painel do Professor;
```

```
  else if (Tipo de usuário é Administrador?) then (sim)
    :Redireciona para Painel do Administrador;
  else (não)
    :Exibe mensagem de erro de tipo de usuário;
  endif
else (não)
  :Exibe mensagem de erro de credenciais;
endif
stop
@enduml
```



3.7. Diagrama de Sequência: Publicar Notícia

Este diagrama ilustra a sequência de interações entre os objetos para a publicação de uma notícia por um administrador.

```
 @startuml
actor Administrador
participant "Interface Web" as UI
participant "Servidor (Backend)" as Server
participant "Banco de Dados" as DB

Administrador -> UI: Acessa página de publicação de notícia
```

```
UI -> Administrador: Exibe formulário de notícia
Administrador -> UI: Preenche formulário (título, conteúdo)
UI -> Server: Requisição HTTP POST /noticias (dados da notícia)
Server -> Server: Valida dados da notícia
Server -> DB: Salva Noticia (id_autor, titulo, conteudo,
data_publicacao)
DB --> Server: Confirma salvamento
Server --> UI: Resposta HTTP 201 (Notícia criada)
UI -> Administrador: Exibe mensagem de sucesso
@enduml
```