**DERCIO ANLI ANSELMO-706230196**

**DOCUMENTACAO DO PROJECTO REVISTA ONLINE KWENU**

# Documentação da Aplicação Front-end: Revista Online KWENU

## INTRODUÇÃO

Esta documentação descreve a aplicação front-end da **Revista Online**, uma plataforma web para publicação e leitura de artigos, com funcionalidades de gerenciamento administrativo, chat em tempo real, e autenticação de usuários. O objetivo é fornecer uma visão completa das funcionalidades, estrutura, tecnologias utilizadas, e instruções para instalação, uso, e manutenção.

## INFORMAÇÕES GERAIS

- **Nome da aplicação**: Revista Online KWENU
- **Objetivo principal**: Permitir a leitura de artigos organizados por categorias, com funcionalidades de administração (gerenciamento de artigos, categorias, e chat) e interação via chat em tempo real.
- **Público-alvo**: Leitores interessados em artigos, administradores responsáveis por gerenciar conteúdo, e usuários que desejam interagir via chat.
- **Tecnologias utilizadas**:
    - HTML5
    - CSS3 (com Bootstrap 5.3.3 para estilização)
    - JavaScript (ES6+)
    - AJAX (via API fetch) para integração com backend
    - WebSocket para chat em tempo real
    - Backend: Node.js, Express, Multer, WebSocket (ws)

## 3. INSTALAÇÃO E CONFIGURAÇÃO

### Pré-requisitos

- Node.js (v18 ou superior)
- npm (gerenciador de pacotes do Node.js)
- Navegador moderno (Chrome, Firefox, Edge)
- Editor de código (VS Code)

### Como Iniciar o Projeto

1. Iniciando o servidor:
2. npm start

   O servidor será executado em http://localhost:3000.

3. Acesse a aplicação no navegador:
   - URL inicial: http://localhost:3000/index.html

# 4. Estrutura do Projeto

A estrutura do projeto é organizada para separar frontend, backend, e dados:

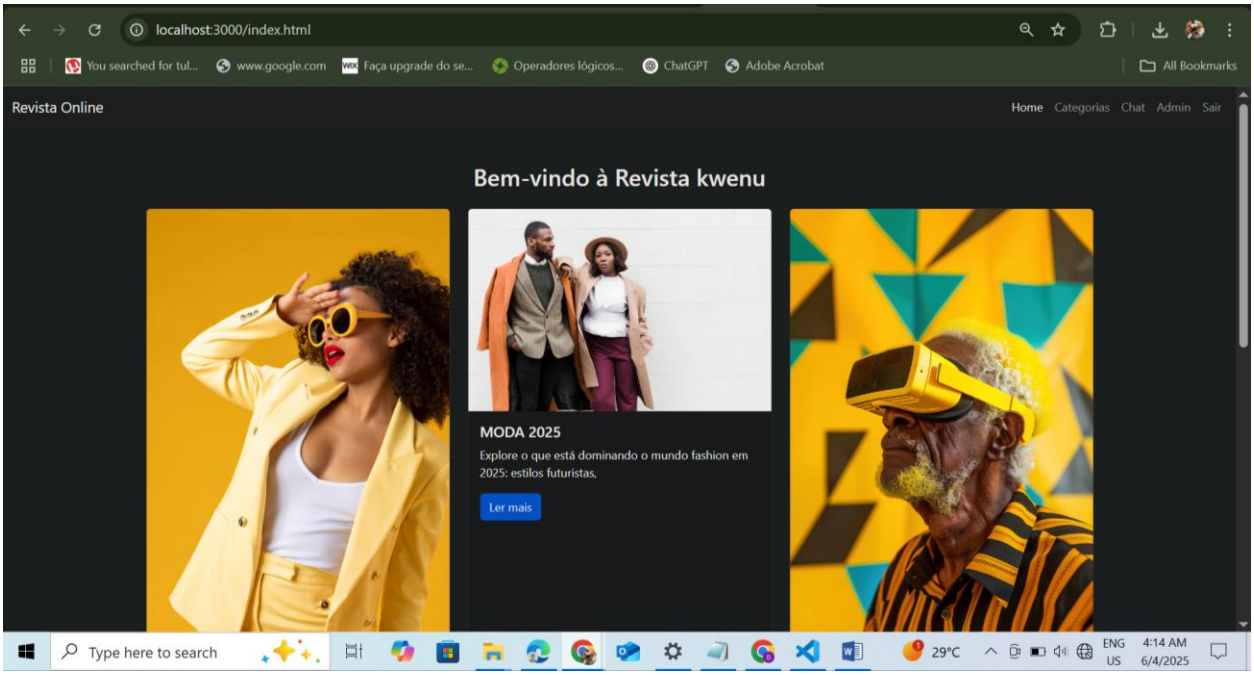
```
/revista-online
├── /public
│   ├── /css
│   │   └── styles.css (estilos personalizados)
│   ├── /uploads
│   │   └── (imagens enviadas via admin.html)
│   ├── admin.html (painel de administração)
│   ├── artigo.html (página de visualização de artigo)
│   ├── categorias.html (listagem de artigos por categoria)
│   ├── index.html (página inicial)
│   ├── login.html (página de login)
│   ├── register.html (página de cadastro)
```

│ └── chat.html (página de chat)
├── /server
│ ├── /routes
│ │ └── users.js (rotas de autenticação)
│ ├── app.js (servidor Node.js)
│ └── data.json (banco de dados simulado)
├── package.json (dependências e scripts)
└── README.md (instruções básicas)

# 5. Funcionalidades

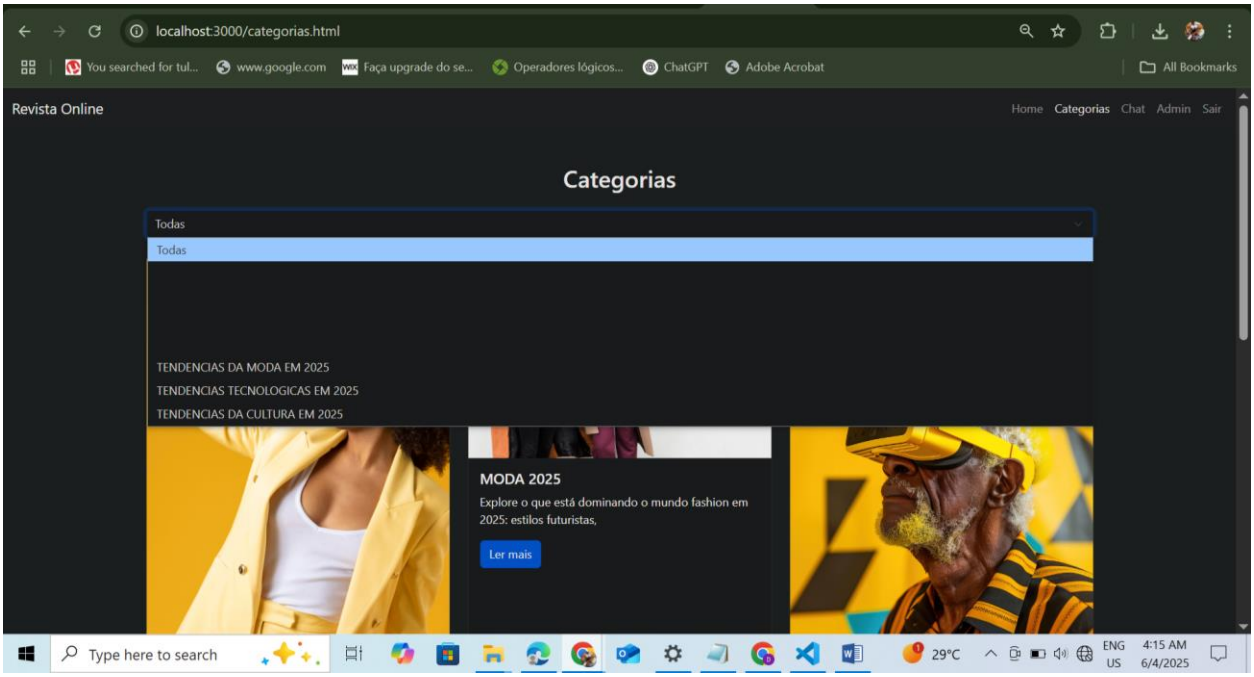
## Página Inicial (index.html)

- Exibe uma lista de artigos carregados via AJAX (fetch /api/articles).
- Inclui imagens dos artigos (de public/uploads/ ou public/imgs/) com fallback para placeholder.jpg.
- Navbar dinâmica: ajusta links com base no estado do usuário (logado/não logado/admin).




## Página de Categorias (categorias.html)

- Lista artigos filtrados por categoria, selecionada via dropdown.
- Carrega categorias e artigos via AJAX (fetch /api/categories, fetch /api/articles).
- Exibe imagens com fallback, similar à página inicial.



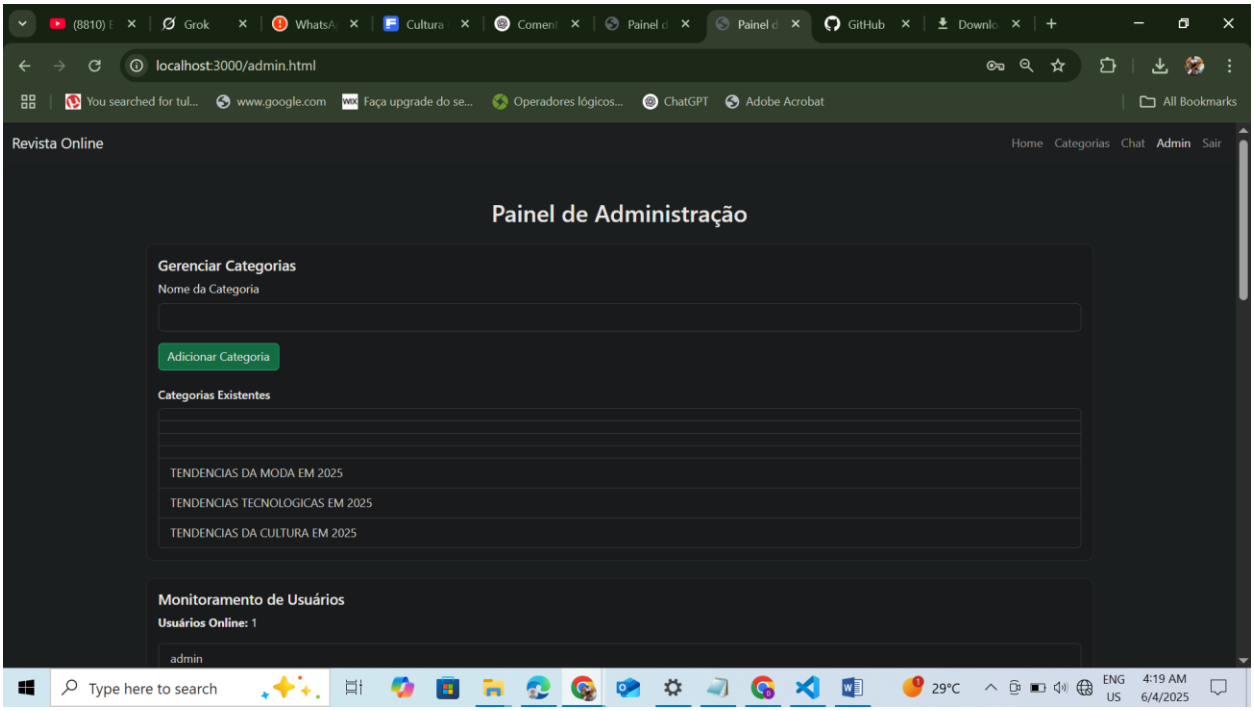

## Página de Artigo (artigo.html)

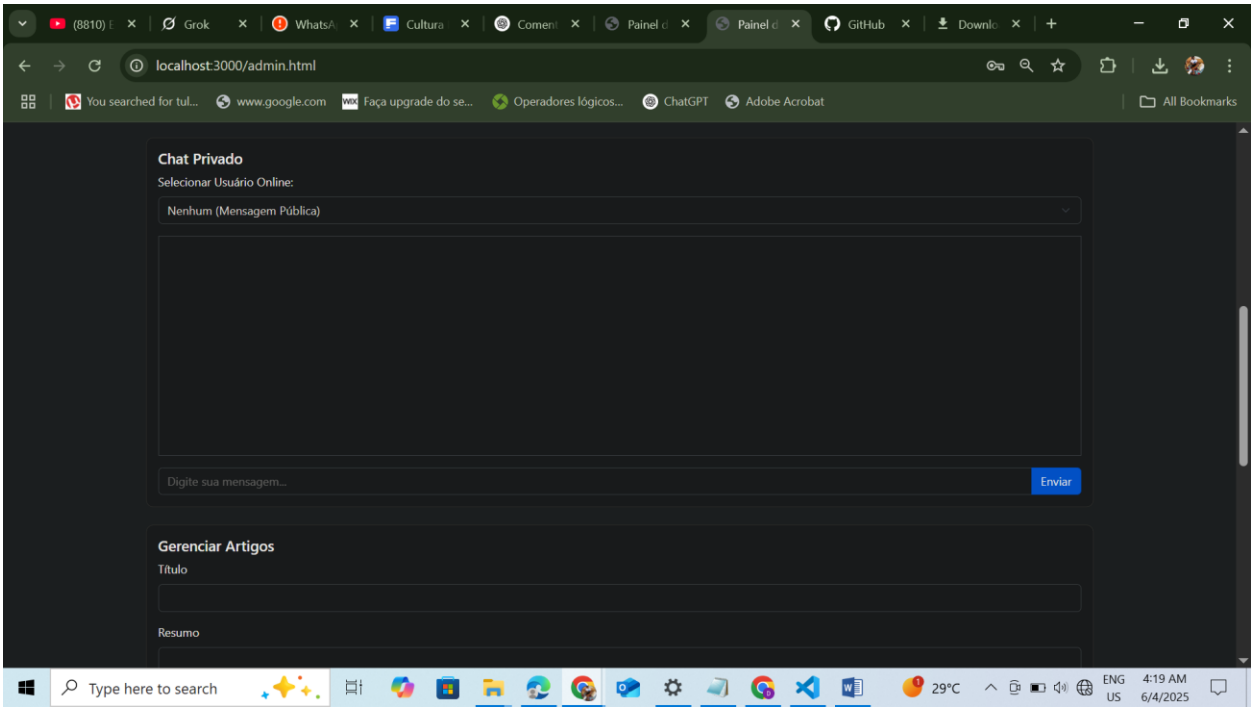
- Exibe o conteúdo completo de um artigo (com base no parâmetro id na URL, ex.: artigo.html?id=1).
- Carrega dados via AJAX (fetch /api/articles/:id).
- Mostra imagem do artigo com fallback.

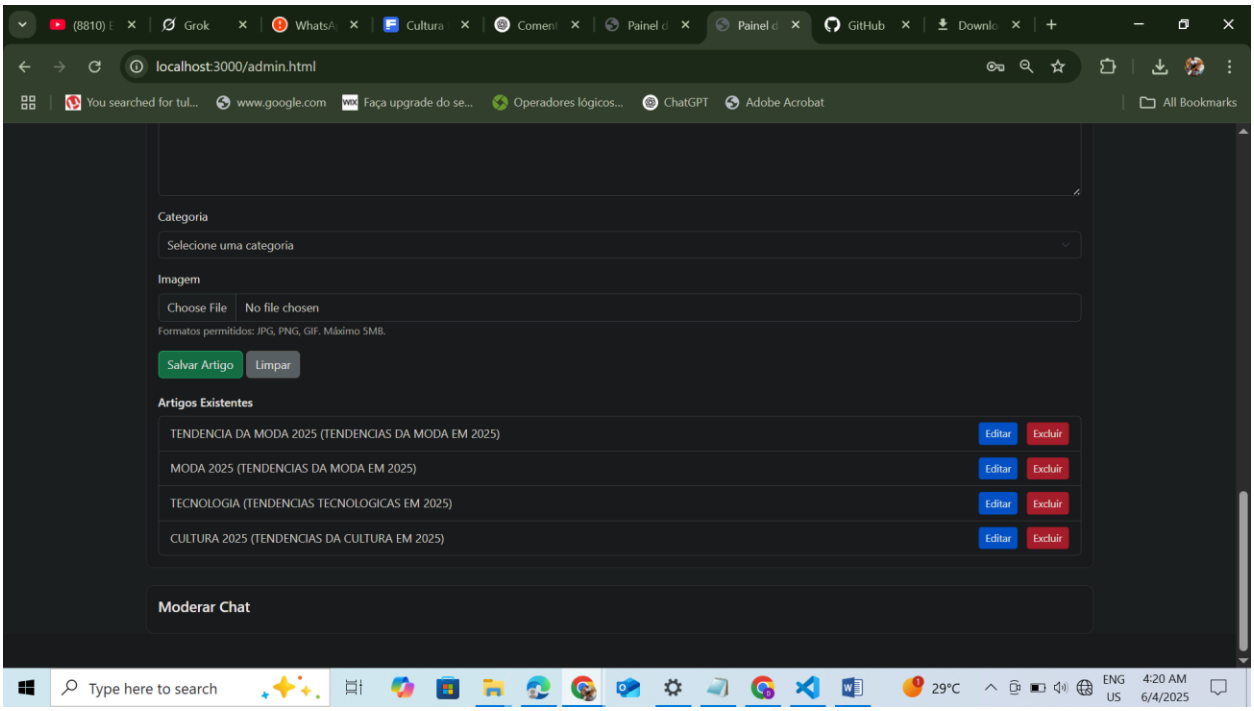
## Página de Administração (admin.html)

- **Restrita a administradores** (verifica user.role === 'admin').
- **Gerenciamento de categorias**: Adiciona e lista categorias via AJAX (POST /api/categories, GET /api/categories).
- **Gerenciamento de artigos**: Cria, edita, e exclui artigos com upload de imagens (POST /api/upload, POST/PUT/DELETE /api/articles).
- **Monitoramento de usuários**: Exibe usuários online via WebSocket.

- **Chat privado/público**: Envia/recebe mensagens em tempo real com WebSocket.
- **Moderação de chat**: Lista e exclui mensagens via AJAX (GET /api/messages, DELETE /api/messages/:id).

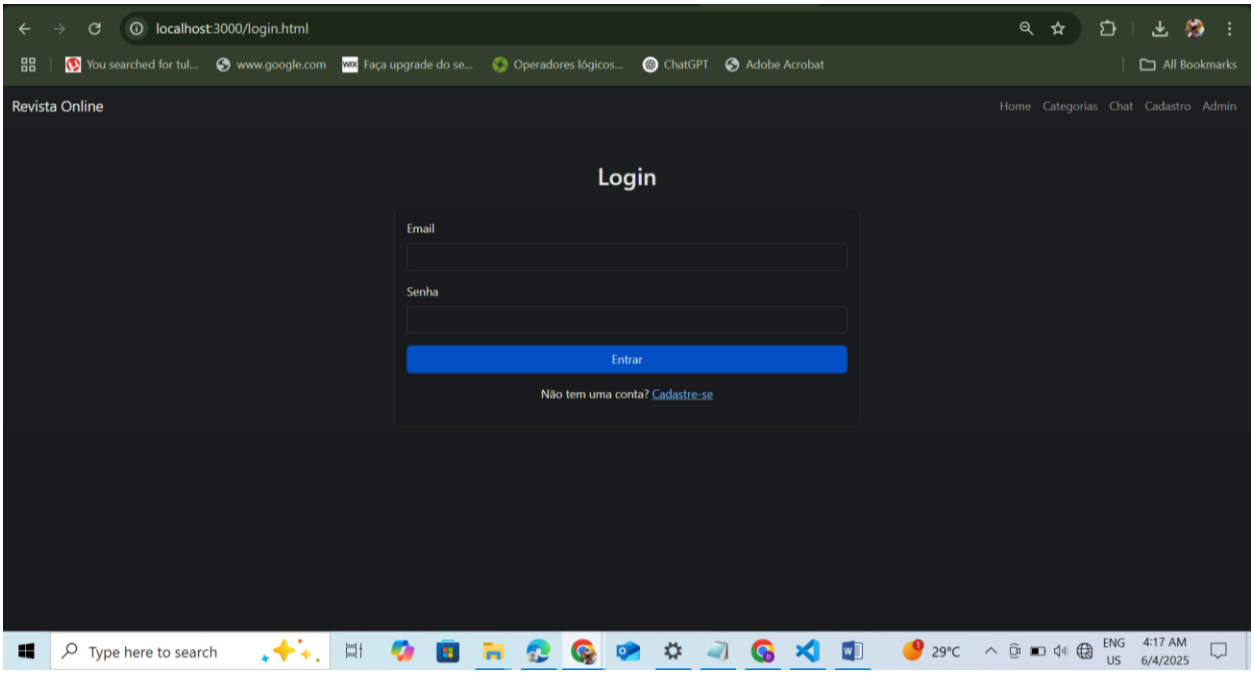




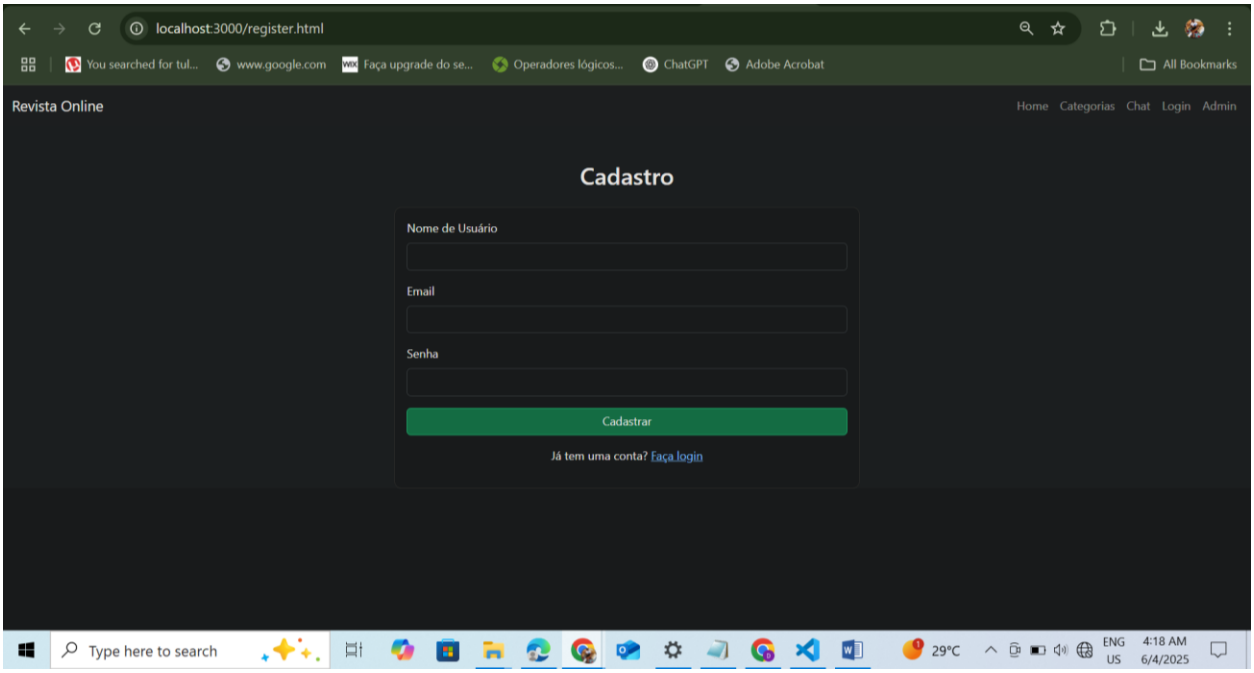

## Página de Login (login.html)

- Autentica usuários comparando credenciais com data.json via AJAX (POST /api/login).
- Salva usuário autenticado em localStorage.



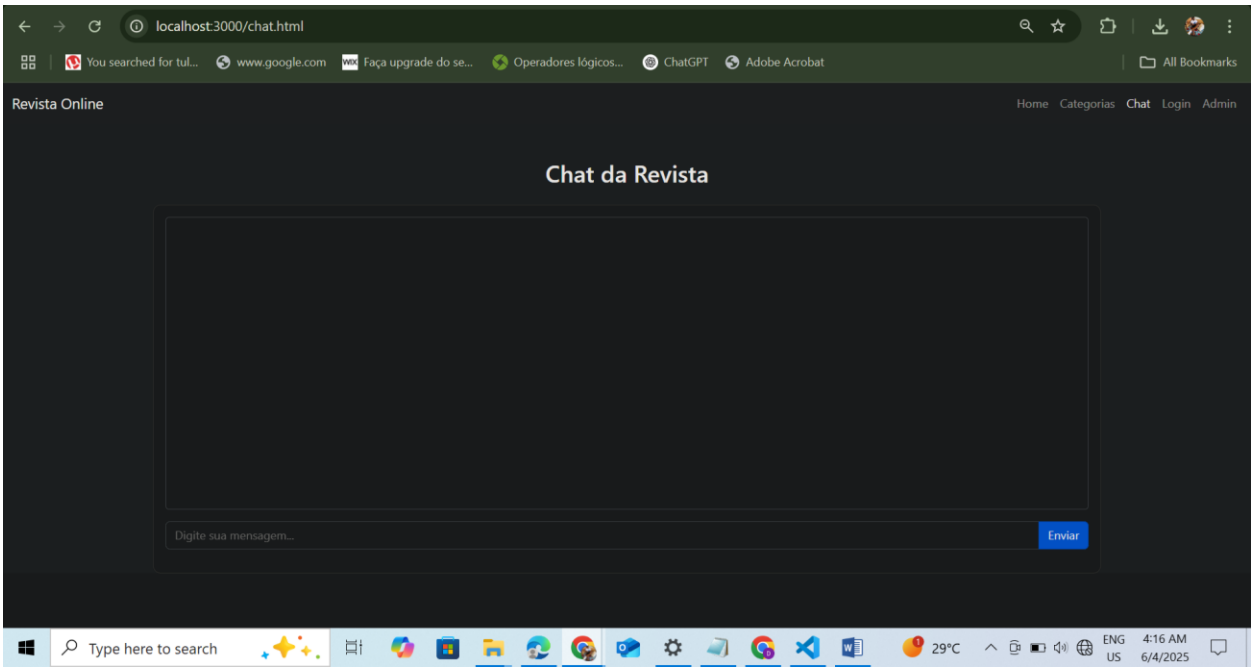
## Página de Cadastro (register.html)

- Registra novos usuários via AJAX (POST /api/register).
- Adiciona usuário ao data.json.




## Página de Chat (chat.html)

- Permite envio/recebimento de mensagens públicas ou privadas via WebSocket.
- Exibe histórico de mensagens e usuários online.




### Responsividade

- Usa Bootstrap 5.3.3 para layout responsivo, adaptado a desktops, tablets, e smartphones.

### Integração com APIs

- **AJAX**: Usado para carregar/salvar artigos, categorias, e mensagens.
- **WebSocket**: Implementa chat em tempo real e monitoramento de usuários online.
- **Upload de imagens**: Integra com multer para salvar imagens em public/uploads/.

## 6. Documentação de Código

- **Comentários**: O código JavaScript inclui comentários explicando funções principais (ex.: renderNavbar, loadArticles).
- **Boas práticas**:
  - Uso de async/await para chamadas assíncronas.
  - Tratamento de erros em requisições fetch e WebSocket.
  - Código modularizado com funções reutilizáveis.

- **Ferramentas**: Não usa JSDoc, mas os comentários são claros e suficientes para manutenção.
- **Nomenclatura**: Variáveis e funções seguem convenção camelCase (ex.: `loadCategories`, `articleTitle`).

# 7. Testes

- **Tipos de testes**: Não foram implementados testes automatizados (unitários, integração, ou end-to-end).
- **Testes manuais**:
  - Testado em navegadores Chrome e Firefox.
  - Verificado o carregamento de artigos/categorias via AJAX.
  - Testado upload de imagens (JPG/PNG/GIF, <5MB).
  - Validado o chat WebSocket (mensagens públicas/privadas).
  - Confirmada a restrição de acesso em `admin.html`.
- **Como executar testes manuais**:
  1. Inicie o servidor (`npm start`).
  2. Acesse `http://localhost:3000` e teste cada página.
  3. Use credenciais do `data.json` (ex.: `admin@revista.com`/`admin123`).
- **Sugestão futura**: Implementar Jest para testes unitários e Cypress para testes end-to-end.

# 8. Hospedagem

- **Status atual**: A aplicação ja foi hospedada.
- **Plataforma**: Render (suporta Node.js e estáticos).
- **Passos para deploy no Render**:
  1. Crie um repositório no GitHub com o projeto.
  2. Crie uma conta no Render (https://render.com).
  3. Configure um novo **Web Service**:
     - Conecte ao repositório GitHub.
     - Defina o comando de build: `npm install`.

        - Defina o comando de start: npm start.
    4. O app.js usa process.env.PORT || 3000.
- **Configuração de domínio**: Não aplicável no momento.
- **Link de acesso**: A ser definido após deploy.

# 9. Conclusão

A **Revista Online** é uma aplicação funcional para publicação de artigos, com administração robusta e interação em tempo real via chat. O uso de HTML, JavaScript, e Bootstrap garantiu uma interface simples e responsiva, enquanto o backend Node.js com WebSocket e AJAX proporcionou dinamismo.

**Lições aprendidas**:

- A importância de tratamento de erros em WebSocket para evitar travamentos (ex.: SyntaxError: "undefined" is not valid JSON).
- Benefícios do Bootstrap para layouts responsivos rápidos.
- Desafios de gerenciar estado de usuário com localStorage.

**Sugestões para melhorias futuras**:

- Adicionar testes automatizados (Jest, Cypress).
- Implementar comentários em artigo.html.
- Adicionar busca de artigos em index.html e categorias.html.
- Usar bcrypt para criptografar senhas no data.json.