INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA CAMPUS ARIQUEMES
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

BRUNO IRINEU SILVA DO NASCIMENTO

ODONTO OPEN: DESENVOLVIMENTO DE SOFTWARE OPEN SOURCE PARA AUTOMATIZAÇÃO DE CLÍNICAS ODONTOLÓGICAS

Ariquemes
2025

BRUNO IRINEU SILVA DO NASCIMENTO

ODONTO OPEN: DESENVOLVIMENTO DE SOFTWARE OPEN SOURCE PARA AUTOMATIZAÇÃO DE CLÍNICAS ODONTOLÓGICAS

Relatório técnico entregue como Trabalho de Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – (IFRO), Campus Ariquemes, como requisito parcial para obtenção do grau de tecnólogo junto ao curso de Análise e Desenvolvimento de Sistemas.
Orientador: Luciano Topolniak

Ariquemes
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

N244o

Nascimento, Bruno Irineu Silva do.

OdontoOpen - desenvolvimento de Software Open Source para automatização de clínicas odontológicas / Bruno Irineu Silva do Nascimento. - Ariquemes, 2025.

36 f. : il.

Orientador(a): Prof. Dr. Luciano Topolniak.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Ariquemes, 2025.

1. software odontológico. 2. Open source. 3. automação de clínicas. 4. Django. 5. Flutter. I. Topolniak, Luciano (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Renilce Silva Morais, CRB-11/906

BRUNO IRINEU SILVA DO NASCIMENTO

ODONTO OPEN: DESENVOLVIMENTO DE SOFTWARE OPEN SOURCE PARA AUTOMATIZAÇÃO DE CLÍNICAS ODONTOLÓGICAS

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Analista e Desenvolvedor de Sistemas" e aprovado em sua forma final pelo Curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Ariquemes, 05 de dezembro de 2025.

Banca Examinadora:

gov.br
Documento assinado digitalmente
LUCIANO TOPOLNIAK
Data: 17/12/2025 15:52:24-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Luciano Topolniak
(Orientador)
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia
(IFRO - Campus Ariquemes)

gov.br
Documento assinado digitalmente
ANDREY ALENCAR QUADROS
Data: 17/12/2025 16:22:50-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Andrey Alencar Quadros
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia
(IFRO - Campus Ariquemes)

gov.br
Documento assinado digitalmente
MARCOS ALVES FAINO
Data: 17/12/2025 18:52:01-0300
Verifique em https://validar.iti.gov.br

Prof. Especialista Marcos Alves Faino
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia
(IFRO - Campus Ariquemes)

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida e pela sabedoria concedida para superar os desafios desta jornada.

À minha esposa, à minha filha e a toda minha família, pelo amor incondicional, pela paciência nos momentos de ausência e pelo incentivo constante. Vocês são a minha base e minha maior motivação.

Ao meu orientador, Prof. Luciano Topolniak, pela direção, paciência e pelos valiosos ensinamentos que foram fundamentais para a concretização deste trabalho.

Aos professores Andrey Alencar Quadros, Claudinei de Oliveira, Vagner Schoaba, Marcos Alves Faino e Eudóxia Lottie Silva Moura pela excelência no ensino e por compartilharem seus conhecimentos. Estendo este agradecimento a toda a equipe docente do Instituto Federal de Rondônia (IFRO), que contribuiu significativamente para minha formação profissional e humana.

“ Ora, a fé é o firme fundamento das coisas que se esperam,
e a prova das coisas que se não veem.
(Hebreus 11:1) ”

RESUMO

Este relatório técnico apresenta o desenvolvimento do Odonto Open, um software *open source* de automação para clínicas odontológicas. O sistema visa modernizar a gestão clínica, automatizando processos como agendamento de consultas, gerenciamento de prontuários eletrônicos de pacientes e oferecendo um canal de acesso direto para pacientes via aplicativo *mobile*. Utilizando tecnologias modernas como Django para o *backend*, Flutter para o aplicativo *mobile* e Supabase como provedor de banco de dados e autenticação, o projeto propõe uma solução robusta, escalável e segura. A arquitetura do sistema foi projetada para ser modular, facilitando a manutenção e futuras expansões. A escolha pelo código aberto busca fomentar a colaboração da comunidade odontológica, reduzir custos de implementação para clínicas de pequeno e médio porte e garantir a conformidade com as normas do Conselho Federal de Odontologia (CFO) para prontuários digitais. Os resultados demonstram a viabilidade técnica da solução, criando uma plataforma integrada que melhora a eficiência operacional da clínica e a qualidade do atendimento ao paciente.

Palavras-chave: Software odontológico. *Open source*. Automação de clínicas. Django. Flutter.

ABSTRACT

This technical report presents the development of Odonto Open, an open-source automation software for dental clinics. The system aims to modernize clinical management by automating processes such as appointment scheduling, electronic patient health record management, and offering a direct access channel for patients via a mobile application. Using modern technologies such as Django for the backend, Flutter for the mobile application, and Supabase as a database and authentication provider, the project proposes a robust, scalable, and secure solution. The system architecture was designed to be modular, facilitating maintenance and future expansions. The choice of open source seeks to foster collaboration within the dental community, reduce implementation costs for small and medium-sized clinics, and ensure compliance with the Federal Council of Dentistry (CFO) standards for digital records. The results demonstrate the technical feasibility of the solution, creating an integrated platform that improves the clinic's operational efficiency and the quality of patient care.

Keywords: Dental software. Open source. Clinic automation. Django. Flutter.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT Associação Brasileira de Normas Técnicas

ACID Atomicidade, Consistência, Isolamento e Durabilidade

API Application Programming Interface

BaaS Backend as a Service

CDN Content Delivery Network

CFO Conselho Federal de Odontologia

CPF Cadastro de Pessoas Físicas

CRUD Create, Read, Update, Delete

DNS Domain Name System

DRF Django Rest Framework

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICP-Brasil Infraestrutura de Chaves Públicas Brasileira

IFRO Instituto Federal de Educação, Ciência e Tecnologia de Rondônia

iOS iPhone Operating System

JSON JavaScript Object Notation

JWT JSON Web Token

KPI Key Performance Indicator

MVVM Model-View-ViewModel

NPM Nginx Proxy Manager

ORM Object-Relational Mapping

PDF Portable Document Format

PEP Prontuário Eletrônico do Paciente

REST Representational State Transfer

SGBD Sistema Gerenciador de Banco de Dados

SQL Structured Query Language

SSL Secure Sockets Layer

TI Tecnologia da Informação

UML Unified Modeling Language

WAF Web Application Firewall

SUMÁRIO

1 INTRODUÇÃO

O setor de saúde odontológico enfrenta um desafio constante de modernização. Muitas clínicas ainda dependem de processos manuais, prontuários em papel e sistemas de agendamento telefônico, resultando em ineficiência e dificuldade no gerenciamento de dados do paciente. A transição para o digital é muitas vezes impedida pelo alto custo de *softwares* proprietários e pela complexidade de infraestrutura necessária.

Este trabalho dá continuidade e moderniza os conceitos explorados no projeto "COD (Clínica Odontológica)" [12], que validou a necessidade de um sistema *open source* para a área. O projeto Odonto Open reimagina essa solução, migrando para uma arquitetura *cloud-native* com Django, Supabase e Flutter, visando escalabilidade e segurança.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver o protótipo *open source* Odonto Open para automação clínica (Web/-Mobile), em conformidade com as normas documentais da área.

1.1.2 Objetivos Específicos

- Modelar e implementar um banco de dados relacional robusto utilizando PostgreSQL e Supabase, garantindo a integridade e perenidade dos dados de pacientes;
- Desenvolver uma API RESTful com Django Rest Framework para comunicação segura entre interfaces;
- Criar interfaces *Web* e *Mobile* para gestão eficiente de agenda e prontuários;
- Implementar infraestrutura com Docker para facilitar a política de *backups* e segurança dos dados;
- Estruturar o sistema para suportar futuras implementações de assinatura digital (ICP-Brasil), conforme exigido para documentos eletrônicos.

1.2 JUSTIFICATIVA

A justificativa para este projeto apoia-se na democratização tecnológica e na eficiência operacional. Softwares *open source* reduzem barreiras de entrada para pequenas clínicas, enquanto a arquitetura moderna (Web/Mobile) reflete a tendência de acesso remoto. Além disso, o sistema busca solucionar o problema crítico da gestão documental física, oferecendo uma alternativa digital organizada e segura.

1.3 FUNDAMENTAÇÃO LEGAL E NORMATIVA

A relevância do Odonto Open é reforçada pelas exigências legais que regem a odontologia no Brasil. O Conselho Federal de Odontologia (CFO) e a legislação vigente estabelecem a obrigatoriedade do arquivamento de documentos e registros dos pacientes, sendo fundamental para a segurança jurídica, planejamento do tratamento e identificação humana.

1.3.1 Imperativo Ético e Legal

O Código de Ética Odontológica (CEO) [5] determina a obrigatoriedade da manutenção de prontuários clínicos legíveis e atualizados. A ausência desse documento constitui infração ética. O prontuário completo serve como instrumento de defesa em processos civis, criminais ou éticos, além de ser crucial para a odontologia legal na identificação humana em casos de desastres ou desaparecimentos.

1.3.2 Diretrizes de Arquivamento e a Solução Digital

A Resolução CFO-SEC-91/2009 [6] estabelece diretrizes claras para a guarda documental:

- **Documentos em papel:** Prazo mínimo de 10 anos após o último registro.
- **Documentos eletrônicos:** A guarda deve ser permanente (*ad eternum*), devido à evolução tecnológica.

Neste contexto, o Odonto Open atua como uma ferramenta facilitadora do cumprimento dessas normas:

1. **Perenidade dos Dados:** Diferente do papel, que se deteriora, o uso do banco de dados PostgreSQL hospedado no Supabase permite o armazenamento seguro e permanente dos registros.
2. **Organização e Segurança:** O sistema padroniza o arquivamento digital, evitando perdas e confusões comuns em arquivos físicos. A arquitetura em contêineres Docker facilita a criação de rotinas de *backup* automatizadas, garantindo que a clínica cumpra a exigência de guarda permanente.
3. **Base para o Tratamento:** Ao centralizar anamneses, odontogramas e evoluções em uma interface digital acessível, o software assegura que o histórico do paciente esteja sempre disponível para o planejamento eficaz do tratamento.

1.4 PLANO ORGANIZACIONAL DO RELATÓRIO

O relatório está estruturado da seguinte forma: o Capítulo 2 detalha a arquitetura técnica e metodologia; o Capítulo 3 apresenta as interfaces desenvolvidas; e o Capítulo 4 discute os resultados frente aos desafios técnicos e normativos.

2 DESENVOLVIMENTO E METODOLOGIA

Esta seção detalha profundamente as ferramentas, a arquitetura da infraestrutura, a modelagem de dados e a implementação do código fonte do sistema Odonto Open. O desenvolvimento seguiu a metodologia ágil, com iterações focadas na modelagem de dados, exposição da API e desenvolvimento das interfaces clientes (*Web* e *Mobile*).

2.1 TECNOLOGIAS UTILIZADAS

A seleção da *stack* tecnológica priorizou ferramentas de código aberto, ampla comunidade e suporte a desenvolvimento rápido. A Tabela 1 resume as camadas da aplicação e suas respectivas justificativas técnicas.

Tabela 1 – Stack Tecnológica Detalhada do Projeto.

Camada	**Tecnologia**	**Descrição e Justificativa**
Backend	Django 5.0 (Python)	*Framework web* de alto nível [7]. Escolhido pela robustez do seu ORM e segurança nativa contra SQL Injection e XSS.
API	Django Rest Framework	Biblioteca para construção de APIs RESTful. Facilita a serialização de dados complexos e autenticação via *Token*.
Mobile	Flutter (Dart)	SDK do Google [9] para criar *apps* nativos compilados. Permite uma única base de código para Android e iOS com alta performance (60fps).
Banco de Dados	PostgreSQL 15	SGBD relacional objeto-relacional [14]. Escolhido pela conformidade ACID, essencial para dados críticos de saúde.
Infraestrutura	Docker	Containerização dos serviços [8] para garantir isolamento e paridade entre os ambientes de desenvolvimento e produção.
BaaS	Supabase	Plataforma *open source* [13] que fornece Auth, Storage e Realtime Database, agilizando o desenvolvimento do *backend*.

Fonte: Elaborado pelo autor (2025).

2.2 ARQUITETURA DE INFRAESTRUTURA E SEGURANÇA

A infraestrutura do sistema foi orquestrada utilizando Docker Compose para garantir isolamento e portabilidade. Além dos contêineres de aplicação e banco de dados, implementou-se uma camada robusta de segurança e roteamento composta pelo **Nginx Proxy Manager** e **Cloudflare**.

2.2.1 Serviços Docker

Uma decisão estratégica foi a utilização do **Supabase Self-Hosted** via Docker, garantindo soberania sobre os dados. A Tabela 2 descreve tecnicamente cada contêiner ativo.

Tabela 2 – Serviços e Portas da Infraestrutura Docker.

Serviço	**Porta Interna**	**Função Técnica**
`db`	5432	Instância do PostgreSQL. Persistência de dados relacionais com volumes Docker para *backup*.
`studio`	3000	Interface de gerenciamento visual do banco, permitindo execução de SQL e visualização de tabelas.
`auth`	9999	Servidor GoTrue responsável pela emissão de *tokens* JWT e gerenciamento de ciclo de vida de usuários.
`kong`	8000	API Gateway que centraliza as requisições e roteia para os serviços internos corretos.
`web`	8000	Aplicação Django servida via Gunicorn, contendo a lógica de negócio e API.

Fonte: Elaborado pelo autor (2025).

2.2.2 Gestão de Tráfego e SSL (Nginx Proxy Manager)

Para expor a aplicação de forma segura, utilizou-se o **Nginx Proxy Manager (NPM)**. Este serviço atua como um *proxy* reverso, recebendo as requisições externas nas portas 80 (HTTP) e 443 (HTTPS) e encaminhando-as para o contêiner interno correto.

Além do roteamento, o NPM gerencia automaticamente os certificados SSL/TLS através da integração com a autoridade certificadora *Let's Encrypt*, garantindo que todo o tráfego seja criptografado. A Figura 1 ilustra a interface de gerenciamento de *Proxy Hosts*.

Figura 1 – Configuração de Proxy e SSL no Nginx Proxy Manager.

Fonte: Elaborado pelo autor (2025).

2.2.3 DNS e Proteção de Borda (Cloudflare)

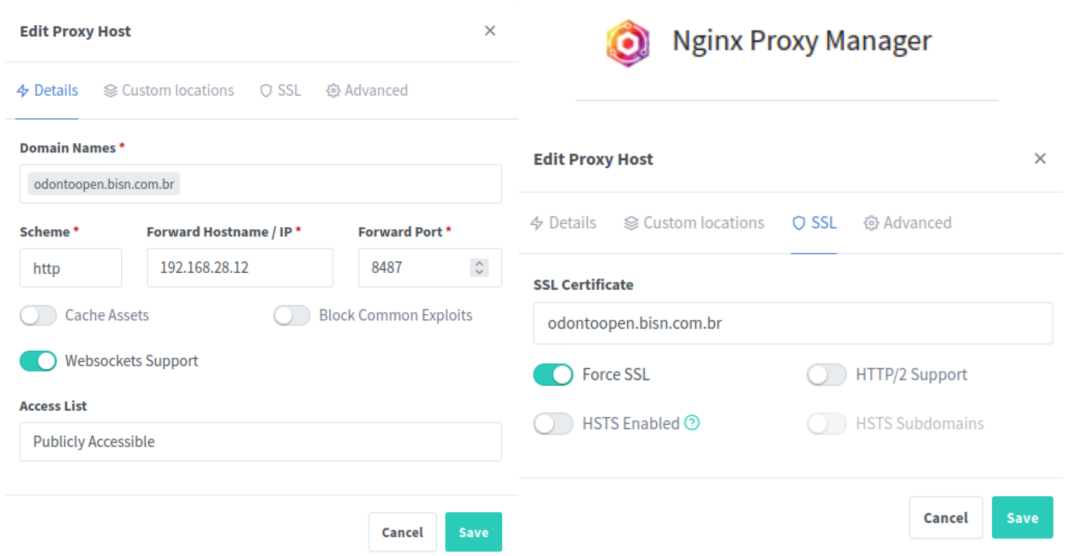
A gestão de DNS (*Domain Name System*) é realizada através do **Cloudflare**. O domínio da aplicação aponta para os servidores do Cloudflare, que atuam como uma camada de proteção de borda antes que o tráfego atinja o servidor da clínica.

As principais vantagens desta integração incluem:

- ***Proxy* Reverso (Nuvem Laranja):** Oculta o IP real do servidor de origem, mitigando ataques diretos de negação de serviço (DDoS).
- **WAF (*Web Application Firewall*):** Permite a criação de regras de *firewall* personalizadas para bloquear tráfego malicioso, SQL Injection e robôs automatizados.
- **Modo SSL Full (Strict):** Garante criptografia ponta a ponta, exigindo um certificado válido (gerado pelo NPM) no servidor de origem.

A Figura 2 demonstra o painel de gerenciamento de DNS e as regras de segurança aplicadas.

Figura 2 – Gerenciamento de DNS e Regras no Cloudflare.



Fonte: Elaborado pelo autor (2025).

2.3 IMPLEMENTAÇÃO DO *BACKEND* (DJANGO)

A lógica de negócios reside no Django. A estrutura de dados utiliza o ORM (*Object-Relational Mapping*) para mapear classes Python em tabelas do PostgreSQL.

2.3.1 Modelagem de Dados e Esquema

A integridade dos dados é a prioridade do sistema. Abaixo, detalhamos a estrutura das principais tabelas geradas pelo Django no PostgreSQL.

Tabela: Pacientes

A classe `Paciente` (`pacientes/models.py`) armazena os dados cadastrais. A Tabela 3 detalha seus atributos.

Tabela 3 – Dicionário de Dados: Tabela Pacientes.

Campo	**Tipo**	**Obrigatório**	**Descrição**
id	BigInt	Sim	Chave primária autoincremental.
nome_completo	Varchar(200)	Sim	Nome completo do paciente.
cpf	Varchar(14)	Sim	Cadastro de Pessoa Física (Único no sistema).
data_nascimento	Date	Sim	Utilizado para cálculo automático da idade.
sexo	Char(1)	Sim	M (Masculino), F (Feminino) ou O (Outro).
criado_em	DateTime	Sim	*Timestamp* automático de criação.

Fonte: Elaborado pelo autor (2025).

Tabela: Consultas

A classe `Consulta` (`agenda/models.py`) gerencia o agendamento. O código a seguir (Trecho de código 2.1) demonstra as relações e regras de negócio.

Trecho de código 2.1 – Modelo de Dados da Consulta (agenda/models.py).

```
class Consulta(models.Model):
    STATUS_CHOICES = [
        ('agendada', 'Agendada'),
        ('confirmada', 'Confirmada'),
        ('cancelada', 'Cancelada'),
        ('realizada', 'Realizada'),
    ]

    # Relacionamentos
    paciente = models.ForeignKey(Paciente, on_delete=models.CASCADE)
    dentista = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        related_name='consultas_dentista'
    )

    # Dados do Agendamento
    data_horario = models.DateTimeField('Data e Hora')
    status = models.CharField(
        max_length=20,
        choices=STATUS_CHOICES,
        default='agendada'
    )
    observacoes = models.TextField('Observações', blank=True)

    class Meta:
        ordering = ['-data_horario']
```

```
28         verbose_name = 'Consulta'
```

Fonte: Elaborado pelo autor (2025).

2.3.2 Serialização de Dados (Serializers)

Para que os dados dos modelos possam ser trafegados via API, utilizamos *Serializers* do DRF. O `ConsultaSerializer` (Trecho de código 2.2) converte objetos complexos do banco de dados para JSON e vice-versa, validando os dados de entrada.

Trecho de código 2.2 – Serializador de Consultas (agenda/serializers.py).

```
1  class ConsultaSerializer(serializers.ModelSerializer):
2      # Campos aninhados para leitura amigável
3      paciente_nome = serializers.CharField(
4          source='paciente.nome_completo',
5          read_only=True
6      )
7      dentista_nome = serializers.CharField(
8          source='dentista.get_full_name',
9          read_only=True
10     )
11
12     class Meta:
13         model = Consulta
14         fields = [
15             'id', 'paciente', 'paciente_nome',
16             'dentista', 'dentista_nome',
17             'data_horario', 'status'
18         ]
```

Fonte: Elaborado pelo autor (2025).

2.4 IMPLEMENTAÇÃO AVANÇADA DO *MOBILE* (FLUTTER)

O aplicativo móvel foi desenvolvido utilizando a arquitetura **MVVM** (*Model-View-ViewModel*), garantindo a separação entre a interface (UI) e a lógica de dados.

2.4.1 Configuração Inicial e Rotas

O arquivo `main.dart` (Trecho de código 2.3) é o ponto de entrada da aplicação. Ele configura o sistema de injeção de dependência (`MultiProvider`), o tema visual e as rotas de navegação.

Trecho de código 2.3 – Configuração Principal do App (lib/main.dart).

```
1  void main() {
2    runApp(
```

```
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => AppProvider()),
      ],
      child: OdontoOpenApp(),
    ),
  );
}

class OdontoOpenApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'OdontoOpen',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        useMaterial3: true,
      ),
      initialRoute: '/login',
      routes: {
        '/login': (ctx) => LoginScreen(),
        '/home': (ctx) => HomeScreen(),
        '/agendamento': (ctx) => AgendamentoScreen(),
      },
    );
  }
}
```

Fonte: Elaborado pelo autor (2025).

2.4.2 Gerenciamento de Estado e Sessão

O estado da aplicação (usuário logado, *token* de autenticação) é gerenciado pelo `AppProvider`. Este componente utiliza o padrão *Observer* para notificar todas as telas quando o estado muda (ex: *login* realizado com sucesso), atualizando a interface reativamente.

2.4.3 Integração com API Backend

A comunicação com o servidor Django é realizada pela classe `ApiService`. O método de *login* (Trecho de código 2.4) exemplifica uma requisição HTTP POST assíncrona, tratamento de erro e *parsing* da resposta JSON.

Trecho de código 2.4 – Serviço de Autenticação (lib/services/api_service.dart).

```
class ApiService {
  final String baseUrl = 'http://odontoopen.bisn.com.br/api';
```

```
3
4   Future<String> login(String username, String password) async {
5     try {
6       final response = await http.post(
7         Uri.parse('$baseUrl/auth/login/'),
8         body: {
9           'username': username,
10          'password': password
11        },
12      );
13
14      if (response.statusCode == 200) {
15        // Decodifica JSON e extrai o token
16        final data = json.decode(response.body);
17        return data['token'];
18      } else {
19        throw Exception('Falha na autenticação: ${response.statusCode}')
    ;
20      }
21    } catch (e) {
22      throw Exception('Erro de conexão: $e');
23    }
24  }
25 }
```

Fonte: Elaborado pelo autor (2025).

2.5 VERSIONAMENTO E CONTROLE DE CÓDIGO

Para garantir a rastreabilidade e segurança do código, utilizou-se o sistema de controle de versão **Git**. O código fonte completo está disponível publicamente na plataforma **GitHub**, reafirmando o compromisso com a filosofia *open source*.

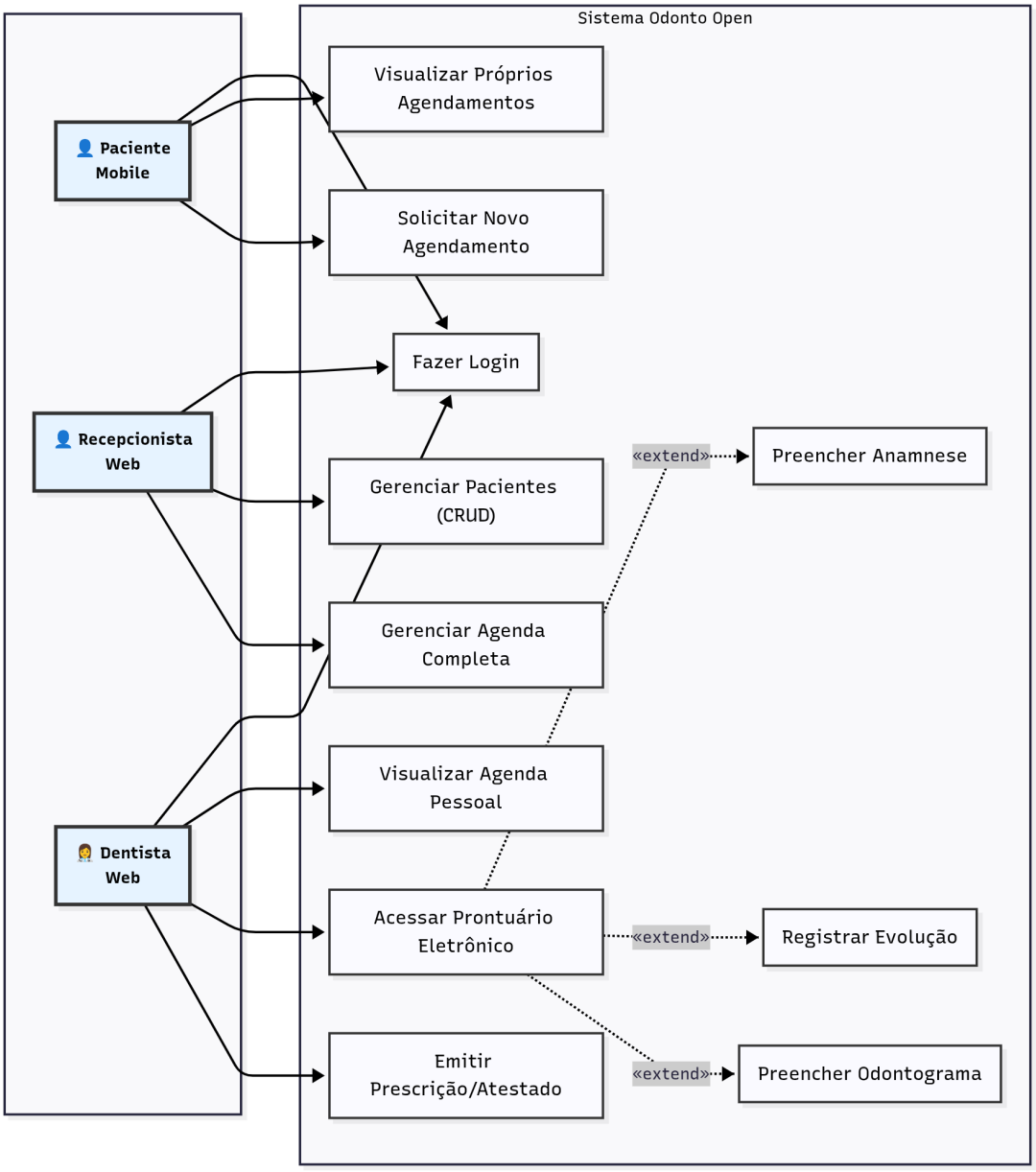
Os repositórios oficiais do projeto são:

- **Backend (Django/API):** https://github.com/irineubruno/odontoopen
- **Aplicativo Mobile:** https://github.com/irineubruno/odontoopen-mobile

2.5.1 Diagramas UML

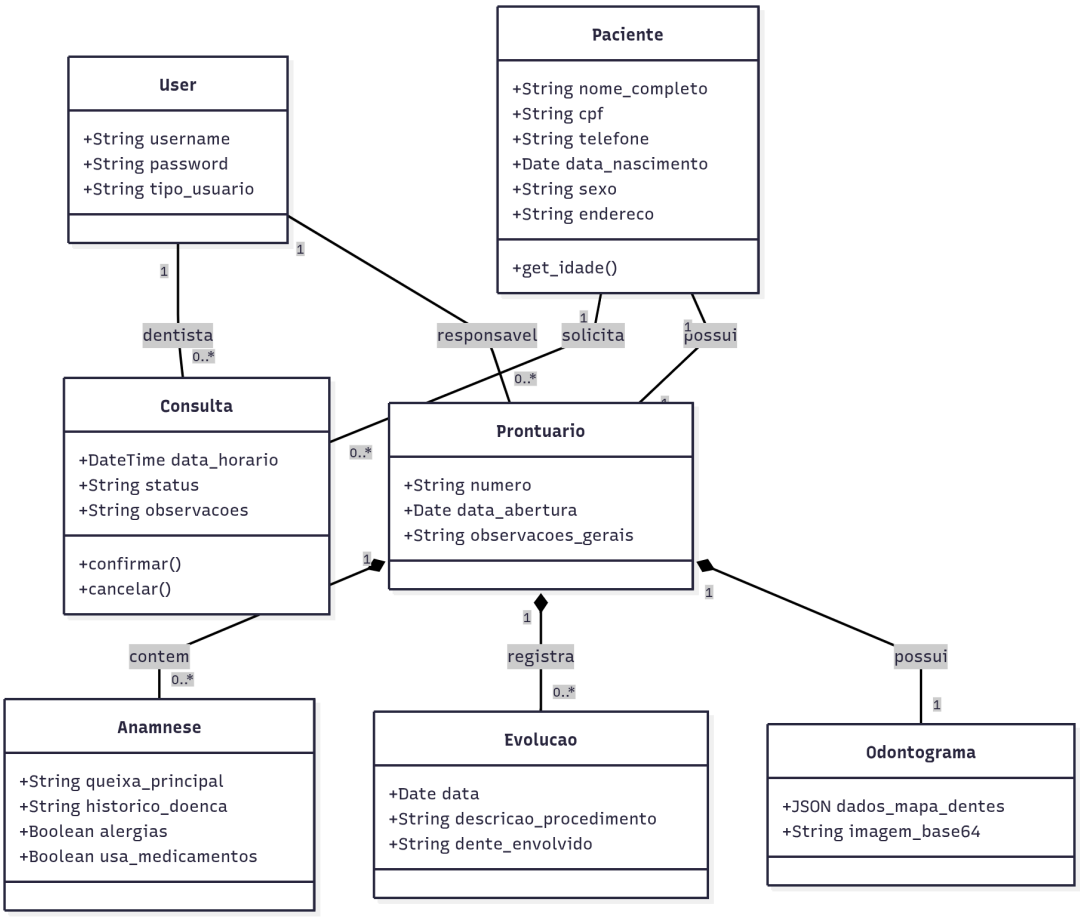
Para documentar a estrutura e o comportamento do sistema, foram elaborados diagramas UML. A Figura 3 ilustra as interações dos atores (Paciente, Dentista, Recepcionista) com o sistema, enquanto a Figura 4 detalha a estrutura das classes principais.

Figura 3 – Diagrama de Casos de Uso (Odonto Open).



Fonte: Elaborado pelo autor (2025).

Figura 4 – Diagrama de Classes do Domínio.



Fonte: Elaborado pelo autor (2025).

3 INTERFACE DO USUÁRIO E IMPLEMENTAÇÃO (RESULTADOS)

Nesta seção são apresentados os resultados visuais do desenvolvimento, detalhando como os requisitos funcionais foram materializados em interfaces de usuário finais. Cada tela apresentada reflete a execução do código documentado no capítulo anterior, demonstrando a integração entre o *backend* robusto em Django e as interfaces clientes (*Web* e *Mobile*).

A interface do usuário foi projetada seguindo princípios de usabilidade e hierarquia visual, visando reduzir a curva de aprendizado para os profissionais de saúde e oferecer agilidade no atendimento diário.

3.1 SISTEMAS WEB (GESTÃO CLÍNICA E ADMINISTRATIVA)

A interface web atua como o painel de controle central da clínica. Desenvolvida utilizando a linguagem de templates do Django [7] combinada com HTML5 e CSS3 (*framework* Bootstrap), ela oferece uma experiência responsiva, adaptando-se a monitores de diferentes resoluções, comuns em recepções e consultórios.

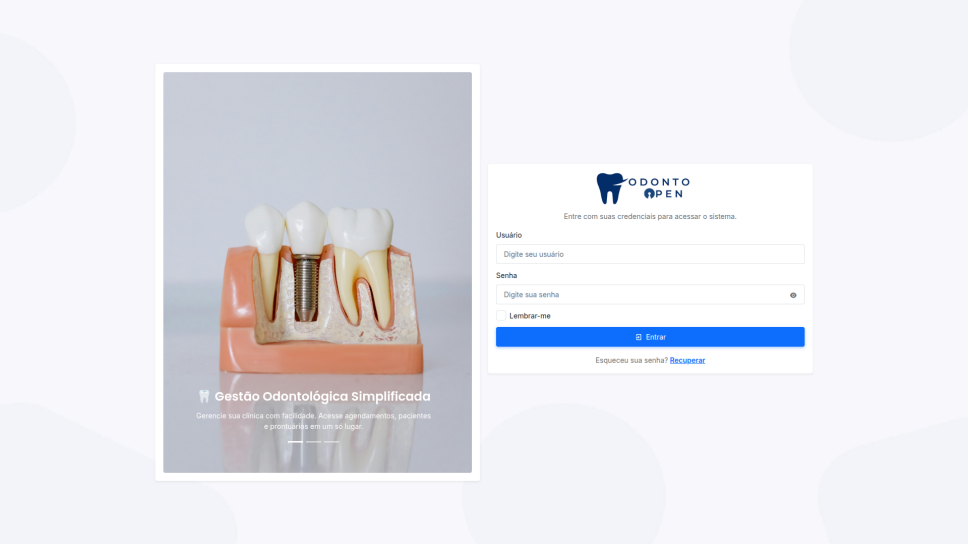
3.1.1 Controle de Acesso e Segurança

A segurança dos dados sensíveis de saúde começa na tela de *Login* (Figura 5). Esta interface é a única porta de entrada para o sistema administrativo, restringindo o acesso apenas a usuários previamente cadastrados e com permissões ativas (Recepcionistas, Dentistas e Administradores).

Tecnicamente, o formulário implementa medidas de segurança cruciais:

- **Proteção CSRF:** Utiliza o *token* nativo do Django para impedir falsificação de solicitações entre sites.
- **Feedback Visual:** Mensagens de erro claras são exibidas caso as credenciais sejam inválidas, sem expor se o erro foi no usuário ou na senha, dificultando ataques de força bruta.
- **Redirecionamento Inteligente:** Após a autenticação, o sistema redireciona o usuário para a página que ele tentou acessar originalmente ou para o *Dashboard* padrão.

Figura 5 – Tela de Login do Sistema Web.



Fonte: Elaborado pelo autor (2025).

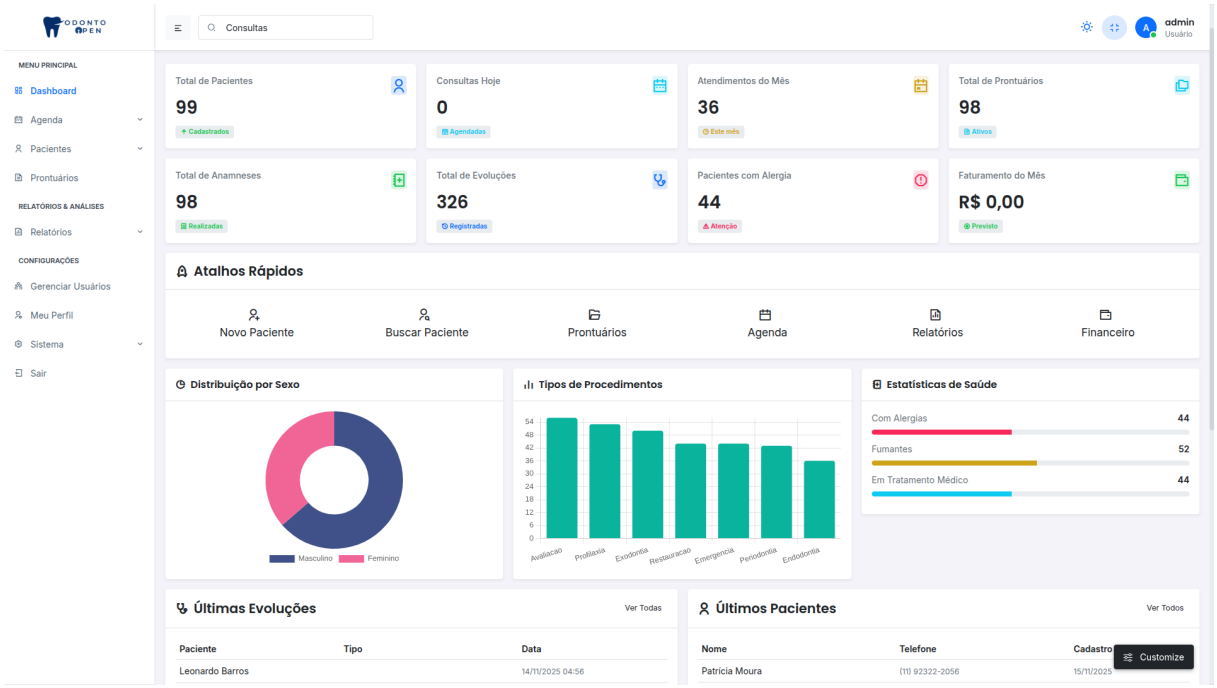
3.1.2 Dashboard e Indicadores de Desempenho

Ao acessar o sistema, o profissional é recebido pelo *Dashboard* (Figura 6). Esta tela foi projetada para oferecer uma visão macroscópica da saúde do negócio e do fluxo clínico do dia.

O *Dashboard* não é estático; ele consome dados agregados do banco de dados PostgreSQL [14] em tempo real, processados por *views* otimizadas no Django. Os principais indicadores (KPIs) apresentados incluem:

- **Total de Pacientes:** Contagem global da base de dados, permitindo acompanhar o crescimento da clínica.
- **Atendimentos do Dia:** Filtra a tabela de Consultas para a data atual, permitindo que a equipe se prepare para a demanda do dia.
- **Resumo Financeiro:** Exibe uma prévia do faturamento baseada nos procedimentos realizados, auxiliando na gestão administrativa.
- **Alertas de Retorno:** Identifica pacientes que não comparecem à clínica há determinado tempo, facilitando ações de fidelização.

Figura 6 – Dashboard Administrativo com Indicadores.



Fonte: Elaborado pelo autor (2025).

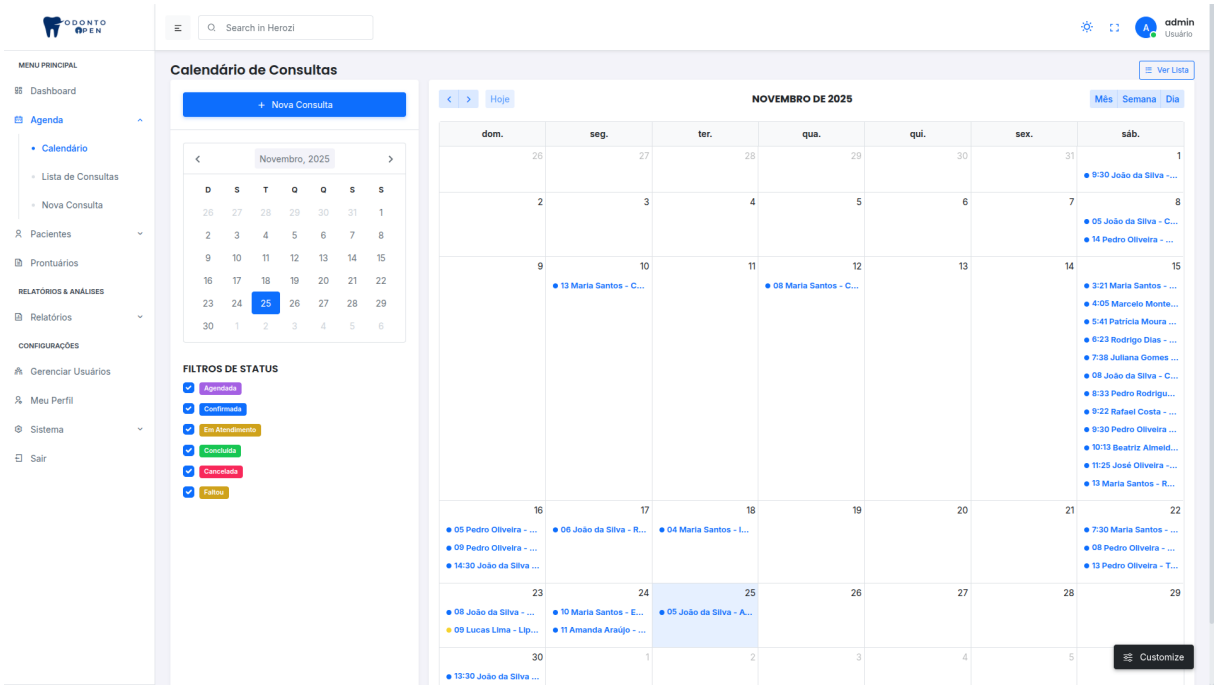
3.1.3 Gestão Visual da Agenda

A Agenda (Figura 7) é a ferramenta de trabalho principal da recepção. Para substituir a agenda de papel, desenvolveu-se uma interface de calendário interativa que permite a manipulação intuitiva dos horários.

Funcionalidades implementadas nesta tela:

- **Visão Multiprofissional:** A recepcionista pode filtrar a agenda por dentista, visualizando apenas os horários daquele profissional, ou ter uma visão geral da ocupação da clínica.
- **Códigos de Cores:** O status de cada consulta é representado visualmente (ex: Amarelo para "Agendada", Verde para "Confirmada", Vermelho para "Cancelada"), permitindo leitura rápida da situação.
- **Prevenção de Conflitos:** Ao tentar criar um agendamento, o *backend* verifica automaticamente se o horário já está ocupado para aquele dentista, retornando um erro de validação e impedindo o *double-booking*.

Figura 7 – Interface de Calendário da Agenda.



Fonte: Elaborado pelo autor (2025).

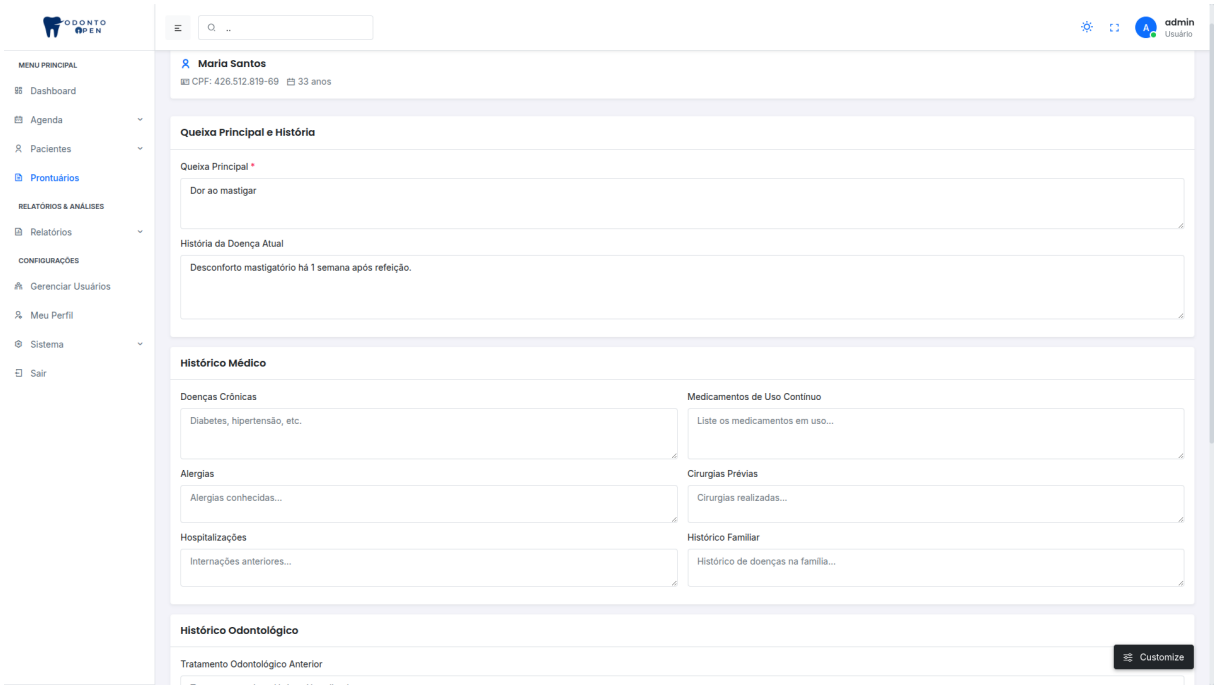
3.2 PRONTUÁRIO ELETRÔNICO DO PACIENTE (PEP)

O Prontuário Eletrônico (Figura 8) é o coração do sistema clínico. Ele foi arquitetado para centralizar e organizar todo o histórico de saúde do paciente, garantindo a integridade e a disponibilidade da informação, requisitos fundamentais segundo as normas do CFO [6].

A interface utiliza um sistema de abas para segmentar a informação sem sobrecarregar a tela:

- **Dados Pessoais:** Informações cadastrais e contatos.
- **Anamnese:** Questionário de saúde preenchido pelo dentista.
- **Evoluções:** Histórico cronológico dos atendimentos.
- **Odontograma:** Representação gráfica da saúde bucal.
- **Documentos:** Repositório de arquivos, prescrições e atestados.

Figura 8 – Prontuário Eletrônico: Visão Geral.



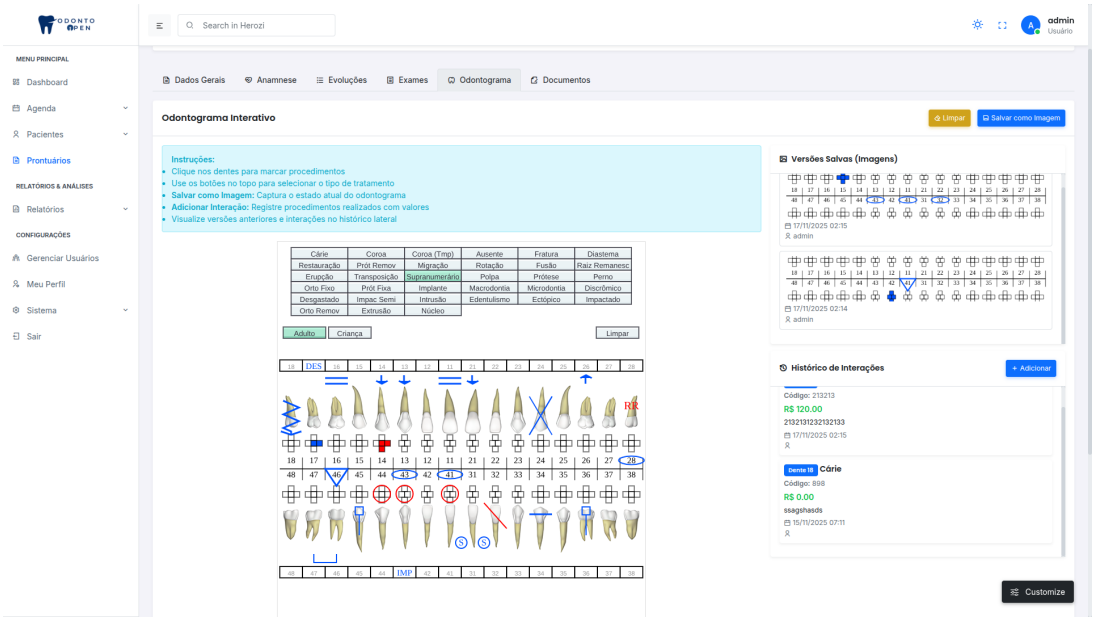
Fonte: Elaborado pelo autor (2025).

3.2.1 Odontograma Digital Interativo

Para facilitar o registro gráfico das condições de saúde bucal, implementou-se o Odontograma Interativo (Figura 9). Diferente de uma imagem estática, este componente é interativo: o dentista clica na face do dente desejado e seleciona o procedimento ou condição (ex: Cárie, Restauração, Ausente).

No banco de dados, essas informações são salvas de forma estruturada (JSON), vinculando o número do dente (notação FDI) ao procedimento realizado. Isso permite gerar estatísticas futuras e manter um histórico visual da evolução do tratamento.

Figura 9 – Odontograma Digital Interativo.



Fonte: Elaborado pelo autor (2025).

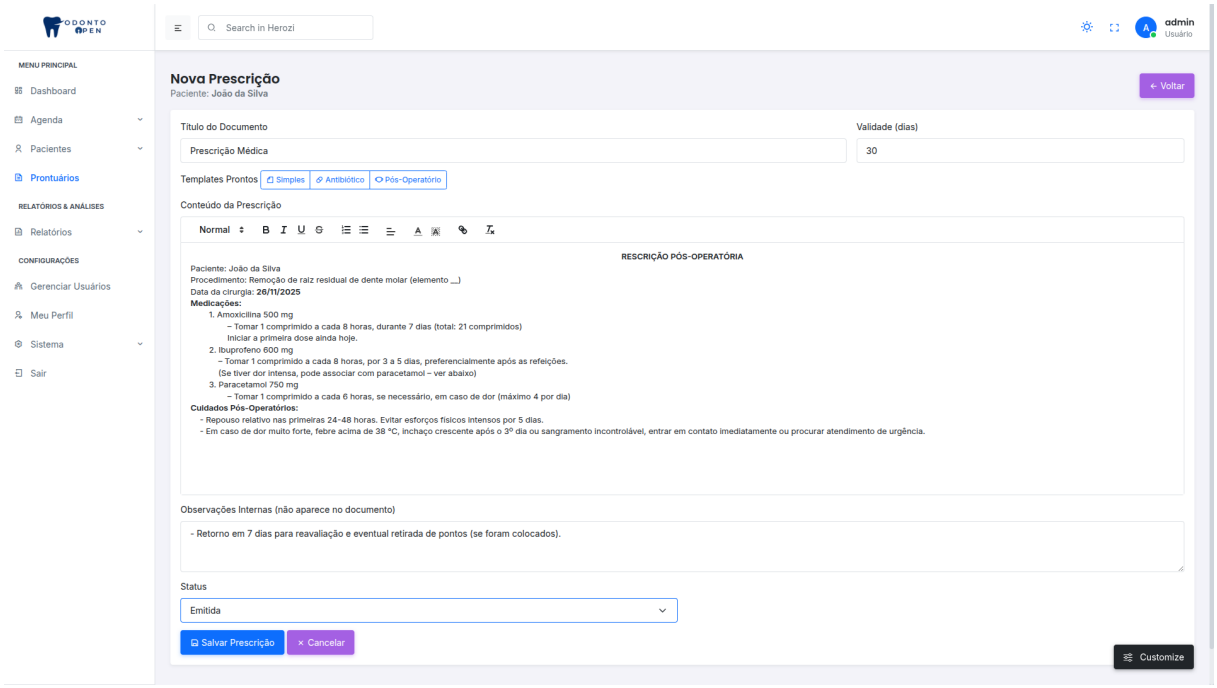
3.2.2 Documentos Médicos: Prescrições e Atestados

A segurança do paciente depende diretamente da clareza das prescrições médicas. O sistema Odonto Open implementa um módulo dedicado (Figura 10) para a emissão de receitas e atestados, visando eliminar os riscos associados à ilegibilidade da caligrafia manual.

Este módulo utiliza o modelo `Prescricao` documentado no capítulo anterior para gerar documentos padronizados. A interface oferece:

- **Editor de Texto Rico (WYSIWYG):** Permite ao dentista formatar o texto, criar listas de medicamentos e inserir observações com clareza.
- **Modelos Pré-definidos:** O sistema permite salvar modelos de receitas comuns (ex: "Pós-operatório de Extração", "Antibioticoterapia"), agilizando o atendimento.
- **Histórico Imutável:** Uma vez emitida, a prescrição fica salva no histórico do paciente, permitindo consultas futuras sobre quais medicamentos foram receitados e quando.

Figura 10 – Editor de Prescrições e Atestados.



Fonte: Elaborado pelo autor (2025).

3.3 APLICATIVO MÓVEL (EXPERIÊNCIA DO PACIENTE)

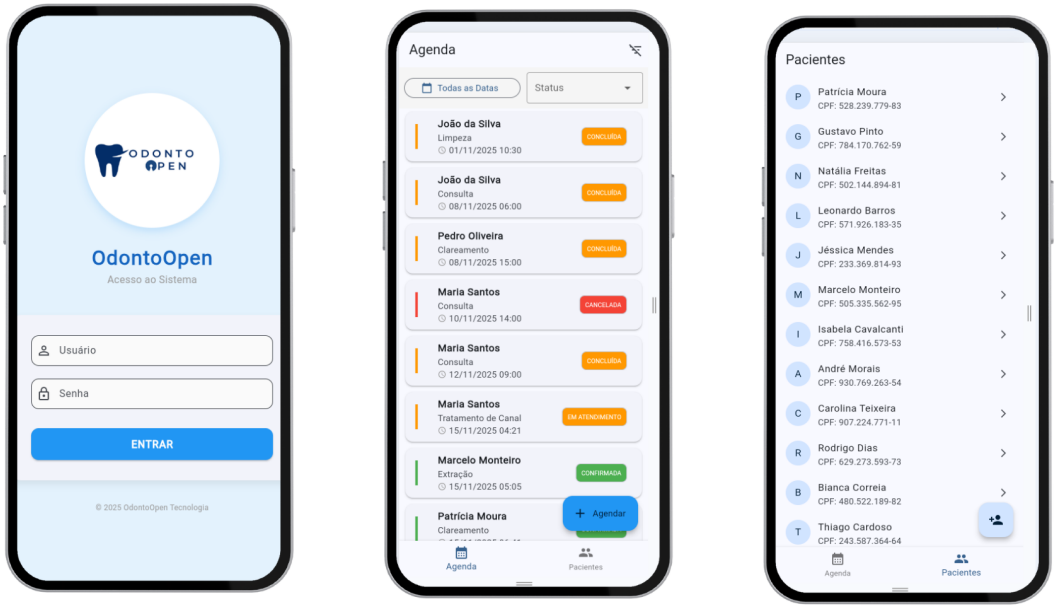
O aplicativo móvel, desenvolvido em Flutter [9], foca na autonomia do paciente. A Figura 11 ilustra a jornada do usuário através das telas principais:

1. **Login (Esquerda):** Interface limpa que solicita CPF e senha. Ao clicar em "Entrar", o app realiza uma requisição POST para `/api/auth/login/`. Se bem-sucedido, o *token* JWT retornado é salvo no dispositivo para manter a sessão ativa.

2. **Home/Dashboard (Centro):** Apresenta ao paciente suas próximas consultas confirmadas e um histórico simplificado. Esta tela consome o *endpoint* `/api/consultas/` filtrando pelo ID do usuário logado.

3. **Agendamento (Direita):** Permite que o paciente selecione uma data no calendário. O aplicativo consulta a API em tempo real para mostrar apenas os horários disponíveis, evitando a frustração de tentar marcar em horários ocupados.

Figura 11 – Telas do App Mobile: Login, Home e Agendamento.



Fonte: Elaborado pelo autor (2025).

4 CONSIDERAÇÕES FINAIS

O projeto Odonto Open atingiu com êxito seu objetivo principal de desenvolver uma solução de *software open source* moderna, distribuída e tecnicamente robusta para a automação de clínicas odontológicas. A transição da arquitetura legada (*Desktop*/Local) para uma arquitetura em nuvem e móvel (*Web*/*Mobile*) provou ser essencial para atender às demandas atuais de conectividade e acesso remoto.

4.1 ANÁLISE DOS RESULTADOS TÉCNICOS

A implementação técnica demonstrou alta qualidade nos seguintes aspectos:

- **Desacoplamento *Frontend*/*Backend*:** A utilização do Django Rest Framework [7] permitiu que o mesmo *backend* servisse dados tanto para a interface administrativa *Web* quanto para o aplicativo móvel Flutter, validando a eficiência da arquitetura API RESTful.
- **Performance *Mobile*:** O uso do *framework* Flutter [9] resultou em um aplicativo fluido e responsivo, capaz de rodar em dispositivos Android e iOS com uma única base de código, reduzindo drasticamente o tempo de desenvolvimento e manutenção.
- **Segurança e Infraestrutura:** A containerização com Docker [8] e o uso do Supabase [13] (PostgreSQL [14]) garantiram um ambiente seguro, com dados isolados e fácil replicabilidade em diferentes servidores.

4.2 CONTRIBUIÇÃO PARA A COMUNIDADE E SOCIEDADE

Ao disponibilizar o código fonte sob licença livre no **GitHub**, o projeto contribui para a democratização da tecnologia em saúde. Clínicas de pequeno porte, estudantes e desenvolvedores podem utilizar, estudar e modificar o sistema sem custos de licenciamento, fomentando a inovação no setor.

Do ponto de vista clínico, o sistema entrega valor imediato ao centralizar os registros, eliminar o risco de perda de prontuários físicos e garantir a legibilidade das informações (anamneses e evoluções), auxiliando no cumprimento das normas do Conselho Federal de Odontologia (CFO) [6].

4.3 TRABALHOS FUTUROS

Para a evolução contínua do Odonto Open rumo a um produto de mercado, os próximos passos são as seguintes implementações:

- **Assinatura Digital (ICP-Brasil):** Implementação obrigatória para validade jurídica plena dos documentos eletrônicos [11], permitindo a eliminação total do papel.

- **Módulo Financeiro:** Desenvolvimento de fluxo de caixa, gestão de convênios, contas a pagar/receber e emissão de notas fiscais eletrônicas (NFSe).
- **Notificações Inteligentes:** Integração com Firebase Cloud Messaging (FCM) para enviar lembretes automáticos de consulta aos pacientes, reduzindo o absenteísmo.
- **Teleodontologia:** Integração de módulo de videochamada para triagem e acompanhamento remoto de pacientes.

Conclui-se que o Odonto Open estabelece uma fundação sólida e escalável, cumprindo seu papel acadêmico e social.

REFERÊNCIAS

1 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10520: Informação e documentação: citações em documentos: apresentação**. Rio de Janeiro, 2023.

2 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 10719: Informação e documentação: relatório técnico e/ou científico: apresentação**. Rio de Janeiro, 2015.

3 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023: Informação e documentação: referências: elaboração**. Rio de Janeiro, 2018.

4 ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6028: Informação e documentação: resumo: apresentação**. Rio de Janeiro, 2021.

5 CONSELHO FEDERAL DE ODONTOLOGIA. **Código de Ética Odontológica: aprovado pela Resolução CFO-118/2012**. Rio de Janeiro, 2012. Disponível em: https://website.cfo.org.br/wp-content/uploads/2013/07/codigo_etica.pdf. Acesso em: 25 nov. 2025.

6 CONSELHO FEDERAL DE ODONTOLOGIA. **Resolução CFO-91/2009: Normatiza o prontuário odontológico**. Rio de Janeiro, 2009. Estabelece normas para adoção de arquivo digital e digitalização de prontuários.

7 DJANGO SOFTWARE FOUNDATION. **Django Documentation**. [*S.l.*], 2025. Disponível em: https://docs.djangoproject.com/. Acesso em: 25 nov. 2025.

8 DOCKER INC. **Docker Documentation**. [*S.l.*], 2025. Disponível em: https://docs.docker.com/. Acesso em: 25 nov. 2025.

9 GOOGLE. **Flutter: Build apps for any screen**. [*S.l.*], 2025. Disponível em: https://flutter.dev/. Acesso em: 25 nov. 2025.

10 INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Normas de apresentação tabular**. 3. ed. Rio de Janeiro: IBGE, 1993.

11 INSTITUTO NACIONAL DE TECNOLOGIA DA INFORMAÇÃO. **Infraestrutura de Chaves Públicas Brasileira - ICP-Brasil**. [*S.l.*], 2025. Disponível em: https://www.gov.br/iti/pt-br. Acesso em: 25 nov. 2025.

12 NASCIMENTO, B. I. S.; BARRANCO, A.; TOPOLNIAK, L. **COD (Clínica Odontológica): Projeto de automação em Java e MySQL**. [*S.l.*], 2022.

13 SUPABASE. **Supabase Documentation**. [*S.l.*], 2025. Disponível em: https://supabase.com/docs. Acesso em: 25 nov. 2025.

14 THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL 15 Documentation**. [*S.l.*], 2025. Disponível em: https://www.postgresql.org/docs/. Acesso em: 25 nov. 2025.