Documento de Especificação de Requisitos e Arquitetura

Sistema de Gestão de Clínica Odontológica (OdontoSys Lite)

Elaborado por: Luidy Costa dos Santos

3 de dezembro de 2025

1 Requisitos Funcionais

Liste e descreva tudo o que o sistema deve fazer.

- **Gestão de Pacientes:** O sistema deve permitir registar, visualizar, editar e eliminar pacientes. Dados obrigatórios: Nome, CPF, Data de Nascimento e Telefone.
- **Prontuário Eletrónico:** O sistema deve permitir registar evoluções clínicas no cadastro do paciente.
  - *Nota:* Funcionalidade de upload de imagens (Raio-X) não incluída na versão Lite.
- **Gestão de Agendamentos:** O sistema deve permitir agendar, visualizar e cancelar consultas, vinculando um paciente a uma data e horário.
- **Controlo de Acesso (Login):** O sistema deve autenticar utilizadores mediante login e senha para diferenciar perfis de acesso.
- **Gestão Financeira:** O sistema deve permitir o lançamento de valores recebidos e a visualização do total faturado no período.
- **Limitações (Âmbito Negativo):**
  - O sistema **não** enviará notificações automáticas (WhatsApp/Email) nesta versão.
  - O sistema **não** gerará gráficos visuais complexos, apenas relatórios textuais/tabulares.

2 Requisitos Não Funcionais

Descreva como o sistema deve comportar-se.

- **Segurança e Autenticação:** O acesso ao sistema é restrito por login. As senhas não devem ser armazenadas em texto simples.
- **Interface Amigável:** A interface deve ser limpa, intuitiva e responsiva, utilizando CSS para boa visualização em desktops e tablets.
- **Acesso via Navegador (Web-based):** O sistema deve ser executado inteiramente no navegador do cliente, sem instalação de executáveis (`.exe`).
- **Desempenho e Disponibilidade:** O sistema deve responder instantaneamente às interações do utilizador (tempo de resposta < 200ms) e funcionar offline (após carregado), dependendo apenas do navegador.
- **Persistência de Dados:** Os dados devem ser persistidos localmente no navegador (`localStorage`) para garantir continuidade entre sessões.

3 Módulos Principais

Identifique e descreva os módulos do sistema.

O sistema é dividido logicamente nos seguintes módulos funcionais:

1. **Módulo de Receção:** Responsável pelas operações diárias de atendimento, incluindo o registo de novos pacientes e a gestão visual da agenda de consultas.
2. **Módulo Clínico:** Focado no registo médico. Permite ao dentista aceder e atualizar o histórico de tratamentos (prontuário) de cada paciente.
3. **Módulo Financeiro:** Gere o fluxo de caixa simples, permitindo o lançamento de pagamentos recebidos e a visualização de totais.
4. **Módulo de Autenticação:** Gere o controlo de acesso, validando credenciais e definindo o que cada utilizador pode ver.

4 Utilizadores do Sistema

Descreva os perfis que terão acesso ao sistema.

- **Administrador / Dentista (Dr. Ricardo):**
  - Tem acesso total ao sistema.
  - Pode visualizar e editar Prontuários Clínicos.
  - Pode visualizar relatórios financeiros completos.
- **Secretária:**
  - Tem acesso focado na organização.
  - Pode gerir a Agenda (marcar/desmarcar) e registar Pacientes.
  - Pode lançar pagamentos, mas **não** tem permissão para ler detalhes sensíveis do Prontuário Clínico.

5 Design e Arquitetura de Software

Descreva brevemente a arquitetura escolhida.

5.1 Modelo de Arquitetura: Monolítica (Client-Side)

A aplicação segue o modelo de **Single Page Application (SPA)** monolítica. Todos os componentes (apresentação, lógica e dados) são entregues ao navegador num pacote único.

5.2 Tecnologias Possíveis

- **Front-end:** HTML5 (Estrutura), CSS3 (Estilo), JavaScript (Lógica).
- **Back-end:** Simulado via JavaScript no navegador.
- **Base de Dados:** `localStorage` (NoSQL Key-Value Store do navegador).

5.3 Estrutura Geral e Divisão de Camadas

Para evitar o "código espaguete", o JavaScript será organizado internamente no padrão **MVC (Model-View-Controller)**:

1. **Camada de Dados (Model):** Classes responsáveis por ler e guardar JSON no `localStorage`.
2. **Camada de Lógica (Controller):** Funções que validam regras (ex: verificar senha, impedir agendamento duplicado).
3. **Camada de Apresentação (View):** Funções que manipulam o DOM (HTML) para mostrar tabelas e formulários.

5.4 Organização do Código (Classes e Atributos)

Exemplo da estrutura de dados (JSON):

- **Classe** `Paciente`**:**
  - *Atributos:* `id`, `nome`, `cpf`, `telefone`, `historico_clinico`.
- **Classe** `Usuario`**:**
  - *Atributos:* `login`, `senha`, `perfil`.
  - *Métodos:* `+autenticar()`.
- **Classe** `Agendamento`**:**
  - *Atributos:* `id`, `data_hora`, `paciente_id`, `dentista_id`.

5.5 Relacionamentos

- 1 **Dentista** atende muitos **Pacientes**.
- 1 **Paciente** possui muitos **Agendamentos**.

5.6 Diagrama da Arquitetura

```
classDiagram
    class Usuario {
        +id: int
        +login: string
        +senha: string
        +perfil: string
        +autenticar()
    }

    class Paciente {
        +id: int
        +nome: string
        +cpf: string
        +telefone: string
        +historico_clinico: string
        +cadastrar()
        +editar()
    }
```

```
class Agendamento {
    +id: int
    +data_hora: datetime
    +paciente_id: int
    +dentista_id: int
    +status: string
    +agendar()
    +cancelar()
}

class Pagamento {
  +id: int
  +valor: float
  +data: datetime
  +status: string
  +registar()
}

Usuario "1" -- "*" Agendamento : realiza
Paciente "1" -- "*" Agendamento : possui
Agendamento "1" -- "1" Pagamento : gera
```

6 Análise de Riscos (Bónus Conceitual)

Aponte riscos de acoplamento e proponha soluções.

- **Risco:** Alto acoplamento entre a interface (HTML) e a lógica (JS). Se mudar o ID de um botão, o código para de funcionar.
- **Risco:** Segurança de dados no cliente.
    - *Solução:* O uso é restrito a computadores confiáveis da clínica, pois a base de dados reside no navegador.