**Documentação do Projeto: Sistema de Gestão Hospitalar**

**Versão:** 1.0 **Data:** 1 de agosto de 2025

**Autores:** Minoru Yamanaka, Lucimara Dias, Carlos Gonçalves

**1. Objetivo e Introdução**

Este documento detalha a estrutura, funcionalidades e regras de negócio do **Sistema de Gestão Hospitalar**. Trata-se de uma aplicação web desenvolvida para centralizar e otimizar a administração de informações essenciais em uma unidade de saúde, como hospitais ou clínicas.

O **objetivo principal** do projeto é substituir processos manuais ou descentralizados por uma plataforma única e segura, garantindo maior eficiência na gestão de dados de pacientes, médicos e consultas, além de assegurar a integridade e a rastreabilidade das informações.

**2. Briefing do Projeto**

* **Problema:** A gestão de informações hospitalares sem uma ferramenta centralizada pode levar a ineficiências, perda de dados, dificuldade no agendamento e falta de um histórico consolidado dos pacientes.
* **Solução Proposta:** Desenvolver um sistema web com acesso restrito por login e senha, onde administradores possam realizar o cadastro e o gerenciamento completo de pacientes, médicos, convênios associados e o agendamento de consultas.
* **Público-Alvo:** Usuários administrativos responsáveis pela organização e controle dos dados da instituição de saúde.
* **Tecnologias Base:** A aplicação é construída utilizando PHP para o back-end, MySQL para o banco de dados e HTML/CSS para a interface do usuário [cite: Conexao.php, minoru\_gestao\_hospitalar.sql, style.css].

**3. Escopo do Projeto**

**Dentro do Escopo:**

* Todas as funcionalidades descritas nos Requisitos Funcionais (Autenticação e CRUDs de Pacientes, Médicos, Consultas, Convênios e Endereços).
* Interface web para administração dos dados.

**Fora do Escopo:**

* Portal de acesso para pacientes ou médicos.
* Módulo financeiro ou de faturamento de convênios.
* Módulo de controle de estoque de medicamentos ou suprimentos.
* Integração com sistemas de exames laboratoriais.
* Notificações por e-mail ou SMS para lembretes de consulta.

**4. Regras de Negócio**

* **RN01:** O acesso ao sistema de gerenciamento é restrito e requer autenticação prévia [cite: index.php, usuario.php].
* **RN02:** O e-mail de um usuário deve ser único no sistema [cite: minoru\_gestao\_hospitalar.sql, cadastro.php].
* **RN03:** Todos os campos dos formulários de cadastro são, em sua maioria, obrigatórios para garantir a consistência dos dados.
* **RN04:** Uma consulta só pode ser agendada se o paciente e o médico selecionados já existirem no banco de dados [cite: minoru\_gestao\_hospitalar.sql].
* **RN05:** Um convênio ou endereço só pode ser cadastrado se estiver associado a um paciente existente [cite: minoru\_gestao\_hospitalar.sql].
* **RN06:** Ao excluir um paciente, todos os seus registros associados (consultas, convênios e endereços) são automaticamente excluídos em cascata para manter a integridade do banco de dados (ON DELETE CASCADE) [cite: minoru\_gestao\_hospitalar.sql].

**5. Requisitos Funcionais (RF)**

* **RF01 - Autenticação de Usuário:**
  + O sistema deve permitir que um usuário se cadastre com nome, e-mail e senha [cite: cadastro.php].
  + O sistema deve permitir que um usuário realize login usando e-mail e senha [cite: login.php].
  + O sistema deve validar os dados de login e conceder acesso apenas a usuários cadastrados [cite: login.php].
  + O sistema deve ter uma funcionalidade de logout que encerre a sessão do usuário [cite: logout.php].
  + O sistema deve restringir o acesso às páginas de gerenciamento apenas para usuários autenticados [cite: index.php].
* **RF02 - Gerenciamento de Pacientes:**
  + O sistema deve permitir o cadastro de novos pacientes com nome, sobrenome, data de nascimento, sexo e data de cadastro [cite: form\_paciente.php].
  + O sistema deve permitir a visualização de todos os pacientes cadastrados em uma lista [cite: lista\_paciente.php].
  + O sistema deve permitir a alteração dos dados de um paciente existente [cite: form\_paciente.php, PacienteDAO.php].
  + O sistema deve permitir a exclusão de um paciente [cite: excluirPaciente.php].
* **RF03 - Gerenciamento de Médicos:**
  + O sistema deve permitir o cadastro de novos médicos com nome, especialidade e CRM [cite: form\_medicos.php].
  + O sistema deve listar todos os médicos cadastrados [cite: lista\_medicos.php].
  + O sistema deve permitir a alteração dos dados de um médico existente [cite: form\_medicos.php, MedicoDAO.php].
  + O sistema deve permitir a exclusão de um médico [cite: excluirMedicos.php].
* **RF04 - Gerenciamento de Consultas:**
  + O sistema deve permitir agendar uma nova consulta, associando um paciente e um médico a uma data, hora e especialidade [cite: form\_consultas.php].
  + O sistema deve listar todas as consultas agendadas [cite: lista\_consultas.php].
  + O sistema deve permitir a alteração dos dados de uma consulta [cite: form\_consultas.php, ConsultaDAO.php].
  + O sistema deve permitir a exclusão (cancelamento) de uma consulta [cite: excluirConsultas.php].
* **RF05 - Gerenciamento de Convênios e Endereços:**
  + O sistema deve permitir o cadastro de convênios, associando-os a um paciente [cite: form\_convenio.php].
  + O sistema deve permitir o cadastro de múltiplos endereços, associando-os a um paciente [cite: form\_endereco.php].
  + O sistema deve permitir a visualização, alteração e exclusão de convênios e endereços [cite: lista\_convenio.php, lista\_endereco.php, excluirConvenio.php, excluirEndereco.php].

**6. Requisitos Não Funcionais (RNF)**

* **RNF01 - Segurança:**
  + As senhas dos usuários devem ser armazenadas no banco de dados de forma criptografada (password\_hash) [cite: UsuarioDAO.php, cadastro.php].
  + A sessão do usuário deve ser mantida por um token seguro gerado aleatoriamente [cite: login.php, UsuarioDAO.php].
* **RNF02 - Usabilidade e Interface:**
  + A interface deve ser intuitiva e de fácil navegação, com um menu consistente em todas as páginas.
  + O sistema deve ser responsivo, adaptando-se a diferentes tamanhos de tela (desktop e mobile), conforme definido em style.css [cite: style.css].
* **RNF03 - Desempenho:**
  + As consultas ao banco de dados devem ser otimizadas para um tempo de resposta rápido.
  + A conexão com o banco de dados deve ser gerenciada de forma eficiente, utilizando o padrão Singleton para evitar múltiplas conexões desnecessárias [cite: Conexao.php].
* **RNF04 - Manutenibilidade:**
  + O código deve ser organizado em camadas (Apresentação, Modelo, DAO), separando as responsabilidades para facilitar futuras manutenções [cite: Paciente.php, PacienteDAO.php, lista\_paciente.php].

**7. Diagrama de Casos de Uso**

* **Ator Principal:** Usuário Administrativo
* **Casos de Uso:**
  + **Autenticar-se:** Inclui as ações de Login, Logout e Cadastro.
  + **Gerenciar Pacientes:**
    - Cadastrar Paciente
    - Listar Pacientes
    - Editar Paciente
    - Excluir Paciente
  + **Gerenciar Médicos:**
    - Cadastrar Médico
    - Listar Médicos
    - Editar Médico
    - Excluir Médico
  + **Gerenciar Consultas:**
    - Agendar Consulta
    - Listar Consultas
    - Editar Consulta
    - Cancelar Consulta
  + **Gerenciar Dados Associados:**
    - Gerenciar Convênios
    - Gerenciar Endereços

**8. Modelo Conceitual - DER (Diagrama de Entidade-Relacionamento)**

Este diagrama representa as principais entidades de dados e como elas se relacionam.

* **Entidades:**
  + USUARIO (id, nome, email, senha, token)
  + PACIENTE (id, nome, sobrenome, data\_nascimento, sexo, data\_cadastro)
  + MEDICO (id, nome, especialidade, crm)
  + CONSULTA (id, data, horas, especialidade)
  + CONVENIO (id, empresa, cnpj, telefone, email)
  + ENDERECO (id, logradouro, bairro, cidade, estado)
* **Relacionamentos:**
  + Um PACIENTE pode ter **muitas** CONSULTAS. Uma CONSULTA pertence a apenas **um** PACIENTE (Relacionamento 1-N).
  + Um MEDICO pode realizar **muitas** CONSULTAS. Uma CONSULTA é realizada por apenas **um** MEDICO (Relacionamento 1-N).
  + Um PACIENTE pode ter **muitos** CONVENIOS (Relacionamento 1-N).
  + Um PACIENTE pode ter **muitos** ENDERECOS (Relacionamento 1-N).

**9. Diagrama de Classe - MER (Modelo Entidade-Relacionamento de Classes)**

Este diagrama descreve a estrutura das classes PHP que representam as entidades e suas operações de acesso a dados.

* **Classe: Usuario** [cite: Usuario.php]
  + Atributos: -id, -nome, -email, -senha, -token
  + Métodos: +\_\_construct(), +getId(), +getNome(), +getEmail(), +getSenha(), +getToken()
* **Classe: UsuarioDAO** [cite: UsuarioDAO.php]
  + Métodos: +create(Usuario), +updateToken(id, token), +getByEmail(email): Usuario, +getByToken(token): Usuario
* **Classe: Paciente** [cite: Paciente.php]
  + Atributos: -id, -nome, -sobrenome, -data\_nascimento, -sexo, -data\_cadastro
  + Métodos: +\_\_construct(), +get/set... para todos os atributos.
* **Classe: PacienteDAO** [cite: PacienteDAO.php]
  + Métodos: +create(Paciente), +getAll(): Paciente[], +getById(id): Paciente, +update(Paciente), +excluir(id)
* **Classe: Medico** (definida como Medicos no arquivo) [cite: Medico.php]
  + Atributos: -id, -nome, -especialidade, -crm
  + Métodos: +\_\_construct(), +get/set... para todos os atributos.
* **Classe: MedicoDAO** [cite: MedicoDAO.php]
  + Métodos: +create(Medico), +getAll(): Medico[], +getById(id): Medico, +update(Medico), +excluir(id)
* *(O mesmo padrão de Classes Model e DAO se aplica para Consulta, Convenio e Endereco)*.