**Modelo de Projeto – Desenvolvimento Low-Code**

#### 1) Identificação do Projeto Nome do projeto: MediConnect Pro Sponsor / Área demandante: Projeto Acadêmico Gestor do Projeto (PO): Luzaydha Sousa Scrum Master / Facilitador: Autogerenciado Plataforma Low-Code: lovable (com apoio exploratório em BoxBlack e Gemini) e Xampp. Versão do Documento: Data: v1.0 / 15-09-2025

#### 2) Propósito e Objetivos

**Propósito:** Desenvolver um sistema multifuncional para uma clínica fictícia, demonstrando habilidades em criação de aplicações low-code. O sistema, batizado de MediConnect Pro, centralizará canais de atendimento e automatizará processos internos, utilizando a plataforma lovable.

**Objetivos SMART:**

* + **Semana 1:** Criar uma lista de tarefas funcional com marcação de status.
  + **Semana 2:** Implementar um formulário de cadastro com edição e exclusão de pacientes.
  + **Semana 3:** Integrar uma API pública e exibir dados em um dashboard para o perfil de gerente.
  + **Semana 4:** Configurar a automação de notificações para pacientes, simulando agendamentos.
  + **Semana 5:** Teste de inclusão de novos cadstro no banco da dados.

#### 3) Escopo

**Escopo incluído:**

* + Lista de tarefas para gestão interna de atendimentos.
  + Formulário de cadastro de pacientes com as informações de nome, e-mail, telefone, CPF, data de nascimento e endereço.
  + Páginas de login e dashboard dinâmicos para diferentes papéis de usuário (paciente, atendente, gerente).
  + Automação de envio de mensagens de confirmação de agendamento.
  + Módulo de histórico de conversas e notas internas para atendentes.
  + Mini loja virtual com catálogo e carrinho de compras.

**Fora de escopo:**

* + Integração com sistemas corporativos legados.
  + Aplicativo mobile nativo (foco em interface web responsiva).
  + Sistema de pagamentos real.
  + Integrações com APIs de redes sociais (apenas simulação).

**Entregáveis:**

* Aplicação funcional na plataforma lovable com os cinco módulos.
* Páginas web (HTML/CSS/JS) para as interfaces de usuário (paciente, atendente, gerente) e login.
* Documentação básica de uso e arquitetura da solução.

#### 4) Partes Interessadas (Stakeholders)

**Sponsor:** Projeto Acadêmico

**PO:** Luzaydha Sousa Santos

**TI/Segurança:** Suporte da própria plataforma lovable e servidor local XAMPP.

**Usuários-chave:** Avaliadores, colegas de curso e usuários de teste (paciente, atendente, gerente).

#### 5) Requisitos

**Requisitos Funcionais (RF):**

* **RF01 (Login e Perfis):** O sistema deve permitir o login de usuários com diferentes papéis (paciente, atendente, gerente) e redirecioná-los para a página correspondente ao seu perfil.
* **RF02 (Cadastro de Pacientes):** O formulário de cadastro deve coletar nome, e-mail, telefone, CPF, data de nascimento e endereço.
* **RF03 (Gestão de Pacientes):** O perfil de atendente deve poder visualizar, agendar e interagir com conversas de pacientes, além de registrar notas internas.
* **RF04 (Dashboard):** O perfil de gerente deve ter acesso a um dashboard com métricas em tempo real e um resumo dos canais de atendimento.
* **RF05 (Comunicação Paciente):** O perfil de paciente deve visualizar seu histórico de agendamentos, lembretes, resultados de exames e enviar mensagens para a clínica.

**Requisitos Não Funcionais (RNF):**

* **RNF01 (Usabilidade):** A interface deve ser intuitiva e responsiva para diferentes tamanhos de tela (desktop e mobile).
* **RNF02 (Segurança):** O sistema deve utilizar senhas hasheadas e sessões para autenticação de usuários.
* **RNF03 (Desempenho):** As páginas devem carregar em no máximo 3 segundos, exceto em caso de problemas de conexão local.

#### 6) Arquitetura e Desenho da Solução

* **Camada de UI:** Construída na plataforma lovable com componentes visuais como cards, formulários e botões, além de interfaces personalizadas usando HTML, CSS (Tailwind CSS) e JavaScript.
* **Camada de Lógica:** Workflows internos da lovable para salvar e editar dados, além de scripts PHP para autenticação e conexão com o banco de dados.
* **Camada de Dados:** Banco de dados MySQL local via XAMPP, com uma tabela central de usuarios para gerenciar os perfis.
* **Integrações:** Simulação de integração com API pública no dashboard do gerente e com canais de comunicação (WhatsApp, Instagram, E-mail).
* **Identidade:** Sistema de login com validação de e-mail e senha, e redirecionamento baseado no papel do usuário (role).

#### 7) Segurança, Governança e Compliance

* **Controle de Acesso:** Perfis de usuário com permissões distintas (paciente, atendente, gerente).
* **Proteção de Dados:** Senhas armazenadas com password\_hash e criptografia em trânsito via HTTPS (simulada em ambiente de desenvolvimento).
* **LGPD:** Termo de consentimento simulado no cadastro do paciente.
* **Auditoria:** O sistema deve registrar logs de criação e edição de registros (futura implementação).
* **Backups:** Simulados via infraestrutura lovable e rotinas locais no XAMPP.

#### 8) Metodologia e Cronograma

**Abordagem:** Desenvolvimento incremental com foco em entregas a cada "exercício" ou semana.

**Marcos (milestones):**

* **Semana 1:**
* Instalação e configuração do ambiente XAMPP e MySQL.
* Criação do banco de dados e da tabela usuarios.
* Desenvolvimento do login.php e das páginas login\_clinica.html, paciente.html, atendente.html e gerente.html.
* Criação da lista de tarefas e interface de atendente.
* **Semana 2:**
  + - Implementação do formulário de cadastro com todos os campos de paciente.
    - Ajustes nos scripts de PHP para registrar novos usuários.
    - **Semana 3:**
    - Criação do dashboard para o perfil de gerente, exibindo métricas fictícias e dados de canais.
    - Simulação da integração com uma API pública.
  + **Semana 4:**
* Desenvolvimento da funcionalidade de agendamento na interface do atendente.
* Configuração do envio de notificações simuladas para o paciente via mensagem interna.
* **Semana 5:**
  + - Criação de um módulo de mini loja virtual para o paciente com catálogo, carrinho e envio de pedidos por e-mail (simulado).

#### 9) Testes e qualidade

**Níveis de Testes:**

* **Testes manuais por funcionalidade:** Verificação de cada módulo (login, cadastro, dashboard).
* **Testes de integração:** Validação da comunicação entre o front-end (HTML/JS) e o back-end (PHP/MySQL).

**Casos de Teste (Exemplos):**

* **CT-L-01 (Login de Sucesso):**
  + - **Cenário:** O usuário atendente@clinica.com insere a senha atendente123 e clica em "Entrar".
    - **Resultado Esperado:** O sistema redireciona para atendente.html.
  + **CT-L-02 (Login Falho):**
    - **Cenário:** O usuário insere um e-mail válido, mas uma senha incorreta.
    - **Resultado Esperado:** O sistema exibe a mensagem "Email ou senha inválidos.".
  + **CT-C-01 (Cadastro Válido):**
    - **Cenário:** Um novo paciente preenche todos os campos obrigatórios e clica em "Cadastrar".
    - **Resultado Esperado:** O sistema cria um novo registro na tabela usuarios e exibe uma mensagem de sucesso.

#### 10) KPIs e Métricas de Sucesso

* **Adoção:** Número de usuários que acessam cada módulo.
* **Eficiência:** Tempo médio para concluir tarefas de atendimento.
* **Qualidade:** Quantidade de bugs reportados durante os testes.
* **Satisfação:** Feedback de usuários de teste (avaliadores, colegas).