Análise da Proposta de ERP Web para Laboratório de Prótese Dentária

1. Introdução

O projeto propõe a criação de um ERP web simples, funcional e amigável para o laboratório LR Prótese Dental. O sistema será 100% web, acessível via navegador, e suportará cerca de 40 usuários simultâneos. O objetivo principal é otimizar o fluxo de trabalho do laboratório, desde o cadastro de clientes até a entrega das próteses, com foco em eficiência e facilidade de uso. Será uma solução de código aberto, construída com tecnologias amplamente adotadas pela comunidade para facilitar manutenção e contribuições via GitHub.

2. Módulos Propostos

O sistema será composto pelos seguintes módulos:

2.1. Cadastro de Clientes e Pacientes

- Cadastro de clientes (dentistas ou clínicas) e pacientes associados.
- Vinculação de cada paciente a um cliente responsável.
- Inclusão de informações pessoais (nome, contato, endereço) e observações.
- Filtros e buscas eficientes por nome de paciente, clínica ou dentista.
- Visualização do histórico de ordens de serviço por cliente/paciente.
- Controle de permissões: administradores podem adicionar/editar clientes; técnicos apenas consultam.

2.2. Ordem de Serviço / Ficha de Trabalho

- Módulo central para registro e acompanhamento de trabalhos protéticos.

- Criação de OS com dados como cliente, paciente, tipo de prótese, materiais, data de entrada e prazo.
- Identificador único para cada OS.
- Status da OS: recebido, em produção, finalizado, entregue.
- Anexar arquivos relevantes (radiografias, modelos digitais, fotos).
- Geração e impressão de ficha de trabalho com dados principais e checklist de etapas.
- Atualização de status e edição de informações da OS por usuários autorizados.
- Visão em tempo real de todas as ordens em aberto e seu status.

2.3. Controle de Produção

- Modelagem das etapas do processo produtivo de cada prótese (desenho, fundição, acabamento, etc.).
- Configuração de fluxo de etapas específico por tipo de prótese.
- Atribuição de cada etapa a um técnico ou funcionário responsável.
- Registro de início e término de etapa, com tempo gasto.
- Acompanhamento do progresso em tempo real e identificação de gargalos.
- Painel (Kanban ou lista) das ordens em produção, mostrando etapa e responsável.
- Geração de estatísticas de produção (tempo médio, produtividade por técnico).
- Registro de movimentação no processo produtivo para rastreabilidade e garantia de qualidade.

2.4. Financeiro

- Faturamento e controle de receitas/despesas.
- Atribuição de valor a cada OS.
- Geração de relatórios de faturamento por cliente.
- Registro de contas a receber (pagamentos de clientes) e a pagar (fornecedores, despesas).
- Fluxo de caixa consolidado.

- Identificação de faturas vencidas (inadimplência) e alertas.
- Integração com cadastro de clientes e ordens de serviço.

2.5. Controle de Estoque

- Gestão de materiais e insumos (resinas, ligas metálicas, porcelanas, etc.).
- Cadastro de materiais com descrição, código, fornecedor, unidade, quantidade, mínimo, localização, validade.
- Registro de entrada de estoque (novas aquisições).
- Registro de saída/consumo de materiais (baixa automática ao vincular a OS).
- Alertas de estoque mínimo para reposição.
- Relatórios de estoque (materiais abaixo do mínimo, histórico de movimentações, valorização).

2.6. Agenda e Comunicação Interna

- Agenda de entregas e coletas (vinculada a OS).
- Agenda de produção (visualização do cronograma de trabalho por técnico).
- Quadro de avisos digital para comunicados internos.
- Sistema simples de mensagens internas ou comentários nas OS.
- Notificações por e-mail ou push para eventos críticos (prazo expirando, estoque crítico).

2.7. Relatórios e Dashboards

- Relatórios gerenciais e dashboards interativos.
- Relatórios de produção (ordens concluídas, tempo médio, desempenho por técnico).
- Relatórios financeiros (faturamento, DRE, contas a receber/pagar, fluxo de caixa, inadimplência).
- Relatórios de clientes (ranking, histórico de pedidos).
- Relatórios de estoque (materiais mais utilizados, itens críticos).

- Dashboard visual com gráficos e KPIs (ordens em produção, atrasadas, entregas do dia, faturamento, estoque crítico).
- Filtros por período e exportação (PDF/Excel).

3. Arquitetura, Tecnologias e Implementação

3.1. Arquitetura

- Arquitetura modular com separação de front-end e back-end via API RESTful.
- Permite expansão e manutenção independentes.

3.2. Tecnologias Sugeridas

- **Back-end:** JavaScript/Node.js (Express), Python (Django ou Flask), ou PHP (Laravel).
- **Banco de Dados:** Relacional (PostgreSQL ou MySQL/MariaDB).
- **Front-end:** React, Angular ou Vue.js (SPA - Single Page Application).
- **Componentes Visuais:** Bootstrap, Material UI ou Ant Design.

3.3. Segurança

- Sistema de usuários com diferentes perfis de acesso (Administrador, Financeiro, Técnico, Recepção/Atendimento).
- Validação de permissões em cada endpoint da API.
- Autenticação: JSON Web Tokens (JWT) ou cookies/sessão tradicional.
- Armazenamento seguro de dados sensíveis (senhas com hash forte).
- Comunicação apenas via HTTPS (SSL/TLS).

3.4. Desenvolvimento e Documentação

- Projeto estruturado no repositório GitHub (diretórios /backend e /frontend).
- Controle de versão (Git).
- Arquivo README.md detalhado com:

    - Pré-requisitos.
    - Instruções de instalação (clonar, configurar variáveis de ambiente, instalar dependências, inicializar DB).
    - Execução em desenvolvimento (rodar back-end e front-end).
    - Instruções de build e deploy (sugestões de hospedagem, Docker).
    - Uso básico (acesso, criação de usuário admin).
- Boas práticas de codificação e documentação (comentários no código, API docs).

3.5. Desempenho e Escalabilidade

- Projetado para suportar 40 usuários simultâneos, com possibilidade de crescimento.
- Uso eficiente de consultas ao banco e estrutura de dados escalável.

3.6. Qualidade e Confiabilidade

- Priorização da experiência do usuário e confiabilidade.
- Funcionalidades integradas para evitar retrabalho.
- Visão unificada do negócio.
- Baseado em melhores práticas e sistemas de mercado (LabFácil, onPrótese).
- Projeto consistente, de fácil manutenção e expansão, open source com documentação completa.

4. Referências

O documento faz referência a sistemas existentes como LabFácil (Urgtec) e onPrótese, além de um repositório GitHub de um ERP/CRM open-source (idurar/idurar-erp-crm) para embasar as funcionalidades e a arquitetura proposta.

Esta análise detalha os principais pontos da proposta, incluindo os módulos funcionais, a arquitetura técnica e as diretrizes de desenvolvimento. O próximo passo é identificar os requisitos e funcionalidades propostas de forma mais granular.

5. Requisitos e Funcionalidades Detalhadas

Com base na análise dos módulos, os requisitos e funcionalidades propostas podem ser detalhados da seguinte forma:

5.1. Cadastro de Clientes e Pacientes

- **RF001:** O sistema deve permitir o cadastro de clientes (dentistas ou clínicas) com nome, contato, endereço e observações.
- **RF002:** O sistema deve permitir o cadastro de pacientes, vinculando-os a um cliente (dentista responsável).
- **RF003:** O sistema deve permitir a busca e filtragem de clientes e pacientes por nome, clínica ou dentista.
- **RF004:** O sistema deve exibir o histórico de ordens de serviço associadas a cada cliente ou paciente.
- **RF005:** O sistema deve implementar controle de permissões para o cadastro de clientes e pacientes (ex: administradores podem editar, técnicos apenas consultar).

5.2. Ordem de Serviço (OS) / Ficha de Trabalho

- **RF006:** O sistema deve permitir a criação de novas Ordens de Serviço (OS) com os seguintes campos: cliente solicitante, paciente, tipo de prótese, materiais, data de entrada e prazo estimado de entrega.
- **RF007:** Cada OS deve ter um identificador único.
- **RF008:** O sistema deve permitir a atualização do status da OS (recebido, em produção, finalizado, entregue).
- **RF009:** O sistema deve permitir anexar arquivos (radiografias, modelos digitais, fotos) a uma OS.
- **RF010:** O sistema deve gerar e permitir a impressão de uma ficha de trabalho para cada OS, contendo dados principais e um checklist de etapas.
- **RF011:** O sistema deve fornecer uma visão em tempo real de todas as OS em aberto e seus respectivos status.

5.3. Controle de Produção

- **RF012:** O sistema deve permitir a configuração de etapas de produção para cada tipo de prótese.
- **RF013:** O sistema deve permitir a atribuição de etapas de produção a técnicos ou funcionários responsáveis.
- **RF014:** O sistema deve registrar o início, término e tempo gasto em cada etapa de produção por OS.
- **RF015:** O sistema deve exibir um painel (Kanban ou lista) das ordens em produção, mostrando a etapa atual e o responsável.
- **RF016:** O sistema deve gerar estatísticas de produção, como tempo médio por tipo de prótese e produtividade por técnico.
- **RF017:** O sistema deve registrar todas as movimentações no processo produtivo para rastreabilidade.

5.4. Financeiro

- **RF018:** O sistema deve permitir a atribuição de um valor a cada OS para faturamento.
- **RF019:** O sistema deve gerar relatórios de faturamento agrupando OS por cliente em um período.
- **RF020:** O sistema deve registrar contas a receber (pagamentos de clientes) e contas a pagar (fornecedores, despesas).
- **RF021:** O sistema deve apresentar um fluxo de caixa consolidado.
- **RF022:** O sistema deve identificar faturas vencidas (inadimplência) e gerar alertas.
- **RF023:** O sistema deve integrar o módulo financeiro com o cadastro de clientes e ordens de serviço.

5.5. Controle de Estoque

- **RF024:** O sistema deve permitir o cadastro de materiais e insumos com descrição, código, fornecedor, unidade de medida, quantidade em estoque, quantidade mínima, localização e data de validade.

- **RF025:** O sistema deve registrar a entrada de novas aquisições de materiais no estoque.
- **RF026:** O sistema deve registrar a saída/consumo de materiais, com baixa automática ao vincular a uma OS.
- **RF027:** O sistema deve gerar alertas quando a quantidade de um item em estoque atingir o limite mínimo.
- **RF028:** O sistema deve gerar relatórios de estoque (materiais abaixo do mínimo, histórico de movimentações, valorização).

5.6. Agenda e Comunicação Interna

- **RF029:** O sistema deve possuir uma agenda para agendamento de entregas e coletas, vinculada a OS.
- **RF030:** O sistema deve apresentar uma visão do cronograma de produção por técnico.
- **RF031:** O sistema deve ter um quadro de avisos digital para comunicados internos.
- **RF032:** O sistema deve permitir mensagens internas ou comentários nas OS.
- **RF033:** O sistema deve enviar notificações (e-mail ou push) para eventos críticos (prazo expirando, estoque crítico).

5.7. Relatórios e Dashboards

- **RF034:** O sistema deve gerar relatórios de produção (quantidade de ordens concluídas, tempo médio, desempenho por técnico).
- **RF035:** O sistema deve gerar relatórios financeiros (faturamento, DRE, contas a receber/pagar, fluxo de caixa, inadimplência).
- **RF036:** O sistema deve gerar relatórios de clientes (ranking, histórico de pedidos).
- **RF037:** O sistema deve gerar relatórios de estoque (materiais mais utilizados, itens críticos).
- **RF038:** O sistema deve apresentar um dashboard visual com gráficos e KPIs (ordens em produção, atrasadas, entregas do dia, faturamento, estoque crítico).

- **RF039:** Todos os relatórios e gráficos devem ser filtráveis por período e exportáveis (PDF/Excel).

5.8. Requisitos Técnicos e de Segurança

- **RT001:** O sistema deve ser desenvolvido com arquitetura modular (front-end e back-end separados por API RESTful).
- **RT002:** O back-end deve ser implementado em Node.js (Express), Python (Django/Flask) ou PHP (Laravel).
- **RT003:** O banco de dados deve ser relacional (PostgreSQL ou MySQL/MariaDB).
- **RT004:** O front-end deve ser uma Single Page Application (SPA) utilizando React, Angular ou Vue.js.
- **RT005:** A interface deve ser responsiva e utilizar bibliotecas de componentes visuais (Bootstrap, Material UI, Ant Design).
- **RT006:** O sistema deve implementar autenticação de usuários com diferentes perfis de acesso (Administrador, Financeiro, Técnico, Recepção/Atendimento).
- **RT007:** As permissões devem ser validadas em cada endpoint da API.
- **RT008:** Dados sensíveis (senhas) devem ser armazenados com hash forte.
- **RT009:** A comunicação entre cliente e servidor deve ser via HTTPS (SSL/TLS).
- **RT010:** O projeto deve ser versionado no GitHub com estrutura de diretórios clara (/backend, /frontend).
- **RT011:** O repositório deve conter um arquivo README.md detalhado com instruções de pré-requisitos, instalação, execução em desenvolvimento, build e deploy, e uso básico.
- **RT012:** O sistema deve ser capaz de suportar 40 usuários simultâneos e ser escalável para crescimento futuro.
- **RT013:** O sistema deve priorizar a experiência do usuário e a confiabilidade, com funcionalidades integradas.

Esta seção detalha os requisitos funcionais e não funcionais do sistema, fornecendo uma base sólida para o desenvolvimento. O próximo passo é elaborar um relatório final de análise e entregar os resultados.