ChatGPT

Estrutura de Pastas da Plataforma Leitícia

A estrutura de diretórios abaixo organiza o projeto **Plataforma Leitícia** de forma modular, atendendo a todos os requisitos mencionados. Está separada por áreas (pública, parceiros, comunidade, chat, gestores, admin), com divisões claras entre conteúdo público, seções restritas (que exigem login) e administração. Também inclui pastas para arquivos estáticos (CSS, JS, imagens), scripts PHP reutilizáveis, APIs segmentadas por funcionalidade e configurações. Cada pasta e arquivo exemplificado é comentado com sua finalidade para facilitar o rápido desenvolvimento em dois dias, conforme planejado pela equipe.

Área Pública e Diretórios Globais

- `plataforma-leiticia/` – **Diretório raiz do projeto**, contendo todo o código-fonte da plataforma.
- `index.php` – Página inicial **pública** (landing page informativa inspirada na home do iFood, exibindo informações da plataforma e um botão discreto de acesso/login no topo direito).
- `assets/` – **Arquivos estáticos públicos** (CSS, JS e imagens) usados em toda a plataforma.
  - `css/` – Folhas de estilo CSS compartilhadas ou globais do site.
  - `styles.css` – *Folha de estilo global* contendo os estilos comuns a todas as páginas da plataforma (cores, fontes, layout base).
  - `js/` – Scripts JavaScript globais da plataforma (sem frameworks, apenas JS puro ou bibliotecas open-source necessárias).
  - `main.js` – *Script JS global* com funcionalidades gerais (por exemplo, inicialização do widget de chat, manipuladores de menu, utilitários).
  - `images/` – Imagens utilizadas na interface pública do site (logos, ícones, banners etc.).
  - `logo.png` – *Exemplo de imagem:* logotipo da plataforma exibido na página inicial e no cabeçalho.
- `includes/` – **Scripts PHP reutilizáveis** em diversas partes do site (como cabeçalho, rodapé, conexão de banco de dados, funções utilitárias).
  - `db_connect.php` – Script de conexão ao **banco de dados MySQL**, instanciando a conexão (usado por demais scripts que precisam acessar o banco).
  - `header.php` – Script que gera o **cabeçalho** HTML comum de todas as páginas (contém as tags `<head>` com imports de CSS/JS, e o topo do `<body>` com logo, menu de navegação e inclusão do widget de chat de suporte).
  - `footer.php` – Script de **rodapé** HTML comum (fecha tags abertas no header, scripts finais comuns, copyrights etc.).
  - `auth.php` – Script de **autenticação** e controle de acesso. Verifica se o usuário está logado e possui permissão adequada, redirecionando ou barrando acesso se necessário. É incluído no topo das páginas **restritas** (parceiros, comunidade, gestores, admin).
- `config/` – **Configurações** globais do sistema.
  - `config.php` – Arquivo central de configuração com constantes e variáveis de ambiente (credenciais do MySQL, configurações de e-mail, chaves de API, e outras definições). *(Deve ser mantido fora do acesso público direto por segurança, contendo apenas código PHP de configuração.)*

- `logs/` – (Opcional) **Logs** do sistema gravados em arquivo.
  - `sistema.log` – *Exemplo de log:* arquivo texto onde o sistema pode registrar eventos importantes, erros ou atividades para auditoria. (A área admin terá página para visualizar esses logs.)
- `README.md` – Documentação do projeto, com instruções de instalação, configuração e informações para a equipe de desenvolvimento.
- `vendor/` – (Opcional) Dependências de terceiros instaladas via Composer, caso sejam usadas bibliotecas PHP externas. **Observação:** Somente bibliotecas/plugins **gratuitos e de código aberto** são utilizados na plataforma (por exemplo, PHPMailer para email, ou bibliotecas de gráficos JS como Chart.js em `assets/js`).

Área de Parceiros (`partner/`)

- `partner/` – **Portal dos parceiros** (restaurantes/lojas), semelhante à gestão de loja do iFood. **Acesso restrito:** requer login do parceiro.
- `login.php` – Página de **login do parceiro**, com formulário de acesso para restaurantes/lojas entrarem no seu painel.
- `index.php` – **Dashboard do parceiro** após login, página inicial da área do parceiro com resumo da loja (pedidos em andamento, estatísticas rápidas, notificações).
- `pedidos.php` – Página de **gerenciamento de pedidos/entregas** do parceiro. Lista pedidos atuais, status de cada entrega e permite atualizar o andamento (similar ao que um restaurante vê no iFood).
- `configuracoes.php` – Página de **configurações da loja** (perfil do parceiro): permite editar dados da loja, horários de funcionamento, detalhes de contato, opções de entrega etc.
- *(Outras páginas conforme necessidade: por exemplo,* `menu.php` *para gerenciamento de itens/produtos da loja,* `relatorios.php` *para histórico de vendas/pedidos, etc.)*
- `css/` – Folhas de estilo **específicas da área de parceiros**.
  - `partner.css` – Estilos CSS do portal do parceiro (ex.: layout do dashboard de parceiros, cores distintas para diferenciar da área pública).
- `js/` – Scripts **específicos da área de parceiros**.
  - `partner.js` – Lógica JavaScript para funcionalidades do parceiro (ex.: atualização dinâmica da lista de pedidos via AJAX, validações de formulários de menu/configurações).

Área de Comunidade (`community/`)

- `community/` – **Fórum e comunidades de usuários**, ao estilo das comunidades do Orkut. Usuários cadastrados podem criar comunidades, tópicos e postar comentários. **Acesso:** leitura possivelmente pública, mas ações (criar post, comentar) requerem login do usuário; moderadores (suporte) têm permissões adicionais.
- `index.php` – Página **principal da comunidade**, exibindo uma visão geral: lista de comunidades populares ou recentes, últimos tópicos em destaque, botão para criar nova comunidade.
- `criar_comunidade.php` – Página de **criação de comunidade/tema**. Formulário para o usuário definir nome, descrição e regras de uma nova comunidade.
- `comunidade.php` – Página de **comunidade específica**. Exibe detalhes da comunidade (nome, descrição, moderadores) e a lista de tópicos dentro dela. Pode incluir opção de entrar/sair da comunidade.
- `topico.php` – Página de **tópico específico** dentro de uma comunidade. Mostra a discussão: post inicial e comentários/respostas em sequência, além de um formulário para o usuário responder ou comentar.

- `postagem.php` – Script (ação) para **publicar uma nova postagem** ou tópico. Recebe dados do formulário (novo tópico ou resposta) e insere no banco de dados, depois redireciona de volta para a página apropriada (topico/comunidade).
- `moderacao.php` – **Painel de moderação** da comunidade. Permite aos moderadores (que são os agentes de suporte da plataforma) gerenciar conteúdo: visualizar denúncias de posts, remover tópicos ou comentários inadequados, banir ou advertir usuários dentro das comunidades. *(Esta página é acessível apenas a usuários com papel de moderador/suporte.)*
- *(Outras páginas se necessário: por exemplo,* `buscar.php` *para pesquisa de comunidades/tópicos, página de perfil do usuário da comunidade etc.)*
- `css/` – Estilos **específicos da área de comunidade** (visual do fórum).
  - `community.css` – CSS para layout de listas de tópicos, posts, perfil de comunidade, etc. (por ex.: estilização de postagens, quadros de comentários, destaque para posts do moderador).
- `js/` – Scripts **específicos do fórum/comunidade**.
  - `community.js` – Lógica JS para funcionalidades interativas da comunidade, como envio assíncrono de novos posts/comentários (AJAX), atualização em tempo real de conversas (long polling/WebSocket opcional), validação de formulário de criação de tópico, etc.

Chat de Atendimento (`chat/`)

- `chat/` – **Módulo de chat de suporte ao usuário**, disponível em todas as páginas do site para atendimento em tempo real. Os agentes deste chat também atuam como moderadores da comunidade.
- `chat_widget.php` – Componente de **janela de chat** embutido na interface. Este script PHP gera o HTML do pequeno widget de chat (por exemplo, um ícone ou janela minimizada no canto da tela) e é incluído pelo `includes/header.php` para estar presente em todas as páginas. Permite que o usuário inicie uma conversa de suporte a qualquer momento.
- `agendamento.php` – Página/formulário de **agendamento de atendimento**. Caso o usuário precise marcar um horário para conversar com um profissional de apoio (por exemplo, psicólogo, consultor), essa página possibilita escolher data e hora disponíveis e registra o agendamento para acompanhamento pelo suporte.
- `css/` – Estilos do **componente de chat**.
  - `chat.css` – CSS para personalizar a aparência da janela de chat, mensagens enviadas/recebidas, lista de conversas, etc., de forma que fique sempre visível de maneira discreta na página.
- `js/` – Scripts do **chat de suporte**.
  - `chat.js` – Lógica do chat em tempo real: pode incluir abertura/fechamento da janela de chat, envio de mensagens via AJAX ou WebSockets, atualização periódica de novas mensagens, notificações sonoras/visuais de resposta do agente, etc. Garante que o chat funcione fluidamente enquanto o usuário navega pelo site.

Área de Gestores (`manager/`)

- `manager/` – **Painel dos gestores** (administradores executivos), com dashboards analíticos e dados em tempo real sobre a plataforma. **Acesso restrito:** somente usuários com perfil de gestor.
- `index.php` – **Dashboard principal do gestor**, exibindo indicadores ao vivo: ex. número de usuários online, quantidade de pedidos no dia, métricas de desempenho, gráficos resumidos de doações e entregas, etc. (Atualiza automaticamente via AJAX para dados em tempo real).

- `relatorios.php` – Página de **relatórios detalhados**. Oferece filtragem de dados históricos e geração de relatórios aprofundados (por exemplo, desempenho mensal, comparativos semanais, gráficos de longo prazo).
- *(Outras páginas conforme necessidade: por exemplo,* `alertas.php` *para configurar alertas/metas,* `financeiro.php` *se houver necessidade de visão financeira para gestores, etc.)*
- `css/` – Estilos do **painel de gestão**.
  - `manager.css` – CSS específico para o layout de dashboards (por ex.: disposição de gráficos lado a lado, cores corporativas para dados, responsividade das tabelas e charts).
- `js/` – Scripts do **dashboard de gestores**.
  - `manager.js` – Lógica JS para atualizar os componentes do dashboard em tempo real. Pode usar bibliotecas abertas (como Chart.js ou D3.js) para renderizar gráficos interativos dos dados da plataforma. Inclui chamadas AJAX periódicas aos endpoints de analytics para obter dados atualizados sem recarregar a página.

Área Administrativa (`admin/`)

- `admin/` – **Portal administrativo** da plataforma, reunindo todos os controles do sistema. **Acesso restrito:** apenas administradores com as mais altas permissões.
- `index.php` – **Dashboard administrativo** com visão geral do sistema: atalhos para módulos (usuários, pedidos, finanças etc.), contadores de itens importantes (ex: total de usuários, total de pedidos do dia, alertas pendentes).
- `usuarios.php` – **Gestão de usuários e permissões**. Permite criar/editar/excluir usuários do sistema (parceiros, clientes, moderadores, gestores, admins), definir papéis/permissões, resetar senhas, etc. *(Inclui gerenciamento dos cadastros realizados na plataforma.)*
- `logs.php` – **Consulta de logs** do sistema. Exibe registros de atividades e eventos (logins, ações sensíveis, erros) para auditoria e acompanhamento de segurança.
- `faturamento.php` – **Gestão de faturamento**. Tela para acompanhar cobranças e pagamentos: exibe valores devidos de parceiros, histórico de pagamentos, gerar faturas/boletos, confirmar recebimentos (seguindo a política financeira da plataforma).
- `entregas.php` – **Gerenciamento de entregas global**. Lista e permite controle de todas as entregas realizadas através da plataforma (independente do parceiro), útil para suporte em casos de problemas na entrega ou análise de desempenho logístico.
- `doacoes.php` – **Gestão de doações** (caso a plataforma envolva doações). Lista doações recebidas, doadores, campanhas, permitindo administrar a destinação dessas doações e emitir recibos se necessário.
- `contratos.php` – **Gestão de contratos e acordos**. Módulo para armazenar e gerenciar contratos com parceiros ou prestadores de serviço, incluindo detalhes de vigência, valores e documentos digitalizados.
- `configuracoes.php` – **Configurações gerais** do sistema. Permite ajustar parâmetros globais da plataforma (por exemplo, ativar/desativar funcionalidades, atualizar textos institucionais, configurar integrações com serviços externos, gerenciamento de plugins).
- *(Outras páginas administrativas conforme necessidade: por exemplo, módulos de conteúdo e CMS, gerenciamento de categorias de produtos, FAQ da plataforma, etc.)*
- `css/` – Estilos específicos para a **interface admin**.
  - `admin.css` – CSS do painel administrativo, garantindo identidade visual distinta (para o administrador reconhecer que está no ambiente restrito) e layout adequado para tabelas de dados e formulários extensos.

- `js/` – Scripts exclusivos para funcionalidades do **admin**.
  - `admin.js` – Lógica JS para apoiar as telas administrativas (ex.: scripts de confirmação ao excluir algo, filtros dinâmicos em listagens, gráficos administrativos de desempenho financeiro ou de uso da plataforma, usando bibliotecas open-source se necessário).

APIs (Endpoints de Funcionalidade)

- `api/` – **Endpoints de API REST/AJAX**, separados por funcionalidade. Esses scripts PHP retornam dados (geralmente em JSON) e executam ações solicitadas via requisições assíncronas (AJAX) ou integrações externas, mantendo a separação do front-end e back-end. Cada endpoint verifica a autenticação/permissão conforme a área correspondente.
- `community_api.php` – Endpoint para operações da **comunidade**. Por exemplo, criação de um novo tópico ou comentário via AJAX, busca de posts, inscrição em comunidade, etc. (Valida se o usuário está logado e pode realizar a ação; interage com o banco e retorna sucesso/erro em formato JSON).
- `partner_api.php` – Endpoint para ações da **área de parceiros**. Ex: atualizar status de um pedido (quando parceiro marca um pedido como "enviado/entregue"), obter dados de vendas para gráficos do parceiro, etc. (Requer autenticação do parceiro).
- `chat_api.php` – Endpoint para o **chat de atendimento**. Gerencia envio e recebimento de mensagens via AJAX caso não se use WebSocket puro. Por exemplo, cliente envia mensagem -> salva no banco -> retorna confirmação; agente responde -> endpoint fornece novas mensagens. (Inclui lógica de filas ou agendamento se necessário).
- `manager_api.php` – Endpoint para dados dos **gestores**. Fornece dados agregados e estatísticas em tempo real para popular os gráficos e indicadores do painel dos gestores (por ex., retorna em JSON o número de novos usuários por hora, taxas de entrega, etc.). (Requer autenticação de gestor).
- `admin_api.php` – Endpoint para funções **administrativas** via AJAX. Por exemplo, criação/edição rápida de um registro via modal, obtenção de dados para um gráfico no dashboard admin, ou outras operações admin que se beneficiem de atualização assíncrona. (Requer autenticação de admin).
- *(Cada arquivo API é autocontido para uma categoria de funcionalidade, facilitando manutenção. Todos usam apenas bibliotecas PHP open-source quando necessário e seguem as permissões definidas em* `auth.php` *para segurança.)*

Observação final: Essa estrutura em forma de árvore, com pastas bem definidas e comentários, permite que a equipe distribua as tarefas e desenvolva em paralelo. A separação entre **área pública**, **áreas restritas** (Parceria, Comunidade, Gestores) e **Administração** garante organização e segurança. Arquivos estáticos ficam centralizados em `assets/`, podendo incluir bibliotecas gratuitas (ex.: Charts, máscaras de input) sem dependências proprietárias. Com essa base pronta, o time poderá implementar todas as funcionalidades em dois dias de maneira coordenada e escalável. Boa codificação!
