# Arquitetura Técnica e Roteiro de Implementação: Stofgard Manager

## 1. Visão Geral Executiva e Filosofia do Projeto

Este relatório técnico constitui a documentação definitiva para a refatoração e desenvolvimento do sistema **Stofgard Manager**. O projeto nasce da necessidade de adaptar uma infraestrutura legada de software jurídico ("Juris Maiden") para atender às demandas operacionais específicas de uma empresa de higienização e impermeabilização de estofados. Diferente das tentativas anteriores que falharam na entrega de uma interface adequada, esta nova iteração prioriza uma arquitetura "Tablet-First", focada na usabilidade tátil, clareza visual e estrita adesão às regras de negócio da Stofgard.1

A filosofia central deste desenvolvimento é a construção de um **Monólito Modular Fechado**. O termo "fechado" refere-se à natureza estática do escopo: não há previsão de escalabilidade horizontal para múltiplos inquilinos (SaaS) ou adição futura de funcionalidades complexas fora do *core business* já mapeado. O sistema deve ser robusto, leve e performático, hospedado em ambiente VPS Contabo (Alemanha), utilizando o stack tecnológico **Laravel 12** e **FilamentPHP v3**.1

A transição do paradigma jurídico (focado em texto, prazos processuais e hierarquias complexas de tribunais) para o paradigma de serviços residenciais (focado em agendamento, execução visual e fluxo de caixa rápido) exige não apenas a criação de novos módulos, mas uma "limpeza cirúrgica" do código existente. A interface, ponto crítico de falha anterior, será radicalmente simplificada para eliminar menus laterais obstrutivos, adotando um layout de "Launcher" (Lançador de Aplicativos) no Dashboard, conforme a visão "Clean" solicitada.

## 2. Fundamentos da Arquitetura e Infraestrutura

### 2.1 Stack Tecnológico e Justificativa

A escolha do **TALL Stack** (Tailwind CSS, Alpine.js, Laravel, Livewire) via FilamentPHP v3 é estratégica para este projeto. Ela permite a criação de interfaces reativas e modernas (SPA-like) sem a complexidade de manter uma API RESTful separada de um frontend em Vue ou React.

* **Laravel 12 (Backend):** Garante a estabilidade, segurança e facilidade na manipulação de dados relacionais complexos (como a relação entre Orçamentos e Ordens de Serviço). A versão 12 traz otimizações de performance cruciais para servidores VPS com recursos compartilhados.
* **FilamentPHP v3 (Painel Administrativo):** Oferece componentes prontos para tabelas e formulários, acelerando o desenvolvimento. O desafio técnico aqui reside na customização profunda necessária para fugir do "look and feel" padrão de painel administrativo e entregar a experiência "Tablet App" desejada.
* **Alpine.js (Interatividade Frontend):** Será essencial para os componentes visuais customizados, como o widget de "Galões" no estoque e o widget de Clima, permitindo manipulação DOM leve sem o overhead de frameworks maiores.
* **MySQL/MariaDB (Banco de Dados):** Estrutura relacional clássica, ideal para garantir a integridade referencial entre clientes, serviços e lançamentos financeiros.

### 2.2 Ambiente de Hospedagem e Deploy

A infraestrutura na **VPS Contabo (Alemanha)** impõe certas restrições e vantagens. A latência geográfica para usuários no Brasil (Ribeirão Preto) deve ser mitigada através de estratégias de cache agressivas no nível da aplicação e do servidor web (Nginx).

* **Configuração do Nginx:** Deve ser otimizado para servir ativos estáticos (imagens de upload, CSS, JS) com compressão Gzip ou Brotli, essencial para o acesso via redes móveis (3G/4G) em tablets de campo.
* **Filament Assets:** É imperativo executar o comando php artisan filament:optimize no deploy para realizar o *cache warming* dos ícones e componentes Blade, reduzindo o tempo de carregamento inicial (First Contentful Paint).

## 3. Engenharia de Interface: O Paradigma "Tablet-First"

A principal falha das tentativas anteriores foi a incapacidade de adaptar a UX (Experiência do Usuário) para o formato tablet. O design padrão do Filament possui uma barra lateral (sidebar) persistente que consome aproximadamente 20% da largura da tela, o que em um tablet (orientação paisagem ou retrato) comprime a área útil de dados.

### 3.1 Solução Técnica para o Dashboard "Clean"

Para atingir o visual "branco, limpo e com ícones grandes" 1, é necessário intervir no ciclo de renderização do Filament. A simples configuração via PHP não permite remover a sidebar apenas na página inicial sem afetar as páginas internas de recursos. A solução arquitetural envolve a injeção condicional de CSS via RenderHooks.

A análise do repositório e da documentação do Filament indica que o AdminPanelProvider é o ponto de entrada para essa modificação. Injetaremos um bloco de estilo que utiliza o seletor CSS :has() (suportado nos navegadores modernos de tablets Android/iOS) para detectar a presença da classe específica do dashboard e ocultar a sidebar.

**Implementação Crítica no AdminPanelProvider.php:**

PHP

->renderHook(  
 'panels::body.end',  
 fn (): string => Blade::render(<<<'HTML'  
 <style>  
 /\* Detecta se o corpo da página pertence ao Dashboard \*/  
 body:has(.fi-page-dashboard).fi-sidebar {  
 display: none!important;  
 }  
 /\* Remove a margem esquerda que compensava a sidebar \*/  
 body:has(.fi-page-dashboard).fi-main {  
 margin-inline-start: 0!important;  
 padding-left: 0!important;  
 max-width: 100%!important;  
 }  
 /\* Centraliza o cabeçalho para a mensagem de Boas-Vindas \*/  
 .fi-page-dashboard.fi-header {  
 text-align: center;  
 justify-content: center;  
 padding-top: 2rem;  
 padding-bottom: 2rem;  
 }  
 /\* Ajuste de Grid para Cards Grandes \*/  
 .fi-page-dashboard.fi-widgets-grid {  
 gap: 2rem;  
 }  
 </style>  
 HTML)  
);

Esta abordagem garante que, ao navegar para módulos internos (ex: Clientes), a sidebar reapareça para facilitar a navegação profunda, mas mantenha a "Home" como um Hub de Lançamento limpo.1

### 3.2 Otimização de Alvos de Toque (Touch Targets)

Seguindo as diretrizes de design para interfaces táteis (como Material Design), elementos interativos devem ter no mínimo 48x48 pixels.3 No contexto do Stofgard Manager:

* **Campos de Formulário:** Devem ser configurados com espaçamento vertical aumentado. O uso de compact() no Filament deve ser evitado.
* **Botões de Ação:** Ícones isolados (como "Editar" ou "Excluir" em tabelas) são propensos a "toques fantasmas" ou erros. A estratégia será utilizar ActionGroups ou botões com *labels* textuais sempre que o espaço permitir, ou aumentar o padding dos ícones via CSS customizado do tema.
* **Inputs Modestos:** A requisição pede "campos modestos" para não comprometer o espaço.1 Isso implica no uso inteligente de Grid::make()->columns(['md' => 2, 'lg' => 3]) para garantir que, em tablets, os campos se organizem lado a lado, evitando o "efeito linguiça" (campos esticados de ponta a ponta) que dificulta a leitura visual.

## 4. Estratégia de Refatoração do Legado ("Juris Maiden")

A base de código "Juris Maiden" contém entidades irrelevantes para o negócio de higienização. A manutenção dessas tabelas e modelos cria "ruído" técnico e riscos de segurança.

### 4.1 Protocolo de Limpeza (Cleanup)

O Agente de IA deve ser instruído a executar uma exclusão destrutiva das seguintes entidades, mapeadas a partir da análise do domínio jurídico versus o novo domínio 1:

* **Models e Migrations:** Lawsuit (Processos), Hearing (Audiências), Court (Tribunais), Judge (Juízes), Petition (Petições).
* **Controllers e Resources:** Todos os arquivos em app/Http/Controllers e app/Filament/Resources que referenciem as entidades acima.
* **Rotas:** O arquivo routes/web.php deve ser resetado para conter apenas a rota padrão de login, removendo quaisquer endpoints de API legados.

A estrutura de usuários, no entanto, deve ser preservada e adaptada. O sistema de permissões (se houver) deve ser simplificado, pois o Stofgard opera com uma hierarquia plana de confiança entre os 4 usuários definidos (Allisson, Maria, Jaelsa, Raelcia).

## 5. Modelagem de Dados e Regras de Negócio

A modelagem do banco de dados deve refletir estritamente o fluxo operacional descrito, suportando as regras de comissionamento e garantia.

### 5.1 Entidade: Ordens de Serviço (service\_orders)

Esta é a tabela central do sistema. Ela orquestra a relação entre o cliente, a execução do serviço e o financeiro.

| **Campo** | **Tipo** | **Descrição/Regra de Negócio** |
| --- | --- | --- |
| id | UUID | Identificador universal único.1 |
| client\_id | FK | Vínculo com a tabela de clientes. |
| partner\_id | FK (Nullable) | Vínculo com Lojas Parceiras. Se nulo, é venda direta (Marketing). |
| partner\_order\_number | String | Número da OS interna da loja parceira. |
| seller\_name | String | Nome do vendedor da loja (para conferência de comissão). |
| type | Enum | higienizacao, impermeabilizacao. |
| status | Enum | pendente, agendado, concluido, cancelado. |
| payment\_status\_cli | Enum | pendente, pago. Status do pagamento do cliente final. |
| payment\_status\_com | Enum | pendente, pago, n/a. Status do repasse da comissão. |
| execution\_date | Date | Data da realização do serviço. |
| warranty\_expires\_at | Date | Calculado automaticamente: Data de execução + 1 ano (apenas para Impermeabilização). |
| value\_agreed | Decimal | Valor final fechado com o cliente. |
| commission\_rate | Integer | Porcentagem de comissão (Padrão: 30% se tiver parceiro, 0% se não). |
| commission\_value | Decimal | Valor monetário da comissão (Calculado). |

Regra de Negócio - Garantia (Countdown):

Para serviços de impermeabilização, o sistema deve calcular a data de validade. Na interface (View), isso não deve ser apenas uma data, mas um componente visual:

* **Lógica:** hoje < warranty\_expires\_at? Ativa (Verde) : Vencida (Vermelho).
* **Visualização:** Um *badge* com contagem regressiva: "Vence em 120 dias".

### 5.2 Entidade: Clientes (clients)

A tabela de clientes deve ser enxuta, mas capaz de armazenar histórico.

* **Importação de Contatos:** A funcionalidade de importar contatos da agenda do celular 1 é tecnicamente complexa em aplicações web. A solução recomendada é implementar um campo de upload de arquivo .vcf (vCard) ou .csv exportado do Google Contacts/iCloud, que o backend processará para criar registros em massa. Uma integração direta via API de hardware do dispositivo (Web Contacts API) é experimental e pode não funcionar em todos os tablets Android, portanto, o upload de arquivo é a rota robusta.

### 5.3 Entidade: Orçamentos (budgets)

O orçamento é a antessala da Ordem de Serviço.

* **Validade:** 07 dias estritos. Um *Job* (Cron) diário deve verificar orçamentos com created\_at > 7 dias e marcar como expirado. Após 30 dias, deve realizar o *Soft Delete* ou exclusão física.
* **Regras de Arredondamento:** O sistema deve possuir uma classe de serviço PricingService que aplica a regra de arredondamento para baixo até R$ 50,00.
  + *Exemplo:* Valor calculado R$ 290,00 -> Mantém. Valor R$ 285,00 -> Arredonda para R$ 250,00 (se a regra for agressiva) ou R$ 250,00. A regra descrita "arredondar para baixo até R$ 50,00" sugere forçar finais 00 ou 50.
  + *Algoritmo Sugerido:* floor($valor / 50) \* 50.

### 5.4 Entidade: Almoxarifado (inventory)

O almoxarifado divide-se em Bens Duráveis (Equipamentos) e Consumíveis (Químicos).

* **Visualização de Galões:** Este é um requisito de UI crítico. O campo quantity\_liters para produtos químicos (Impermeabilizante/Flotador) não deve ser apenas um número.
  + **Implementação:** Um Widget Blade customizado que recebe o valor total de litros e renderiza um ícone SVG de galão para cada 20 litros inteiros.
  + **Lógica de Alerta:** Se quantity\_liters <= 20, o sistema dispara uma Notification::make()->danger()->title('Escassez Urgente').

## 6. Especificações Técnicas dos Módulos

### 6.1 Dashboard (Home)

Este é o "rosto" do sistema. Deve ser construído como uma Page customizada do Filament, substituindo a dashboard padrão.

* **Widgets:**
  1. **WeatherShortcutsWidget:** Consome a API Open-Meteo (gratuita) para as coordenadas de Ribeirão Preto. Exibe temperatura e ícone do clima.
  2. **NavigationGridWidget:** Um grid responsivo (1 coluna em mobile, 2 em tablet, 4 em desktop) contendo Cards grandes. Cada Card é um link <a> estilizado com classes Tailwind (p-8, shadow-lg, rounded-xl, flex-col) contendo o ícone Heroicon correspondente e o título do módulo.
  3. **RemindersWidget:** Lista de tarefas pendentes da Agenda, filtrada pelo usuário logado.

### 6.2 Financeiro

Deve suportar lançamentos polimórficos (ligados a OS ou avulsos).

* **Alertas:** O sistema deve varrer diariamente contas a receber vencidas e comissões de parceiros pendentes.
* **Uploads:** Implementar SpatieMediaLibrary para anexar comprovantes em cada lançamento. A visualização deve ser via Modal para não navegar para fora da página.

### 6.3 Configurações

Para manter a "limpeza", todas as operações de CRUD auxiliares (criação de categorias, ajuste de tabelas de preço base, configuração de usuários) devem ser movidas para um Cluster de Recursos chamado "Configurações", acessível via um único ícone no Dashboard, evitando poluir o menu principal (que estará oculto na home, mas visível internamente).

## 7. Motor de Precificação e Dados (Seeding)

A consistência dos dados iniciais é vital. As tabelas de preço (TABELA HIGI e TABELA IMPER) fornecem a base lógica.1

### 7.1 Lógica de Atualização Monetária (IGPM/IPCA)

Os documentos indicam que os preços base são de **Novembro de 2021** e devem sofrer um reajuste de **22,50%** (IPCA acumulado até Nov/2025). O sistema não deve calcular isso em tempo real para evitar flutuações não intencionais; os valores já atualizados devem ser persistidos no banco de dados durante o *Seeding*.

**Exemplos de Dados para o Seeder (PriceListSeeder):**

* **Cálculo:** Preço\_Base\_2021 \* 1.225
* **Item: Sofá 2 Lugares (Retrátil)**
  + Base 2021 (À Vista): R$ 220,00.
  + **Preço Sistema (2025):** R$ 269,50.
* **Item: Impermeabilização Sofá 3 Lugares**
  + Base 2021 (À Vista): R$ 420,00.
  + **Preço Sistema (2025):** R$ 514,50.

Esses valores devem ser carregados na tabela services ou products e utilizados como suggested\_value na criação de Orçamentos.

### 7.2 Controle de Acesso (Hardcoded Users)

Conforme requisito de segurança, o DatabaseSeeder deve garantir a existência (e apenas a existência) dos 4 usuários autorizados.

* **Estratégia:** Utilizar User::updateOrCreate baseado no e-mail para evitar duplicatas e garantir que as senhas (hash) estejam sempre sincronizadas com a especificação.1

## 8. Geração de Documentos e Relatórios

A formalização via PDF é crucial para a imagem profissional da Stofgard.

### 8.1 Especificações do PDF

* **Ferramenta:** barryvdh/laravel-dompdf é suficiente e mais leve que o Browsershot para VPS básica.
* **Cabeçalho:** Deve ser fixo em todas as páginas.
  + Logotipo Stofgard (Esquerda).
  + Razão Social e CNPJ 58.794.846/0001-20 (Direita).
  + **Restrição Crítica:** O endereço físico da empresa deve ser **suprimido** via código (display: none ou omissão no Blade) em todos os templates.1
* **Rodapé:**
  + Telefone: (16) 99104-0195.
  + Metadados: "Emitido em por [Iniciais do Usuário]".
* **Corpo:** Tabela zebrada (striped) para facilitar leitura, listando itens, quantidades e valores.

## 9. Roteiro Detalhado para o Agente de IA (VS Code)

Abaixo encontra-se o roteiro técnico mastigado, estruturado como um prompt de comando sequencial para um Agente de IA (como GitHub Copilot ou similar) executar a refatoração.

### **Instruções para o Agente de IA**

CONTEXTO:

Você está atuando como Arquiteto de Software Sênior. Sua tarefa é converter uma aplicação Laravel existente ("Juris Maiden") no sistema "Stofgard Manager". O sistema é fechado, monolítico e deve rodar em um Tablet.

RESTRIÇÃO DE INTERFACE (CRÍTICA):

O Dashboard deve ser uma tela cheia, sem barra lateral. Utilize renderHook no AdminPanelProvider para injetar CSS que oculta a .fi-sidebar apenas na rota do dashboard.

#### **ETAPA 1: LIMPEZA E PREPARAÇÃO (CLEANUP)**

1. **Excluir Migrations e Models:** Remova todos os arquivos relacionados a Lawsuits, Hearings, Courts, Judges, Petitions.
2. **Limpar Rotas:** Resete o arquivo routes/web.php e routes/api.php.
3. **Configuração:** Defina config/app.php para timezone => 'America/Sao\_Paulo' e locale => 'pt\_BR'.

#### **ETAPA 2: ARQUITETURA DE DADOS (MIGRATIONS)**

Crie as seguintes tabelas com UUIDs como chave primária:

1. clients: name (string, required), phone, mobile, address, notes, file\_attachments (json).
2. partners: name (Lojas), default\_commission (int, default 30).
3. service\_orders:
   * Relações: client\_id, partner\_id (nullable).
   * Campos: partner\_os\_number, seller\_name, type (enum: higienizacao, impermeabilizacao), status, execution\_date, warranty\_expires\_at (date, nullable), value\_agreed (decimal), commission\_value (decimal), photos (json).
4. budgets: client\_data (json), items (json), total\_value, status, expires\_at.
5. inventory\_items: name, type (equipment, chemical), quantity, unit (liters, units).
6. financial\_transactions: type (in/out), category, amount, due\_date, paid\_at, related\_order\_id.

#### **ETAPA 3: IMPLEMENTAÇÃO DO DASHBOARD (WIDGETS)**

1. Crie app/Filament/Widgets/WeatherShortcutsWidget.php.
   * Implemente chamada HTTP para Open-Meteo API (Latitude -21.1775, Longitude -47.8103).
   * Exiba temperatura e condição atual.
2. Crie app/Filament/Widgets/HomeNavigationWidget.php.
   * Desenvolva uma view Blade customizada com um Grid CSS (grid-cols-2 md:grid-cols-3).
   * Insira 6 Cards grandes com ícones Heroicons (Clientes, OS, Agenda, Financeiro, Orçamento, Estoque).
   * Links devem apontar para MyResource::getUrl('index').
3. Registre os widgets em app/Filament/Pages/Dashboard.php e remova os widgets padrões.

#### **ETAPA 4: LÓGICA DE NEGÓCIO (MODELS)**

1. **ServiceOrder Model:**
   * No método booted (evento saving), implemente:
     + Se partner\_id existir, commission\_value = value\_agreed \* 0.30.
     + Se type == impermeabilizacao, warranty\_expires\_at = execution\_date + 1 year.
   * Crie um Accessor getWarrantyStatusAttribute: Retorna 'Ativa' (Verde) se data atual < validade, senão 'Vencida' (Vermelho).
2. **Budget Resource:**
   * Implemente a Action ApplyRounding: total = floor(total / 50) \* 50.
   * Implemente a Action ConvertToOS: Replica dados para service\_orders e muda status para approved.

#### **ETAPA 5: VISUALIZAÇÃO DE ESTOQUE (BLADE)**

1. No InventoryResource, crie uma View customizada para a coluna quantity quando o tipo for 'chemical'.
2. Lógica Blade:  
   Blade  
   @php $galoes = floor($state / 20); @endphp  
   <div class="flex">  
    @foreach(range(1, $galoes) as $i)  
    <x-heroicon-s-beaker class="w-6 h-6 text-blue-500"/>  
    @endforeach  
   </div>  
   @if($galoes <= 1) <span class="text-red-600 font-bold">ALERTA: ESCASSEZ</span> @endif

#### **ETAPA 6: SEEDING (DADOS INICIAIS)**

1. Popule a tabela users EXATAMENTE com os 4 usuários fornecidos (Allisson, Maria, Jaelsa, Raelcia). Senhas devem ser bcrypt('Swordfish') ou bcrypt('Stofgard2026') conforme especificado.
2. Popule a tabela de preços baseada no cálculo Base\_2021 \* 1.225.

EXECUÇÃO:

Comece pela alteração do AdminPanelProvider para fixar o layout do Dashboard, depois proceda com as Migrations.

## 10. Conclusão

Este roteiro técnico preenche as lacunas deixadas pelas tentativas anteriores ao atacar diretamente o problema de layout do Filament e ao formalizar as regras de negócio implícitas nos documentos de precificação. A estrutura proposta garante um sistema estável, visualmente intuitivo e rigorosamente alinhado com o fluxo de caixa e operacional da Stofgard.

#### Referências citadas

1. Gemini.txt
2. Render hooks - Core Concepts - Filament, acessado em dezembro 30, 2025, <https://filamentphp.com/docs/3.x/support/render-hooks>
3. Touch Target - Material Design, acessado em dezembro 30, 2025, <https://m2.material.io/develop/web/supporting/touch-target>
4. Looking at WCAG 2.5.5 for Better Target Sizes - CSS-Tricks, acessado em dezembro 30, 2025, <https://css-tricks.com/looking-at-wcag-2-5-5-for-better-target-sizes/>