INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA
CAMPUS ARIQUEMES
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LUCAS PEDREIRA VITAL

DESENVOLVIMENTO DE PROTÓTIPO DE SISTEMA WEB COM INTELIGÊNCIA ARTIFICIAL PARA AUXÍLIO A ADVOGADOS EM ATIVIDADES JURÍDICAS

Ariquemes
2025

LUCAS PEDREIRA VITAL

DESENVOLVIMENTO DE PROTÓTIPO DE SISTEMA WEB COM INTELIGÊNCIA ARTIFICIAL PARA AUXÍLIO A ADVOGADOS EM ATIVIDADES JURÍDICAS

Relatório técnico entregue como Trabalho Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – (IFRO), Campus Ariquemes, como requisito parcial para obtenção do grau de tecnólogo junto ao curso de Análise e Desenvolvimento de Sistemas.
Orientador: Luciano Topolniak

Ariquemes
2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

V836d

Vital, Lucas Pedreira.

Desenvolvimento de protótipo de sistema web com inteligência artificial para auxílio a advogados em atividades jurídicas / Lucas Pedreira Vital. - Ariquemes, 2025.

50 f. : il.

Orientador(a): Prof. Me. Luciano Topolniak.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Ariquemes, 2025.

1. Inteligência artificial. 2. Automação jurídica. 3. Geração de documentos jurídicos. I. Topolniak, Luciano (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Renilce Silva Morais, CRB-11/906

LUCAS PEDREIRA VITAL

DESENVOLVIMENTO DE PROTÓTIPO DE SISTEMA WEB COM INTELIGÊNCIA ARTIFICIAL PARA AUXÍLIO A ADVOGADOS EM ATIVIDADES JURÍDICAS

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Analista e Desenvolvedor de Sistemas" e aprovado em sua forma final pelo Curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Ariquemes, 15 de julho de 2025.

Banca Examinadora:

gov.br
Documento assinado digitalmente
LUCIANO TOPOLNIAK
Data: 25/07/2025 18:28:31-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Luciano Topolniak
(orientador)
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO - Campus Ariquemes)

ICP Brasil
Documento assinado digitalmente
ANDREY ALENCAR QUADROS
Data: 27/07/2025 08:21:25-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Andrey Alencar Quadros
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO - Campus Ariquemes)

gov.br
Documento assinado digitalmente
MARCOS ALVES FAINO
Data: 25/07/2025 18:36:52-0300
Verifique em https://validar.iti.gov.br

Prof. Especialista Marcos Alves Faino
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO - Campus Ariquemes)

AGRADECIMENTOS

Em primeiro lugar, gostaria de expressar minha profunda gratidão à minha esposa, Sabrina, por todo o apoio e incentivo incondicional desde o início desta jornada até este momento. Sua compreensão, paciência e palavras de encorajamento foram fundamentais para que eu pudesse concluir este trabalho com dedicação e tranquilidade.

Agradeço imensamente ao meu orientador, Me. Luciano Topolniak, pela orientação precisa, pelos conselhos enriquecedores e pela motivação contínua que impulsionaram este projeto. Sua expertise e comprometimento foram essenciais para o desenvolvimento deste trabalho.

Ao nosso coordenador, Me. Andrey Alencar Quadros, deixo meu reconhecimento por sua dedicação e disposição em ajudar, sempre pronto a oferecer suporte, independentemente da hora ou da circunstância. Seu esforço em garantir um ambiente propício ao aprendizado e à pesquisa é notável e inspirador.

Não poderia deixar de agradecer ao advogado Dr. Anderson Leviski, cuja contribuição foi de extrema importância para o enriquecimento deste trabalho. Suas orientações práticas, experiências compartilhadas e insights valiosos trouxeram uma perspectiva única e fundamental ao projeto.

A todos os que, direta ou indiretamente, contribuíram para a realização deste trabalho, seja por meio de palavras de apoio, sugestões ou colaboração, meu sincero muito obrigado. Este projeto é reflexo não apenas de meu esforço, mas também da contribuição inestimável de cada um de vocês.

RESUMO

Este relatório técnico apresenta o desenvolvimento de um protótipo de sistema web que utiliza inteligência artificial para auxiliar advogados na criação de alegações finais processuais. O sistema combina tecnologias modernas, como Laravel, Vue.js e a API da OpenAI, para automatizar a geração de textos jurídicos estruturados, otimizando tarefas repetitivas e melhorando a qualidade dos documentos produzidos. A pesquisa aborda a problemática enfrentada por escritórios de advocacia diante da alta demanda e prazos reduzidos, propondo uma solução tecnológica que alinha eficiência, consistência e apoio decisional. A arquitetura do sistema foi dividida em camadas, garantindo escalabilidade e flexibilidade, enquanto o banco de dados foi projetado para armazenar e gerenciar informações processuais com segurança e organização. Os resultados obtidos demonstraram funcionalidade integrada, com feedbacks iniciais positivos que destacaram a interface intuitiva e a relevância da automatização. Apesar das limitações observadas, como a dependência da API externa e a necessidade de personalizações manuais, o projeto evidencia potencial para evolução, com melhorias planejadas para ampliar a acessibilidade e a robustez da solução.

Palavras-chave: inteligência artificial; automação jurídica; geração de documentos jurídicos.

ABSTRACT

This technical report presents the development of a web system prototype that utilizes artificial intelligence to assist lawyers in drafting final legal arguments. The system integrates modern technologies, such as Laravel, Vue.js, and the OpenAI API, to automate the generation of structured legal texts, optimizing repetitive tasks and improving document quality. The research addresses challenges faced by law firms dealing with high demands and tight deadlines, proposing a technological solution that emphasizes efficiency, consistency, and decision-making support. The system's architecture was divided into layers, ensuring scalability and flexibility, while the database was designed to securely and systematically store and manage procedural information. The results demonstrated integrated functionality, with initial positive feedback highlighting the intuitive interface and the relevance of automation. Despite some limitations, such as dependency on the external API and the need for manual customization, the project shows potential for further development, with planned improvements to enhance accessibility and system robustness.

Keywords: artificial intelligence; legal automation; legal document generation.

LISTA DE FIGURAS

Figura 1 – Implementação do método `index`. 15
Figura 2 – Implementação do método `store`. 17
Figura 3 – Implementação do método `prepareAttachments`. 18
Figura 4 – Implementação do método `buildPrompt` (Parte 1). 19
Figura 5 – Implementação do método `buildPrompt` (Parte 2). 20
Figura 6 – Implementação do método `createAndRun`. 21
Figura 7 – Implementação do método `waitOnRun`. 21
Figura 8 – Implementação do método `show`. 22
Figura 9 – Implementação do método `uploadFiles`. 23
Figura 10 – Implementação do método `formatarTesesPreliminares`. 24
Figura 11 – Implementação do método `formatarTesesMerito`. 24
Figura 12 – Implementação do método `formatarRequerimentos`. 25
Figura 13 – Implementação do método `formatarAdvogados`. 25
Figura 14 – Estado inicial do formulário - Qualificação do Processo e Anexos. 29
Figura 15 – Estado inicial do formulário - Dinâmica dos Fatos e Teses Preliminares. . . 30
Figura 16 – Estado inicial do formulário - Mérito Processual, Requerimentos e Advogados. 31
Figura 17 – Adição de novo pedido/requerimento. 32
Figura 18 – Configuração do componente para renderizar Markdown. 33
Figura 19 – Template do componente que renderiza Markdown. 34
Figura 20 – Definição de propriedades e estado no componente FileUpload. 35
Figura 21 – Função de upload no componente FileUpload. 36
Figura 22 – Função de reset e template do componente FileUpload. 37
Figura 23 – Recorte da interface - Seção Qualificação. 38
Figura 24 – Recorte da interface - Seção Anexos. 39
Figura 25 – Recorte da interface - Seção Dinâmica dos Fatos. 40
Figura 26 – Recorte da interface - Seção Preliminares de Mérito. 41
Figura 27 – Recorte da interface - Seção Mérito Processual. 42
Figura 28 – Recorte da interface - Seção Dos Pedidos e Requerimentos. 42
Figura 29 – Recorte da interface - Seção Dados do Advogado. 43
Figura 30 – Exibição da peça jurídica gerada na interface do usuário (Parte 1). 44
Figura 31 – Exibição da peça jurídica gerada na interface do usuário (Parte 2). 45

LISTA DE ABREVIATURAS E SIGLAS

FileUpload	Componente reutilizável para envio de arquivos
HTML	Linguagem de Marcação de Hipertexto (*HyperText Markup Language*)
IA	Inteligência Artificial
InertiaJS	Ferramenta de integração entre *backend* e *frontend*
Markdown	Linguagem de marcação para formatação de texto
OpenAI API	Interface de Programação de Aplicações da OpenAI (*Application Programming Interface*)
PDF	Formato Portátil de Documento (*Portable Document Format*)
ShadCN/UI	Biblioteca para estilização de elementos de interface
UI	Interface de Usuário (*User Interface*)
Vue.js	*Framework* JavaScript para desenvolvimento de interfaces

SUMÁRIO

1 INTRODUÇÃO 11
1.1 CONTEXTUALIZAÇÃO DO PROBLEMA 11
1.2 OBJETIVOS 11
1.2.1 Objetivo Geral 11
1.2.2 Objetivos Específicos 12
1.3 JUSTIFICATIVA 12
1.4 ESTRUTURA DO RELATÓRIO 12
2 DESENVOLVIMENTO DO BACKEND 14
2.1 TECNOLOGIAS UTILIZADAS 14
2.1.1 *Laravel* 14
2.1.2 *MySQL 8* 14
2.1.3 *OpenAI API* 14
2.2 EXPLICAÇÃO DO CÓDIGO 14
2.2.1 Controlador Principal 14
2.2.1.1 Método `index` 15
2.2.1.2 Método `store` 15
2.2.1.3 Método `prepareAttachments` 15
2.2.1.4 Método `buildPrompt` 15
2.2.1.5 Método `createAndRun` 16
2.2.1.6 Método `waitOnRun` 16
2.2.1.7 Método `show` 16
2.2.1.8 Método `uploadFiles` 16
2.2.1.9 Método `formatarTesesPreliminares` 16
2.2.1.10 Método `formatarTesesMerito` 16
2.2.1.11 Método `formatarRequerimentos` 16
2.2.1.12 Método `formatarAdvogados` 16
3 IMPLEMENTAÇÃO DO FRONTEND 26
3.1 ESTRUTURA DO FRONTEND 26
3.1.1 Página Index.vue 26
3.1.1.1 Funcionalidade Geral 26
3.1.1.2 Comportamento do Formulário 26
3.1.1.3 Adição e Remoção de Itens Dinâmicos 27
3.1.1.4 Gerenciamento de Anexos 27
3.1.2 Página Show.vue 27
3.1.2.1 Funcionalidade Geral 27
3.1.2.2 Renderização de Markdown 27
3.1.3 Componente FileUpload.vue 27

3.1.3.1 Funcionalidade Geral 27
3.1.3.2 Estado e Eventos 28
3.1.3.3 Interface do Usuário 28
4 INTERFACE DO USUÁRIO 38
4.1 SEÇÃO: QUALIFICAÇÃO 38
4.2 SEÇÃO: ANEXOS 39
4.3 SEÇÃO: DINÂMICA DOS FATOS 39
4.4 SEÇÃO: PRELIMINARES DE MÉRITO 40
4.5 SEÇÃO: MÉRITO PROCESSUAL 41
4.6 SEÇÃO: DOS PEDIDOS E REQUERIMENTOS 41
4.7 SEÇÃO: DADOS DO ADVOGADO 42
4.8 BOTÃO FINAL 43
4.9 EXIBIÇÃO DA PEÇA GERADA 43
4.9.1 Fluxo de Geração e Exibição 43
4.9.2 Interface da Exibição 44
4.9.3 Fluxo do Usuário 45
4.9.4 Facilidade de Uso 46
5 CONSIDERAÇÕES FINAIS 47
5.1 RESULTADOS OBTIDOS 47
5.2 LIMITAÇÕES DO SISTEMA 47
5.3 POSSIBILIDADES DE EVOLUÇÃO 48
5.4 CONCLUSÃO 48
REFERÊNCIAS 49

1 INTRODUÇÃO

O avanço das tecnologias digitais tem impulsionado transformações significativas em diversas áreas profissionais, e o setor jurídico não é exceção. Com a crescente complexidade das demandas judiciais, a necessidade de otimização dos processos se tornou evidente. Ferramentas que integram inteligência artificial (IA) emergem como soluções viáveis para aprimorar a produtividade e a precisão na elaboração de documentos jurídicos, facilitando a rotina de advogados e profissionais da área (Jurídico Ágil, 2025).

Neste relatório, apresenta-se o desenvolvimento de um protótipo de sistema web destinado à geração automatizada de alegações finais em processos judiciais. O sistema combina tecnologias modernas, como o framework *Laravel* (Laravel, 2025) para o backend, *Vue.js* (Vue.js, 2025) e *InertiaJS* (Inertia.js, 2025) para a interface do usuário, e a *API* da *OpenAI* (OpenAI, 2025; OpenAI PHP, 2025) para a automação da redação jurídica. A proposta visa não apenas explorar a viabilidade técnica desse tipo de aplicação, mas também demonstrar sua aplicabilidade prática dentro do contexto jurídico brasileiro.

1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

A redação de alegações finais é uma etapa crucial nos processos judiciais, exigindo precisão, coerência argumentativa e embasamento jurídico adequado (Jurídico Ágil, 2025). No entanto, essa atividade consome tempo considerável dos advogados, especialmente quando se trata de casos com grande volume de informações e prazos reduzidos. A necessidade de revisar documentos, organizar anexos e estruturar argumentos pode se tornar um desafio operacional para escritórios de advocacia e departamentos jurídicos.

A ausência de ferramentas especializadas voltadas à automação desse processo impacta diretamente a produtividade dos profissionais do direito. Soluções baseadas em IA podem contribuir para a geração de textos estruturados, padronizados e juridicamente embasados, reduzindo erros e garantindo maior eficiência. Dessa forma, este projeto busca preencher essa lacuna por meio do desenvolvimento de um sistema capaz de auxiliar na elaboração automatizada de alegações finais, otimizando a rotina dos advogados.

1.2 OBJETIVOS

A seguir, apresentam-se o objetivo geral e os objetivos específicos deste projeto.

1.2.1 Objetivo Geral

Desenvolver um protótipo de sistema web que utilize inteligência artificial para automatizar a geração de alegações finais em processos judiciais, promovendo eficiência, padronização e agilidade na produção de documentos jurídicos.

1.2.2 Objetivos Específicos

a) Criar uma plataforma web responsiva que permita o cadastro de informações processuais e o envio de documentos anexos;

b) Implementar o backend utilizando *Laravel* (Laravel, 2025), garantindo robustez, escalabilidade e integração eficiente com serviços de IA;

c) Desenvolver uma interface interativa e intuitiva com *Vue.js* (Vue.js, 2025) e *InertiaJS* (Inertia.js, 2025), proporcionando uma experiência fluida ao usuário;

d) Integrar a *API* da *OpenAI* (OpenAI, 2025; OpenAI PHP, 2025) para processar os dados fornecidos pelo usuário e gerar documentos jurídicos estruturados;

e) Armazenar os dados processuais e históricos de documentos gerados em um banco de dados *MySQL* (MySQL, 2025);

f) Avaliar a viabilidade do uso de IA na automação de redação jurídica, considerando aspectos técnicos e legais (Jurídico Ágil, 2025).

1.3 JUSTIFICATIVA

A implementação de um sistema para a geração automatizada de alegações finais justifica-se por diversos fatores:

a) **Otimização de Tempo:** A automação reduz o tempo necessário para a redação de documentos jurídicos, permitindo que advogados concentrem esforços em análises estratégicas (Jurídico Ágil, 2025).

b) **Padronização e Consistência:** O uso de IA garante uniformidade na estrutura dos documentos, minimizando inconsistências e erros (GeoLigard, 2025).

c) **Aprimoramento da Eficiência:** A integração de ferramentas tecnológicas ao fluxo de trabalho jurídico melhora a produtividade e a organização dos processos (Jurídico Ágil, 2025).

d) **Adaptação às Tendências de LegalTech:** O setor jurídico está em crescente digitalização, e este projeto alinha-se às inovações promovidas pelo movimento *LegalTech* (Jurídico Ágil, 2025).

Além disso, a adoção de tecnologias avançadas no direito pode contribuir para a democratização do acesso a soluções jurídicas eficientes, beneficiando profissionais e clientes (Jurídico Ágil, 2025).

1.4 ESTRUTURA DO RELATÓRIO

O relatório está organizado em cinco capítulos principais:

a) **Introdução:** Apresenta o contexto do problema, objetivos e justificativa do projeto.

b) **Desenvolvimento do Backend:** Detalha as tecnologias utilizadas, a arquitetura do sistema e a implementação das funcionalidades relacionadas ao processamento de dados e integração com IA (Laravel, 2025; OpenAI PHP, 2025; MySQL, 2025).

c) **Implementação do Frontend:** Descreve a estrutura da interface do usuário, destacando os principais componentes desenvolvidos (Vue.js, 2025; Inertia.js, 2025; Tailwind CSS, 2025; ShadCN Vue, 2025).

d) **Interface do Usuário:** Explica as seções do formulário, suas funcionalidades e a interação do usuário com o sistema (Vue.js, 2025; Inertia.js, 2025).

e) **Considerações Finais:** Apresenta os resultados obtidos, limitações do sistema e possibilidades de evolução futura (Jurídico Ágil, 2025).

Com essa estrutura, busca-se apresentar de forma clara e organizada o desenvolvimento do sistema, desde a concepção até a implementação prática, destacando sua aplicabilidade e relevância para o setor jurídico.

2 DESENVOLVIMENTO DO BACKEND

Neste capítulo, serão detalhadas as tecnologias utilizadas no desenvolvimento do protótipo do sistema *web*, bem como uma explicação sobre a implementação do código principal que permite a geração de alegações finais para processos jurídicos. Este sistema utiliza conceitos modernos de desenvolvimento *web*, integrações com Inteligência Artificial (IA) e boas práticas de programação.

2.1 TECNOLOGIAS UTILIZADAS

2.1.1 *Laravel*

O *framework Laravel* foi escolhido para o *backend* do sistema devido à sua simplicidade e robustez no desenvolvimento de aplicações *web*. O *Laravel* oferece uma arquitetura clara baseada no padrão MVC (*Model-View-Controller*), além de uma integração nativa com ferramentas modernas como o *InertiaJS* (Laravel, 2025). Ele também permite o uso de *middlewares*, filas de tarefas e integrações fáceis com APIs externas, como a *OpenAI API*, utilizada neste projeto.

2.1.2 *MySQL 8*

O sistema utiliza o banco de dados *MySQL* na versão 8, que oferece alta performance, suporte a transações *ACID* e uma sintaxe *SQL* robusta para gerenciamento de dados. O *MySQL* foi configurado com índices em tabelas específicas para otimizar buscas relacionadas a documentos e histórico de operações (MySQL, 2025). Sua capacidade de manipular grandes volumes de dados o torna ideal para sistemas com demandas crescentes.

2.1.3 *OpenAI API*

A integração com a *OpenAI API* possibilitou a utilização de um assistente especializado para a análise e geração de textos jurídicos. Essa *API* permite o processamento de linguagem natural (*NLP*), sendo ideal para tarefas que envolvam a composição automática de documentos complexos. A configuração de autenticação e a gestão de *threads* no sistema garantem que cada solicitação seja processada com segurança (OpenAI PHP, 2025; OpenAI, 2025).

2.2 EXPLICAÇÃO DO CÓDIGO

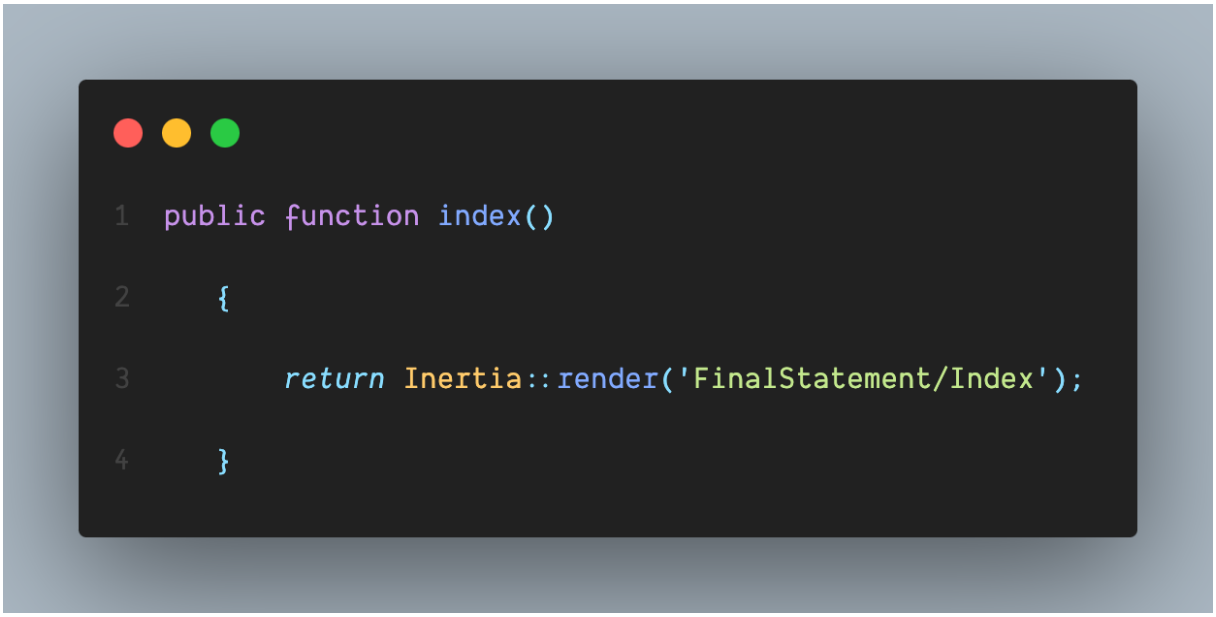
2.2.1 Controlador Principal

O controlador `FinalStatementController` desempenha um papel crucial no sistema, centralizando as funcionalidades relacionadas à geração de peças jurídicas. Cada método desempenha uma função específica no fluxo de trabalho.

2.2.1.1 Método `index`

Este método é responsável por renderizar a página inicial do módulo de alegações finais. A implementação está representada na Figura 1.

Figura 1 – Implementação do método `index`.



Fonte: Elaborado pelo autor, 2024.

2.2.1.2 Método `store`

Este é o método principal, responsável por receber os dados do formulário, processar as informações e interagir com a *API* da *OpenAI*. Sua implementação está representada na Figura 2.

2.2.1.3 Método `prepareAttachments`

Este método organiza os arquivos anexados pelo usuário para que possam ser processados pela *API*. A Figura 3 ilustra sua implementação.

2.2.1.4 Método `buildPrompt`

O método `buildPrompt` é responsável por gerar o *prompt* enviado à *API* da *OpenAI*, contendo as informações processuais fornecidas pelo usuário. A implementação do método foi dividida em duas partes para facilitar a visualização. A Figura 4 apresenta a primeira parte, enquanto a Figura 5 exibe a continuação.

2.2.1.5 Método `createAndRun`

Realiza a integração com a *API* da *OpenAI*, enviando o *prompt* e os anexos. A Figura 6 apresenta sua implementação.

2.2.1.6 Método `waitOnRun`

Monitora o status da execução até sua conclusão. A Figura 7 apresenta sua implementação.

2.2.1.7 Método `show`

O método `show` é responsável por renderizar a página onde o documento gerado é exibido. Ele utiliza o *InertiaJS* para conectar o *backend* ao *frontend*. Sua implementação é apresentada na Figura 8.

2.2.1.8 Método `uploadFiles`

O método `uploadFiles` gerencia o *upload* de arquivos pelo usuário, armazenando-os localmente no servidor e enviando-os para a *OpenAI*. Ele retorna um *JSON* com o ID do arquivo atribuído pela *OpenAI* em caso de sucesso, ou uma mensagem de erro em caso de falha. A Figura 9 apresenta a implementação deste método.

2.2.1.9 Método `formatarTesesPreliminares`

O método `formatarTesesPreliminares` processa uma lista de teses preliminares selecionadas pelo usuário, formatando-as em uma estrutura de lista. Caso nenhuma tese seja selecionada, o método retorna uma mensagem padrão. A Figura 10 exibe sua implementação.

2.2.1.10 Método `formatarTesesMerito`

O método `formatarTesesMerito` é semelhante ao `formatarTesesPreliminares`, mas processa teses relacionadas ao mérito do processo. Sua implementação é ilustrada na Figura 11.

2.2.1.11 Método `formatarRequerimentos`

O método `formatarRequerimentos` organiza os requerimentos enviados pelo usuário em uma lista. Ele é projetado para retornar cada requerimento em uma linha separada. A Figura 12 mostra sua implementação.

2.2.1.12 Método `formatarAdvogados`

O método `formatarAdvogados` formata uma lista de advogados para exibição, incluindo o nome e o número de inscrição na OAB. A Figura 13 apresenta sua implementação.

Figura 2 – Implementação do método `store`.

```php
public function store(StoreFinalStatementRequest $request)
    {
        $attachments = $this->prepareAttachments($request->file_ids);

        $prompt = $this->prompt($request);

        $response = $this->createAndRun($prompt, $attachments);

        $run = $this->waitOnRun($response, $response->threadId);

        $messages = [];

        if ($run->status === 'completed') {
            $messages = OpenAI::threads()->messages()->list($run->threadId);
        }

        $document = $messages->data[0]->content[0]->text->value;

        return Inertia::render('FinalStatement/Show', [
            'document' => $document,
        ]);
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 3 – Implementação do método `prepareAttachments`.

```php
private function prepareAttachments(array $fileIds): array
    {
        $attachments = [];

        foreach ($fileIds as $fileId) {
            if (! $fileId) {
                continue;
            }
            $attachments[] = [
                'file_id' => $fileId,
                'tools'   => [['type' => 'file_search']],
            ];
        }

        return $attachments;
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 4 – Implementação do método `buildPrompt` (Parte 1).

```php
private function buildPrompt($request)
    {
        $prompt = "Você é um assistente especializado em documentos jurídicos brasileiros.
        Gere uma peça de Alegações Finais em formato markdown usando as seguintes informações:

        DADOS DO PROCESSO:
        Endereçamento: {$request->enderecamento}
        Número do Processo: {$request->numero_processo}
        Nome do Acusado: {$request->nome_do_acusado}

        DOCUMENTOS ANEXADOS:
        - Inquérito Policial
        - Denúncia
        - Resposta à Acusação
        - Recebimento da Denúncia
        - Alegações Finais do MP

        DINÂMICA DOS FATOS:
        {$request->dinamica_fatos}

        PRELIMINAR DE MÉRITO:
        {$request->preliminar_merito}
```

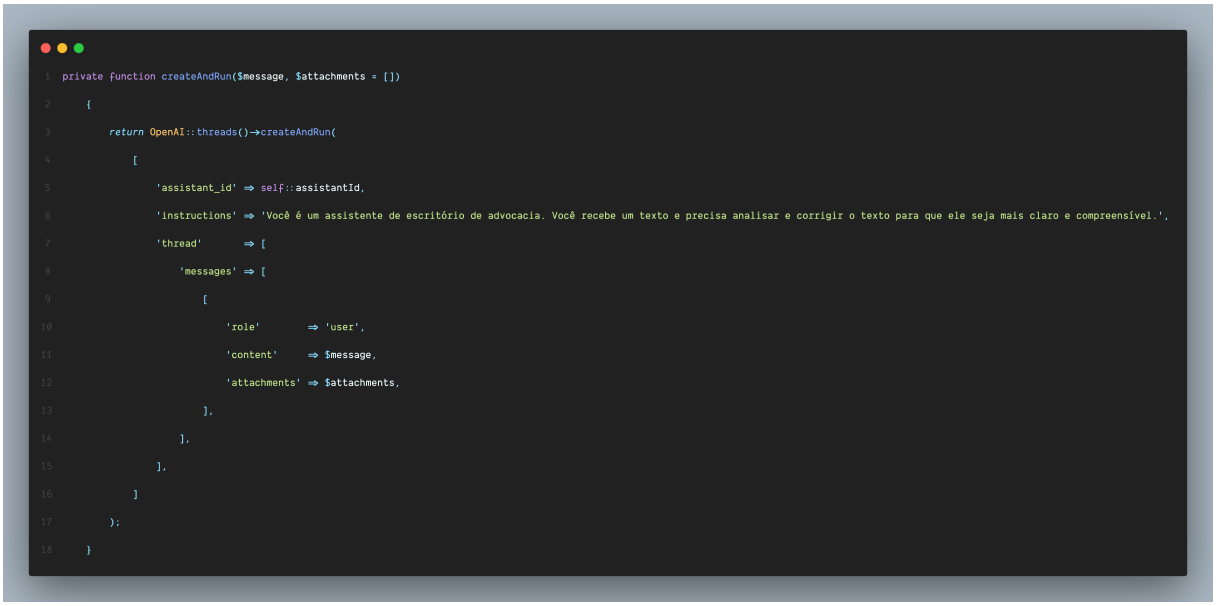
Fonte: Elaborado pelo autor, 2024.

Figura 5 – Implementação do método `buildPrompt` (Parte 2).

```
        TESES PRELIMINARES LEVANTADAS:" .
        $this->formatarTesesPreliminares($request->teses_preliminares_merito) . "

        MÉRITO PROCESSUAL:
        {$request->merito_processual}

        TESES DE MÉRITO LEVANTADAS:" .
        $this->formatarTesesMerito($request->teses_merito_processual) . "

        REQUERIMENTOS:
        " . $this->formatarRequerimentos($request->requerimentos) . "

        ADVOGADOS SUBSCRITORES:
        " . $this->formatarAdvogados($request->advogados) . "

        Por favor, gere um documento em markdown seguindo a estrutura formal de peças jurídicas brasileiras,
        incorporando todas as informações fornecidas de maneira coesa e fundamentada.";

        return $prompt;
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 6 – Implementação do método `createAndRun`.



```php
private function createAndRun($message, $attachments = [])
    {
        return OpenAI::threads()->createAndRun(
            [
                'assistant_id' => self::assistantId,
                'instructions' => 'Você é um assistente de escritório de advocacia. Você recebe um texto e precisa analisar e corrigir o texto para que ele seja mais claro e compreensível.',
                'thread'       => [
                    'messages' => [
                        [
                            'role'        => 'user',
                            'content'     => $message,
                            'attachments' => $attachments,
                        ],
                    ],
                ],
            ]
        );
    }
```

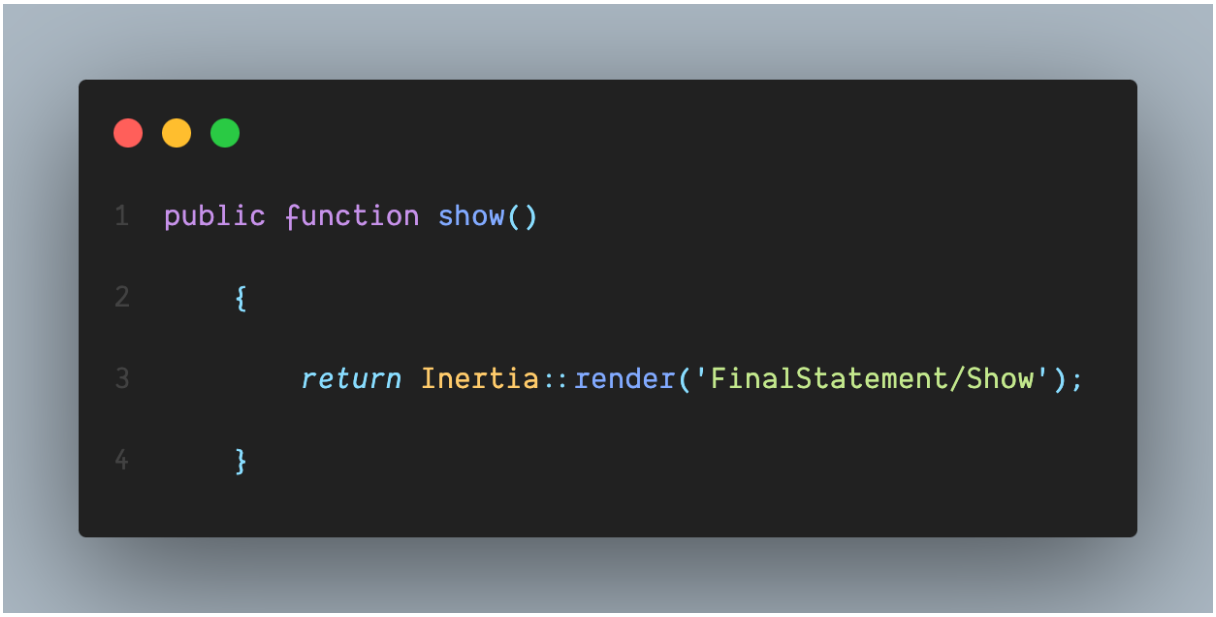
Fonte: Elaborado pelo autor, 2024.

Figura 7 – Implementação do método `waitOnRun`.

```php
private function waitOnRun($run, $threadId)
    {
        while ($run->status === 'queued' || $run->status === 'in_progress') {
            $run = OpenAI::threads()->runs()->retrieve($threadId, $run->id);
            sleep(1);
        }

        return $run;
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 8 – Implementação do método `show`.



Fonte: Elaborado pelo autor, 2024.

Figura 9 – Implementação do método `uploadFiles`.

```php
public function uploadFiles(Request $request)
    {
        $path = $request->file('file')->store('uploads');

        $filePath = storage_path('app/private/' . $path);

        try {
            $fileResponse = OpenAI::files()->upload([
                'file'    => fopen($filePath, 'r'),
                'purpose' => 'assistants',
            ]);

            // Retorna o ID do arquivo da OpenAI
            return response()->json([
                'success' => true,
                'file_id' => $fileResponse['id'],
                'message' => 'File uploaded and sent to OpenAI successfully!',
            ]);
        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => 'Failed to send file to OpenAI: ' . $e->getMessage(),
            ], 500);
        }
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 10 – Implementação do método `formatarTesesPreliminares`.

```php
private function formatarTesesPreliminares($teses)
    {
        $tesesSelecionadas = [];
        foreach ($teses as $tese => $selecionada) {
            if ($selecionada) {
                $tesesSelecionadas[] = "- " . str_replace('_', ' ', ucfirst($tese));
            }
        }
        return empty($tesesSelecionadas) ? "\nNenhuma tese preliminar selecionada." : "\n" . implode("\n", $tesesSelecionadas);
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 11 – Implementação do método `formatarTesesMerito`.

```php
private function formatarTesesMerito($teses)
    {
        $tesesSelecionadas = [];
        foreach ($teses as $tese => $selecionada) {
            if ($selecionada) {
                $tesesSelecionadas[] = "- " . str_replace('_', ' ', ucfirst($tese));
            }
        }
        return empty($tesesSelecionadas) ? "\nNenhuma tese de mérito selecionada." : "\n" . implode("\n", $tesesSelecionadas);
    }
```

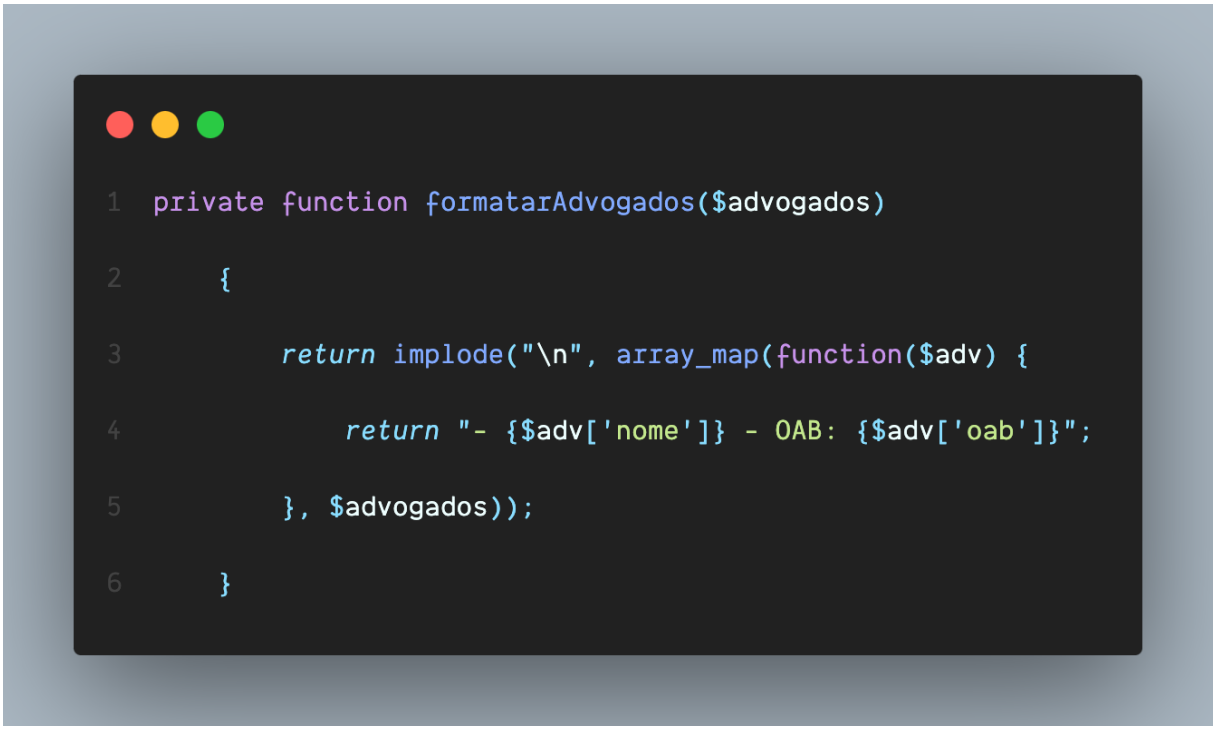
Fonte: Elaborado pelo autor, 2024.

Figura 12 – Implementação do método `formatarRequerimentos`.

```php
private function formatarRequerimentos($requerimentos)
    {
        return implode("\n", array_map(function($req) {
            return "- {$req['value']}";
        }, $requerimentos));
    }
```

Fonte: Elaborado pelo autor, 2024.

Figura 13 – Implementação do método `formatarAdvogados`.



```php
private function formatarAdvogados($advogados)
    {
        return implode("\n", array_map(function($adv) {
            return "- {$adv['nome']} - OAB: {$adv['oab']}";
        }, $advogados));
    }
```

Fonte: Elaborado pelo autor, 2024.

3 IMPLEMENTAÇÃO DO FRONTEND

Neste capítulo, apresentamos a implementação da interface UI (Interface de Usuário (*User Interface*)) do sistema web desenvolvido para gerar peças jurídicas automatizadas. A interface foi criada utilizando Vue.js (*Framework* JavaScript para desenvolvimento de interfaces) 3 e InertiaJS (Ferramenta de integração entre *backend* e *frontend*), com a integração de componentes estilizados por bibliotecas modernas como ShadCN/UI (Biblioteca para estilização de elementos de interface) (Vue.js, 2025; Inertia.js, 2025; ShadCN Vue, 2025). Este capítulo descreve as páginas principais do sistema: `Index.vue`, `Show.vue` e o componente `FileUpload`, detalhando seus propósitos, funcionalidades e a lógica aplicada.

3.1 ESTRUTURA DO FRONTEND

O frontend foi desenvolvido em Vue.js 3, utilizando a abordagem de composição (*Composition API*) para organizar a lógica e o estado dos componentes. As páginas estão localizadas no diretório `Pages/FinalStatement`, e cada uma delas desempenha papéis específicos no fluxo de uso do sistema (Vue.js, 2025).

3.1.1 Página Index.vue

A página `Index.vue` é responsável por exibir o formulário principal que o usuário utiliza para preencher as informações necessárias para a geração de uma peça jurídica. Ela inclui múltiplos campos de entrada, anexos de arquivos e opções de seleção. A estrutura do formulário foi projetada para ser intuitiva e eficiente.

3.1.1.1 Funcionalidade Geral

O formulário desta página oferece as seguintes funcionalidades principais:

a) Preenchimento dos dados de qualificação do processo (endereçamento, número do processo e nome do acusado).

b) Upload de documentos relevantes, como inquérito policial e denúncia, no formato PDF.

c) Campo para descrição detalhada da dinâmica dos fatos processuais.

d) Seleção de teses preliminares e de mérito processual relevantes ao caso.

e) Gerenciamento de pedidos/requerimentos e advogados, permitindo a adição e remoção de múltiplos registros.

3.1.1.2 Comportamento do Formulário

O formulário é inicializado com um estado padrão que organiza as informações necessárias para a geração da peça jurídica. Abaixo, estão os campos divididos em três partes principais:

3.1.1.3 Adição e Remoção de Itens Dinâmicos

A página permite que o usuário adicione ou remova dinamicamente campos relacionados a pedidos/requerimentos e advogados, o que torna o formulário flexível para diferentes necessidades jurídicas.

Requerimentos: O botão + *Pedido/Requerimento* adiciona um novo campo para registro. O layout do formulário reflete essa flexibilidade, conforme ilustrado na Figura 17.

3.1.1.4 Gerenciamento de Anexos

Os campos de anexos foram implementados utilizando o componente reutilizável `FileUpload`, que gerencia o upload de arquivos para o backend. A lógica e interface desse componente são descritas na Seção 3.1.3.

3.1.2 Página Show.vue

A página `Show.vue` é responsável por exibir o documento gerado pelo sistema com base nos dados fornecidos pelo usuário. Ela utiliza a biblioteca `Markdown` para converter o conteúdo gerado em HTML formatado, proporcionando uma exibição clara e organizada (Vue.js, 2025).

3.1.2.1 Funcionalidade Geral

Esta página apresenta o documento jurídico gerado, permitindo que o usuário visualize o resultado final. As características principais incluem:

a) Renderização de documentos no formato Markdown.

b) Interface amigável, que mantém a consistência visual com outras páginas do sistema.

3.1.2.2 Renderização de Markdown

A renderização dinâmica de documentos é realizada com a biblioteca `Markdown`. A configuração do componente e o template utilizado estão representados nas Figuras 18 e 19.

3.1.3 Componente FileUpload.vue

O componente `FileUpload` gerencia o upload de arquivos para o sistema. Ele é utilizado nos campos de anexos da página `Index.vue` e foi projetado para ser modular e reutilizável (Vue.js, 2025; Inertia.js, 2025).

3.1.3.1 Funcionalidade Geral

As principais funcionalidades incluem:

a) Seleção de arquivos pelo usuário.

b) Exibição do progresso do upload em tempo real.

c) Possibilidade de cancelar ou redefinir um upload em andamento.

d) Comunicação com o backend para armazenar os arquivos e associá-los ao processo.

3.1.3.2 Estado e Eventos

O componente gerencia seu estado utilizando variáveis como **isUploading** e **progress**, que representam o status do upload e o progresso em porcentagem, respectivamente.

3.1.3.3 Interface do Usuário

A interface exibe campos para seleção de arquivos, barra de progresso e opções para cancelar o upload. As figuras abaixo ilustram as partes principais do código:

Figura 14 – Estado inicial do formulário - Qualificação do Processo e Anexos.

```
const form = useForm({
    enderecamento: '',
    numero_processo: '',
    nome_do_acusado: '',
    file_ids: {
        anexo_inquerito: '',
        anexo_denuncia: '',
        anexo_resposta_acusacao: '',
        anexo_recebimento_denuncia: '',
        anexo_alegacoes_finais_mp: '',
    },
```

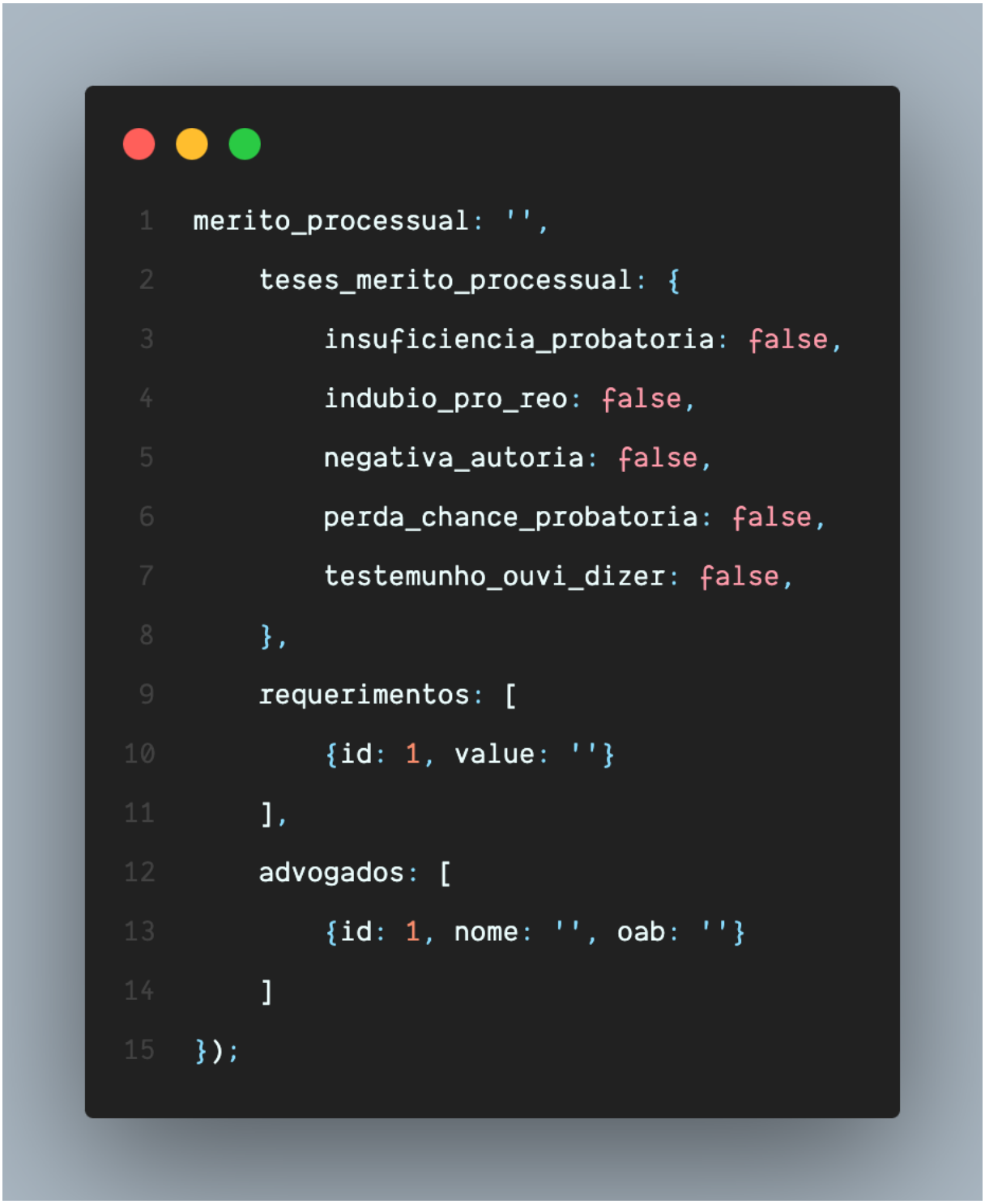
Fonte: Elaborado pelo autor, 2024.

Figura 15 – Estado inicial do formulário - Dinâmica dos Fatos e Teses Preliminares.

```
dinamica_fatos: '',
    preliminar_merito: '',
    teses_preliminares_merito: {
        invasao_domiciliar: false,
        abordagem_busca_pessoal: false,
        cerceamento_defesa: false,
        reconhecimento_ilegal: false,
        ausencia_fundamentacao: false,
    },
```

Fonte: Elaborado pelo autor, 2024.

Figura 16 – Estado inicial do formulário - Mérito Processual, Requerimentos e Advogados.



```
merito_processual: '',
    teses_merito_processual: {
        insuficiencia_probatoria: false,
        indubio_pro_reo: false,
        negativa_autoria: false,
        perda_chance_probatoria: false,
        testemunho_ouvi_dizer: false,
    },
    requerimentos: [
        {id: 1, value: ''}
    ],
    advogados: [
        {id: 1, nome: '', oab: ''}
    ]
});
```

Fonte: Elaborado pelo autor, 2024.

Figura 17 – Adição de novo pedido/requerimento.

```
1  const addRequerimento = () => {
2      form.requerimentos.push({id: form.requerimentos.length + 1, value: ''});
3  };
4
5  const removeRequerimento = (index: number) => {
6      form.requerimentos.splice(index, 1);
7  };
```

Fonte: Elaborado pelo autor, 2024.

Figura 18 – Configuração do componente para renderizar Markdown.

```
1  import {marked} from "marked";
2  import {computed} from "vue";
3  const props = defineProps({
4      document: String
5  });
6  marked.setOptions({
7      breaks: true,
8      gfm: true,
9  });
10 const formattedDocument = computed(() => {
11     if (!props.document) return '';
12     return marked(props.document);
13 });
```

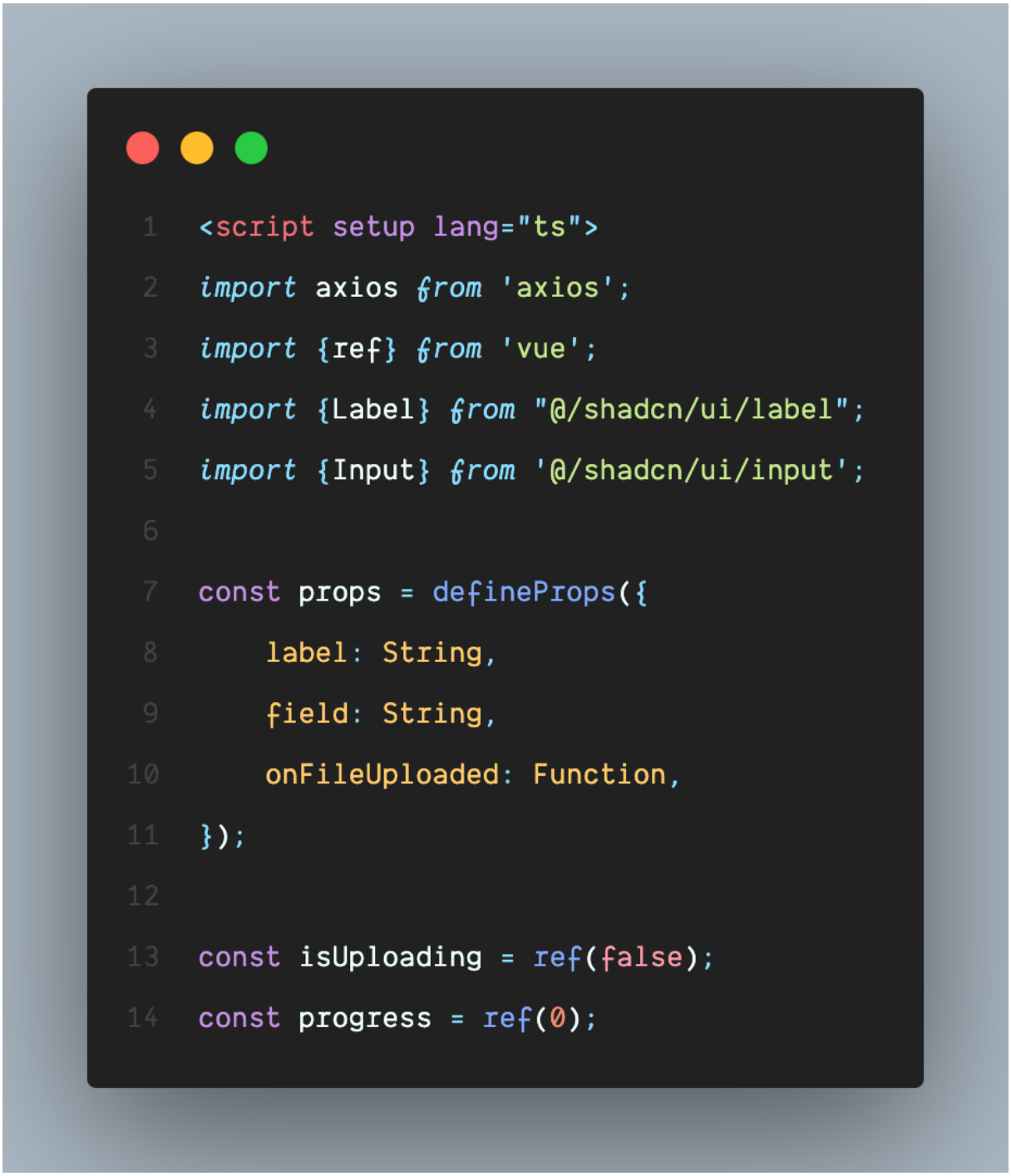
Fonte: Elaborado pelo autor, 2024.

Figura 19 – Template do componente que renderiza Markdown.

```
<Card>
    <CardContent>
        <div
        class="pt-4 prose prose-sm max-w-none dark:prose-invert"
        v-html="formattedDocument">
    </div>
    </CardContent>
</Card>
```

Fonte: Elaborado pelo autor, 2024.

Figura 20 – Definição de propriedades e estado no componente FileUpload.



```
<script setup lang="ts">
import axios from 'axios';
import {ref} from 'vue';
import {Label} from "@/shadcn/ui/label";
import {Input} from '@/shadcn/ui/input';

const props = defineProps({
    label: String,
    field: String,
    onFileUploaded: Function,
});

const isUploading = ref(false);
const progress = ref(0);
```

Fonte: Elaborado pelo autor, 2024.

Figura 21 – Função de upload no componente FileUpload.

```
function handleFileChange(event) {
    const file = event.target.files[0];
    if (!file) return;

    const formData = new FormData();
    formData.append('file', file);

    isUploading.value = true;

    axios.post(route('upload.store'), formData, {
        headers: {
            'Content-Type': 'multipart/form-data',
        },
        onUploadProgress: (progressEvent) => {
            progress.value = Math.round((progressEvent.loaded / progressEvent.total) * 100);
        },
    })
        .then((response) => {
            if (response.data.success) {
                props.onFileUploaded(props.field, response.data.file_id);
            }
        })
        .catch((error) => {
            console.error('Upload failed:', error);
        })
        .finally(() => {
            isUploading.value = false;
            progress.value = 0;
        });
}
```

Fonte: Elaborado pelo autor, 2024.

Figura 22 – Função de reset e template do componente FileUpload.

```
function resetFile() {
    props.onFileUploaded(props.field, null);
}
</script>

<template>
    <div class="flex flex-col">
        <Label :for="field" class="font-medium">{{ label }}</Label>
        <div class="inline-flex items-center gap-2 mt-2">
            <Input type="file" @change="handleFileChange"
            :id="field" />
            <progress v-if="isUploading" :value="progress"
            max="100">{{ progress }}%</progress>
            <button v-if="isUploading" @click="resetFile"
            class="text-red-500">Cancelar</button>
        </div>
    </div>
</template>
```

Fonte: Elaborado pelo autor, 2024.

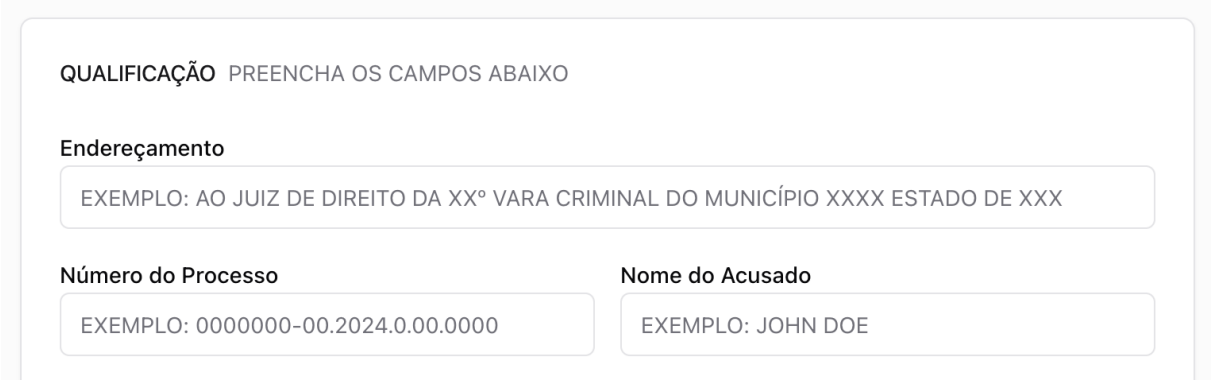
4 INTERFACE DO USUÁRIO

A interface do formulário foi desenvolvida com o objetivo de ser intuitiva e eficiente, permitindo que o advogado preencha as informações necessárias de forma estruturada e lógica. A interface divide-se em seções bem definidas, cada uma abordando aspectos cruciais do processo jurídico. Abaixo, descrevemos as funcionalidades e explicações jurídicas de cada seção, acompanhadas de recortes da interface.

4.1 SEÇÃO: QUALIFICAÇÃO

- **Descrição:** Essa seção solicita informações fundamentais para identificar o processo judicial e as partes envolvidas.
- **Campos:**
  - **Endereçamento:** Campo de texto para inserir o endereço do destinatário da peça jurídica (exemplo: juiz ou vara judicial). Essencial para direcionar corretamente a petição, indicando a autoridade que deve analisá-la.
  - **Número do Processo:** Identificador único do processo no sistema judicial. Indispensável para vincular a peça jurídica ao processo correto.
  - **Nome do Acusado:** Nome completo do réu, garantindo que não haja dúvidas sobre a quem se refere a petição.
- **Explicação Jurídica:** A correta identificação do destinatário, processo e acusado é fundamental para a validade formal da petição e para evitar erros que possam prejudicar a análise do caso.

Figura 23 – Recorte da interface - Seção Qualificação.

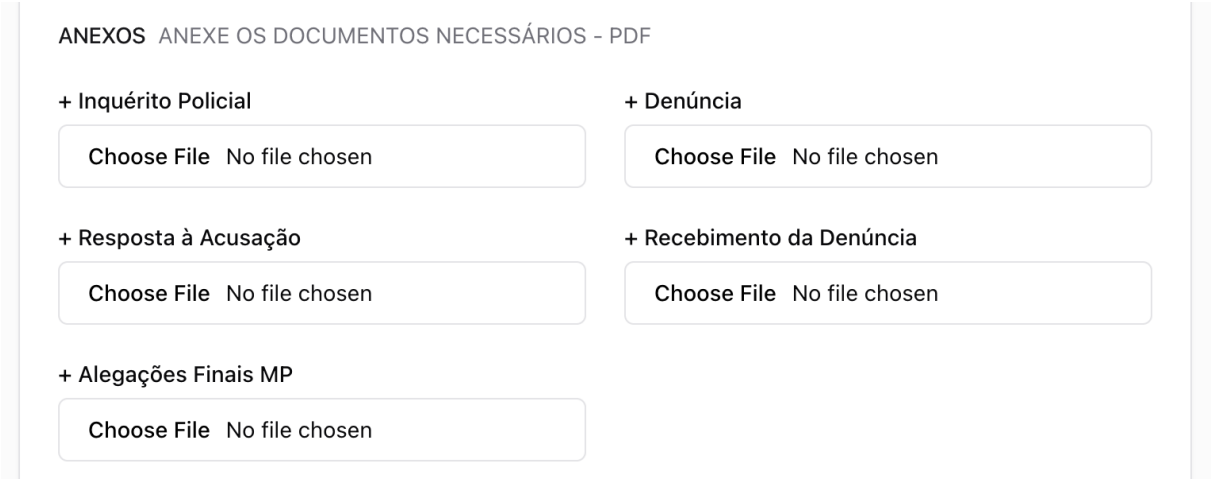


Fonte: Elaborado pelo autor, 2024.

4.2 SEÇÃO: ANEXOS

- **Descrição:** Permite o upload de documentos relevantes em formato PDF.
- **Campos de Upload:**
  - **Inquérito Policial:** Relatório da investigação inicial conduzida pela polícia. Essencial para compreender a base das acusações.
  - **Denúncia:** Peça acusatória apresentada pelo Ministério Público. Fundamenta a acusação e é crucial para a contestação.
  - **Resposta à Acusação:** Defesa prévia apresentada pelo acusado para refutar a denúncia. Documento que fortalece os argumentos da defesa.
  - **Recebimento da Denúncia:** Decisão do juiz que aceita a denúncia e dá início à ação penal. Importante para verificar a legitimidade da acusação.
  - **Alegações Finais MP:** Considerações finais do Ministério Público antes do julgamento. Necessárias para elaborar uma contra-argumentação sólida.
- **Interatividade:** Cada item apresenta um botão para selecionar o arquivo e um botão para cancelar o envio, caso necessário.
- **Explicação Jurídica:** A correta anexação de documentos garante que os argumentos da defesa estejam embasados em provas e decisões relevantes.

Figura 24 – Recorte da interface - Seção Anexos.



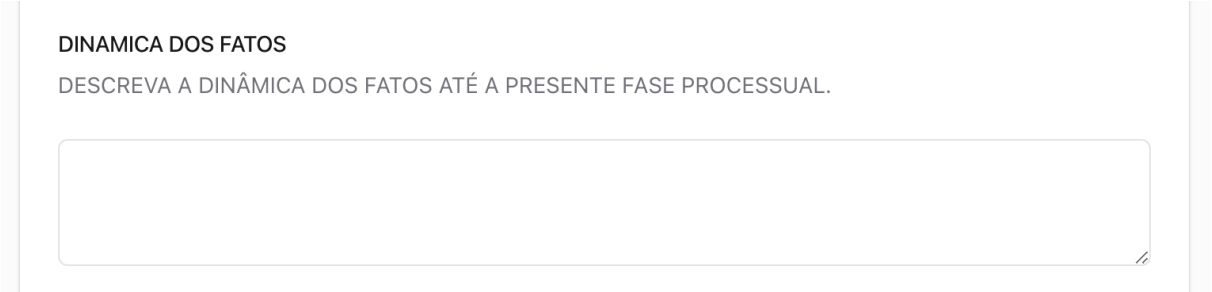
Fonte: Elaborado pelo autor, 2024.

4.3 SEÇÃO: DINÂMICA DOS FATOS

- **Descrição:** Campo de texto amplo para descrever a sequência dos fatos processuais até o momento presente.

- **Interatividade:** Caixa de texto de várias linhas para descrições mais completas.
- **Explicação Jurídica:** Esta seção é utilizada para construir a narrativa jurídica da defesa, destacando eventuais nulidades, inconsistências nas acusações ou aspectos que justifiquem a inocência do acusado.

Figura 25 – Recorte da interface - Seção Dinâmica dos Fatos.

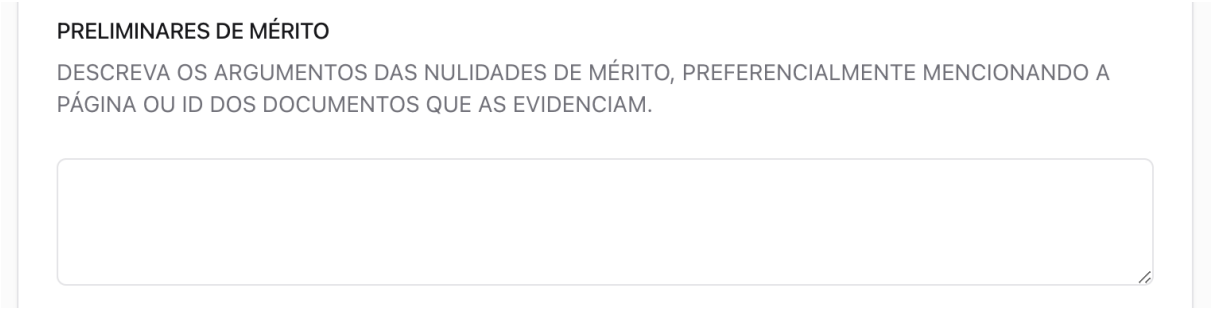


Fonte: Elaborado pelo autor, 2024.

4.4 SEÇÃO: PRELIMINARES DE MÉRITO

- **Descrição:** Parte destinada a relatar argumentos de nulidades processuais.
- **Campos:**
  - **Descrição:** Caixa de texto de várias linhas para inserir argumentos detalhados sobre eventuais nulidades processuais, como falhas na condução do processo ou violações de direitos.
  - **Checkboxes:** Teses jurídicas comuns em nulidades preliminares:
    - **Invasão Domiciliar:** Alegação de entrada ilegal na residência, violando o direito constitucional à inviolabilidade do domicílio.
    - **Abordagem/Busca Pessoal:** Questiona a legalidade de abordagens feitas sem mandado ou justificativa plausível.
    - **Cerceamento de Defesa:** Aponta limitações impostas à defesa, como não acesso a provas ou testemunhas.
    - **Reconhecimento Ilegal:** Contestações sobre identificação do acusado feita sem seguir os procedimentos legais.
    - **Ausência de Fundamentação:** Questiona decisões ou atos judiciais que não apresentam justificativa clara e detalhada.
- **Explicação Jurídica:** A apresentação de nulidades pode resultar no reconhecimento de vícios processuais e na anulação parcial ou total do processo.

Figura 26 – Recorte da interface - Seção Preliminares de Mérito.



Fonte: Elaborado pelo autor, 2024.

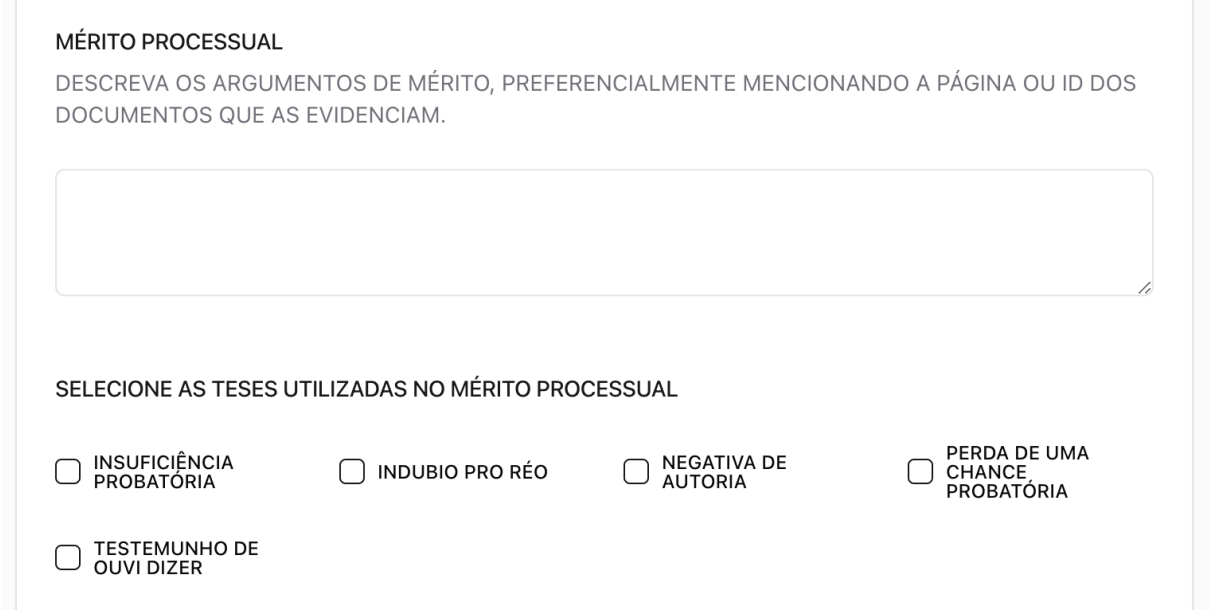
4.5 SEÇÃO: MÉRITO PROCESSUAL

- **Descrição:** Parte dedicada aos argumentos de mérito processual.
- **Campos:**
  - **Descrição:** Caixa de texto para inserir argumentos que busquem desqualificar as acusações ou reforçar a defesa, com base nas provas e fatos apresentados.
  - **Checkboxes:** Teses jurídicas comuns no mérito:
    - **Insuficiência Probatória:** Argumento de que as provas apresentadas são insuficientes para condenação.
    - **Indubio Pro Réo:** Princípio jurídico que favorece o réu na dúvida, garantindo que ninguém seja condenado sem certeza.
    - **Negativa de Autoria:** Alegação de que o acusado não foi o autor do crime.
    - **Perda de uma Chance Probatória:** Falha na obtenção de provas relevantes por parte da acusação ou na preservação de direitos probatórios.
    - **Testemunho de Ouvi Dizer:** Questionamento de provas baseadas em relatos indiretos, que carecem de credibilidade jurídica.
- **Explicação Jurídica:** Nesta etapa, o advogado busca evidenciar a fragilidade das acusações, reforçando os argumentos da defesa.

4.6 SEÇÃO: DOS PEDIDOS E REQUERIMENTOS

- **Descrição:** Permite listar pedidos ou requerimentos a serem incluídos na petição.
- **Campos:**
  - **Pedidos/Requerimentos:** Campo de texto para cada item da lista.

Figura 27 – Recorte da interface - Seção Mérito Processual.

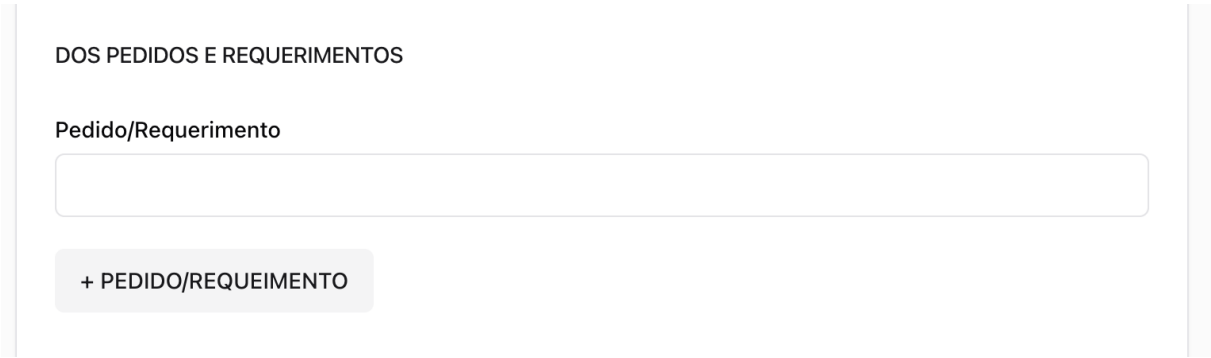


Fonte: Elaborado pelo autor, 2024.

  - **Ações:** Botão para adicionar novos pedidos ou remover existentes.

- **Explicação Jurídica:** O advogado formaliza nesta seção as solicitações ao juiz, como absolvição, nulidade de atos ou outros benefícios legais.

Figura 28 – Recorte da interface - Seção Dos Pedidos e Requerimentos.



Fonte: Elaborado pelo autor, 2024.

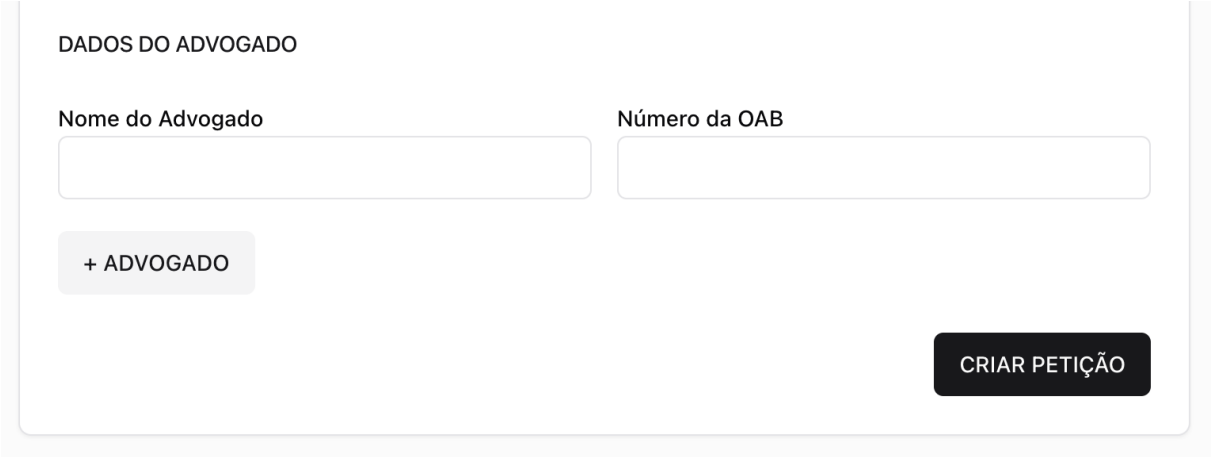
4.7 SEÇÃO: DADOS DO ADVOGADO

- **Descrição:** Espaço para incluir informações dos advogados responsáveis.
- **Campos:**
  - **Nome do Advogado:** Campo de texto para cada advogado.

  - **Número da OAB:** Campo numérico para a identificação profissional.
  - **Ações:** Botões para adicionar ou remover advogados da lista.

- **Explicação Jurídica:** A correta identificação do advogado garante a legitimidade da representação legal e evita questionamentos formais.

Figura 29 – Recorte da interface - Seção Dados do Advogado.



Fonte: Elaborado pelo autor, 2024.

4.8 BOTÃO FINAL

- **Descrição:** Botão de ação para criar a petição com os dados preenchidos.
- **Texto do Botão:** "Criar Petição".

4.9 EXIBIÇÃO DA PEÇA GERADA

Após o preenchimento do formulário e o clique no botão *Criar Petição*, o sistema processa os dados fornecidos pelo usuário e os documentos anexados, gerando automaticamente uma peça jurídica. O documento gerado é exibido ao usuário em uma interface amigável para revisão antes de ser enviado ao tribunal.

4.9.1 Fluxo de Geração e Exibição

O fluxo de geração e exibição do documento segue as etapas descritas abaixo:

1. **Envio de Dados:** O usuário preenche as informações no formulário e anexa os documentos relevantes. Após clicar no botão *Criar Petição*, todos os dados são enviados ao backend para processamento.

2. **Geração do Documento:** No backend, as informações são analisadas e combinadas com os anexos. O documento jurídico é gerado automaticamente em formato `markdown`.
3. **Exibição ao Usuário:** O documento gerado é enviado ao frontend, onde é renderizado e exibido em uma interface clara e estruturada. Essa exibição permite ao usuário revisar o texto final antes de submetê-lo.

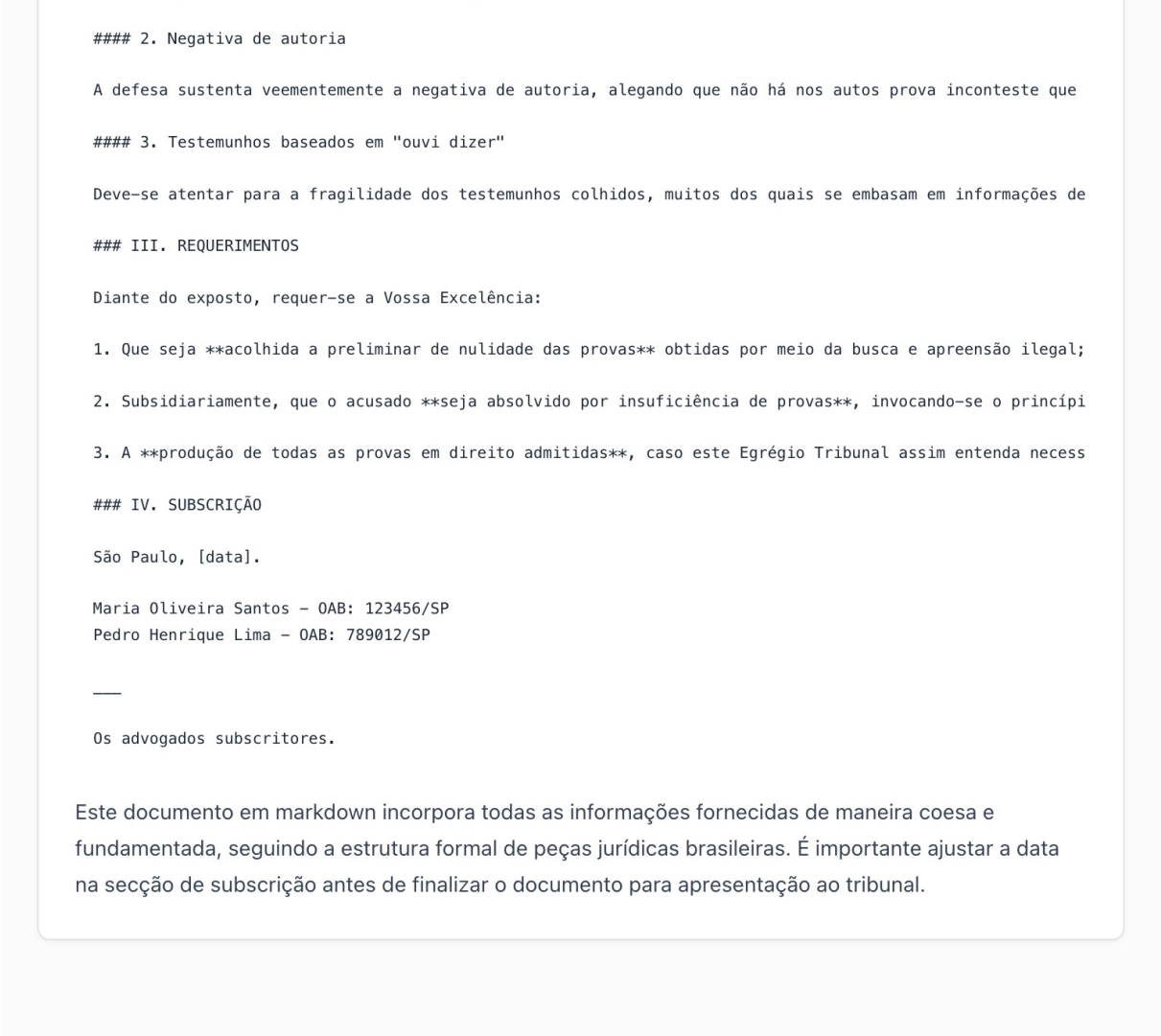
Figura 30 – Exibição da peça jurídica gerada na interface do usuário (Parte 1).

Fonte: Elaborado pelo autor, 2024.

4.9.2 Interface da Exibição

A interface de exibição é composta por dois elementos principais:

Figura 31 – Exibição da peça jurídica gerada na interface do usuário (Parte 2).



```
#### 2. Negativa de autoria

A defesa sustenta veementemente a negativa de autoria, alegando que não há nos autos prova inconteste que

#### 3. Testemunhos baseados em "ouvi dizer"

Deve-se atentar para a fragilidade dos testemunhos colhidos, muitos dos quais se embasam em informações de

### III. REQUERIMENTOS

Diante do exposto, requer-se a Vossa Excelência:

1. Que seja **acolhida a preliminar de nulidade das provas** obtidas por meio da busca e apreensão ilegal;

2. Subsidiariamente, que o acusado **seja absolvido por insuficiência de provas**, invocando-se o princípi

3. A **produção de todas as provas em direito admitidas**, caso este Egrégio Tribunal assim entenda necess

### IV. SUBSCRIÇÃO

São Paulo, [data].

Maria Oliveira Santos – OAB: 123456/SP
Pedro Henrique Lima – OAB: 789012/SP

___

Os advogados subscritores.
```

Este documento em markdown incorpora todas as informações fornecidas de maneira coesa e fundamentada, seguindo a estrutura formal de peças jurídicas brasileiras. É importante ajustar a data na secção de subscrição antes de finalizar o documento para apresentação ao tribunal.

Fonte: Elaborado pelo autor, 2024.

- **Informações de Orientação:** Uma seção inicial apresenta orientações gerais para o usuário sobre como revisar o documento gerado e destaca a importância de verificar todos os detalhes antes do envio.
- **Área de Exibição do Documento:** O documento é exibido em uma área central, utilizando um estilo limpo e legível. Para isso, o conteúdo em `markdown` é renderizado em HTML, mantendo o formato adequado para peças jurídicas.

4.9.3 Fluxo do Usuário

O fluxo de interação do usuário nesta etapa pode ser descrito como:

1. **Revisão do Documento:** O usuário analisa o documento gerado, verificando os argumentos, requerimentos e a formatação geral da peça.

2. **Ajustes Necessários:** Caso o documento apresente inconsistências ou informações ausentes, o usuário pode retornar ao formulário, realizar os ajustes e gerar uma nova versão.
3. **Finalização:** Após revisar e validar a peça, o usuário pode avançar com os próximos passos, como exportar ou enviar o documento ao tribunal.

4.9.4 Facilidade de Uso

A interface foi projetada para garantir uma experiência intuitiva e eficiente, com ênfase nos seguintes aspectos:

- **Legibilidade:** O uso de um estilo claro e padronizado facilita a leitura e revisão da peça jurídica.
- **Navegação Simples:** O uso de *breadcrumbs* permite ao usuário retornar facilmente às etapas anteriores do processo.
- **Feedback Imediato:** A exibição do documento gerado logo após o envio dos dados dá ao usuário a oportunidade de validar rapidamente as informações fornecidas.

5 CONSIDERAÇÕES FINAIS

Este capítulo apresenta uma síntese do desenvolvimento do protótipo de sistema web para a geração automatizada de alegações finais, destacando os resultados obtidos, as limitações identificadas e as possibilidades de evolução futura. O repositório do projeto está disponível em: https://github.com/lucasxpto/juriz.

5.1 RESULTADOS OBTIDOS

O protótipo demonstrou ser tecnicamente viável, validando os fluxos principais propostos. Os resultados observados podem ser diretamente relacionados aos objetivos específicos delineados no início deste trabalho:

a) O cadastro eficaz de informações processuais — como endereçamento, número do processo e nome do acusado — atende ao objetivo de desenvolver uma plataforma que permita o registro completo dos dados jurídicos (Objetivo Específico **a**).

b) A funcionalidade de *upload* de documentos em formato PDF, com processamento e armazenamento adequados, está alinhada aos objetivos de permitir o envio de anexos e gerenciar adequadamente os dados no banco (Objetivos Específicos **a** e **e**).

c) A geração automatizada de alegações finais utilizando a OpenAI API, com base nas informações fornecidas pelo usuário, concretiza o núcleo da proposta, conforme previsto na integração com serviços de IA (Objetivos Específicos **d** e **f**).

d) A exibição dos documentos gerados no formato Markdown, de forma clara e organizada, contribui para a experiência do usuário, alinhando-se ao objetivo de desenvolver uma interface interativa e intuitiva (Objetivo Específico **c**).

Os resultados preliminares indicam que o sistema é capaz de atender aos requisitos funcionais definidos, com boas perspectivas de aprimoramento para aumentar sua robustez, escalabilidade e flexibilidade.

5.2 LIMITAÇÕES DO SISTEMA

Como um protótipo inicial, o sistema apresenta algumas limitações que podem ser endereçadas em versões futuras:

a) Dependência da OpenAI API para a geração de textos, o que pode ser um ponto crítico em casos de instabilidade do serviço.

b) Necessidade de maior flexibilidade para personalizar os textos gerados, especialmente em documentos jurídicos mais complexos.

c) Ausência de suporte otimizado para dispositivos móveis, limitando a acessibilidade em diferentes plataformas.

Essas limitações não comprometem a funcionalidade básica do sistema, mas representam oportunidades para melhorias e expansão.

5.3 POSSIBILIDADES DE EVOLUÇÃO

As perspectivas de evolução do sistema incluem:

a) Adicionar funcionalidades para customização avançada dos textos gerados, ampliando sua aplicação em diferentes contextos jurídicos.

b) Implementar soluções de *cache* para reduzir a dependência em tempo real da OpenAI API.

c) Desenvolver uma interface adaptada para dispositivos móveis, garantindo maior acessibilidade.

d) Ampliar o escopo do sistema para abranger outras etapas processuais, como petições iniciais e recursos, fortalecendo sua utilidade.

e) Realizar otimizações de desempenho e usabilidade com base em estudos e testes contínuos.

Essas melhorias visam tornar o sistema mais robusto, flexível e adequado às demandas do setor jurídico.

5.4 CONCLUSÃO

O protótipo desenvolvido representa uma iniciativa promissora no uso de tecnologias modernas para automação de tarefas jurídicas. Sua capacidade de gerar alegações finais de forma estruturada e automatizada demonstra o potencial de IA e ferramentas *LegalTech* para otimizar processos e aumentar a eficiência no setor jurídico.

As limitações identificadas são comuns a projetos em fase inicial e podem ser abordadas por meio de melhorias técnicas e estudos adicionais. Com as evoluções planejadas, espera-se que o sistema se consolide como uma ferramenta valiosa para advogados e profissionais do Direito, contribuindo para a modernização e acessibilidade na prática jurídica.

REFERÊNCIAS

GeoLigard. **Using Laravel to Interact with OpenAI's Assistant API with Vision**. [*S.l.*], 2025. Disponível em: <https://geoligard.com/using-laravel-to-interact-with-openai-s-assistants-api-with-vision>. Acesso em: 12 jan. 2025.

Inertia.js. **Inertia.js - The Modern Monolith**. [*S.l.*], 2025. Disponível em: <https://inertiajs.com>. Acesso em: 12 jan. 2025.

Jurídico Ágil. **Inteligência Artificial e ChatGPT para Advogados**. [*S.l.*: *s.n.*], 2025. Canal do YouTube. Disponível em: <https://www.youtube.com/@juridicoagil>. Acesso em: 12 jan. 2025.

Laravel. **Laravel - The PHP Framework for Web Artisans**. [*S.l.*], 2025. Disponível em: <https://laravel.com>. Acesso em: 12 jan. 2025.

MySQL. **MySQL Documentation - The world's most popular open source database**. [*S.l.*], 2025. Disponível em: <https://dev.mysql.com/doc/>. Acesso em: 12 jan. 2025.

OpenAI. **OpenAI - Artificial Intelligence for Everyone**. [*S.l.*], 2025. Disponível em: <https://openai.com>. Acesso em: 12 jan. 2025.

OpenAI PHP. **OpenAI PHP - Biblioteca para integração com a API da OpenAI**. [*S.l.*], 2025. Disponível em: <https://github.com/openai-php>. Acesso em: 12 jan. 2025.

ShadCN Vue. **ShadCN Vue - Accessible Vue 3 UI components**. [*S.l.*], 2025. Disponível em: <https://www.shadcn-vue.com>. Acesso em: 12 jan. 2025.

Tailwind CSS. **Tailwind CSS - A utility-first CSS framework**. [*S.l.*], 2025. Disponível em: <https://tailwindcss.com>. Acesso em: 12 jan. 2025.

Vue.js. **Vue.js - The Progressive JavaScript Framework**. [*S.l.*], 2025. Disponível em: <https://vuejs.org>. Acesso em: 12 jan. 2025.