Unioeste - Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Docente: Adriana Postal

Sistema Especialista

Geração de Contratos Imobiliários com Cláusulas Relevantes ao Contexto do Cliente

Kawan de Oliveira

Luiz Eduardo Garzon de Oliveira

Pedro Henrique de Oliveira Berti

Weberson Morelli Leite Junior

CASCAVEL

2025

1. INTRODUÇÃO

Os sistemas especialistas são uma área da inteligência artificial voltada para a simulação do conhecimento humano em domínios específicos. Eles utilizam bases de conhecimento e mecanismos de inferência para resolver problemas e tomar decisões, sendo amplamente empregados em diversas áreas, incluindo a medicina, engenharia e direito.

Este trabalho apresenta o desenvolvimento de um sistema especialista para a geração automatizada de contratos imobiliários, utilizando a Shell CLIPS. O sistema permite a modelagem de contratos de acordo com informações fornecidas pelo usuário, garantindo a inclusão de cláusulas apropriadas ao contexto do contrato, além de identificar quais itens da cláusula devem ser inseridos. A solução proposta busca otimizar o tempo de elaboração desses documentos e garantir maior precisão e conformidade jurídica.

1.1. DOMÍNIO DE CONHECIMENTO ESCOLHIDO

O domínio de conhecimento abordado neste trabalho é o direito imobiliário, com foco na elaboração de contratos imobiliários. O sistema desenvolvido permite a modelagem automatizada de contratos, assegurando que as cláusulas incluídas sejam adequadas ao contexto específico de cada cliente.

1.2. ÁREA DE ATUAÇÃO

O sistema especialista atua na automatização da criação de contratos imobiliários, sendo voltado para advogados, imobiliárias e corretores. O sistema possibilita a geração eficiente de documentos jurídicos, otimizando processos burocráticos e garantindo conformidade com normas jurídicas.

1.3. **OBJETIVOS DO SISTEMA**

O principal objetivo do sistema é automatizar a geração de contratos imobiliários personalizados. Para isso, ele:

- Analisa as informações fornecidas pelo usuário;
- Determina as cláusulas apropriadas utilizando regras de produção;
- Avalia quais itens das cláusulas devem ser inseridos;
- Gera um contrato final no formato HTML, com opção de exportação para DOCX e PDF;

1.4. **ESPECIALISTA DO PROJETO**

Para garantir a validade jurídica do conhecimento incorporado ao sistema, foi consultada a Dra. Fernanda Garzon de Oliveira, advogada empresarial no escritório Taube Toretta de Cascavel/PR.

- **Formação:** Bacharel em Direito pela UNIVEL – Cascavel-PR (2018).
- **Especialização:** Pós-graduanda em Direito Penal e Processo Penal.
- **Registro Profissional:** OAB/PR 118.849.

A especialista auxiliou na definição das cláusulas contratuais, revisão jurídica do sistema e validação das regras de produção.

2. **DESENVOLVIMENTO**

O desenvolvimento do sistema especialista para geração de contratos imobiliários seguiu uma abordagem estruturada, envolvendo a aquisição de conhecimento jurídico, definição de regras de inferência e implementação de um modelo computacional capaz de processar informações fornecidas pelo usuário e gerar contratos personalizados.

Inicialmente, foi realizada a coleta e organização das informações essenciais para a construção da base de conhecimento do sistema. Em seguida, foram definidas regras de produção que permitissem a seleção automática das cláusulas mais adequadas para cada tipo de contrato. A implementação envolveu a utilização da Shell CLIPS, integrada a um backend em Python e uma interface web interativa.

2.1. **PROCESSO DE AQUISIÇÃO DE CONHECIMENTO**

O processo de aquisição de conhecimento para o desenvolvimento do sistema especialista envolveu a coleta, análise e estruturação de informações jurídicas relacionadas à elaboração de contratos imobiliários. Para isso, o integrante Luiz Eduardo Garzon de Oliveira foi à casa da especialista da área, Dra. Fernanda Garzon de Oliveira, que forneceu documentação técnica e orientações sobre a construção das cláusulas contratuais.

Inicialmente, foram reunidos seis tipos de contratos imobiliários, cada um contendo dez cláusulas essenciais. Os contratos analisados foram:

- Contrato de Compra e Venda de Imóvel;
- Contrato de Locação de Imóvel;
- Contrato de Comodato de Imóvel;
- Contrato de Permuta de Imóvel;
- Contrato de Cessão de Direitos sobre Imóvel;
- Contrato de Prestação de Serviços

A especialista foi responsável por validar as cláusulas utilizadas, garantindo que fossem juridicamente adequadas e que pudessem ser generalizadas para diferentes contextos. Durante esse processo, também foram identificadas cláusulas universais, que aparecem em diversos contratos e podem ser aplicadas de forma ampla dentro do sistema.

Com base nessa estrutura, foi criada a base de conhecimento do sistema, composta por regras de produção que permitem ao mecanismo de inferência selecionar automaticamente as cláusulas apropriadas para cada contrato, conforme as respostas fornecidas pelo usuário.

2.2. **REGRAS UTILIZADAS NA IMPLEMENTAÇÃO**

A implementação do sistema especialista foi baseada em regras de produção, utilizando a Shell CLIPS como motor de inferência. As regras seguem o formato IF-THEN (Se-Então), permitindo que o sistema tome decisões automaticamente com base nas respostas fornecidas pelo usuário.

A lógica de funcionamento do sistema baseia-se na identificação das características do contrato, inferindo quais cláusulas devem ser inseridas a partir das respostas dadas pelo usuário a um conjunto de perguntas. Cada pergunta corresponde a uma variável da base de conhecimento, e as regras são responsáveis por relacionar essas variáveis às cláusulas apropriadas.

O documento CLAUSULAS, localizado em:

- **BASE DE CONHECIMENTO → CLAUSULAS**

Contém as regras de utilização de cada cláusula dentro dos contratos, considerando os seguintes aspectos:

- Quando utilizar a cláusula?;
- Como utilizá-la?;
- Em quais contratos a cláusula pode ser aplicada?;
- Como generalizar a cláusula para que seja compatível com os 6 contratos abordando diferentes contextos?;
- Quais variáveis são relevantes para a escolha da cláusula pelo advogado, considerando cada contexto contratual?;
- Qual o contexto da cláusula, esclarecendo sua função e aplicação dentro do contrato.

2.2.1. **VARIÁVEIS UTILIZADAS NA INFERÊNCIA**

- **tipo_contrato:** qual o tipo de contrato está sendo criado? (tipo categórica: os 6 tipos de contratos modelados neste trabalho).

- **objetivo_especifico:** Há um bem (imóvel, serviço ou direito) específico sendo negociado? O usuário descreveu o bem/serviço/direito? (Tipo Booleano).

- **partes_definidas:** As partes (cedente/cessionário, comprador/vendedor etc.) foram identificadas? (Tipo Booleano).

- **tempo_duracao:** O contexto do contrato necessita a determinação de condições sobre prazos? (Tipo Booleano).

- **existe_pagamento:** Existe alguma contraprestação financeira? (Tipo booleano).

- **penalidade_prevista:** Há previsão de multa ou penalidade pela infração e Descumprimento dos direitos aplicáveis negociados entre as partes? (Tipo Booleano).

- **aviso_previo:** É necessário a notificação sobre avisos prévios para situações específicas? (Tipo Booleano).

- **localizacao_partes:** Existe a necessidade sobre a identificação sobre a residência das partes e a região responsável por resoluções jurídicas e de eventuais disputas relacionadas ao contrato? (Tipo Booleano).

- **observacoes_finais:** Existe a necessidade da inclusão de observações sobre a validade do contrato entre outras questões gerais? (Tipo Booleano).

- **ajustes_futuros:** As partes desejam incluir uma previsão de possibilidade de ajustes futuros ao contrato? (Tipo Booleano).

- **confirmacao_legislacao:** Existe a necessidade de observações sobre a interpretação, prevalência de termos e legislação ou regulamentação específica aplicável ao contrato? (Tipo Booleano).

- **exige_garantia:** O contrato exige alguma garantia formal em relação a riscos relevantes de inadimplência? (Tipo Booleano).

- **tipo_garantia:** O contrato deve possuir algum tipo de garantia? (caução, fiador, seguro fiança etc.) (Tipo Booleano).

- **interesse_preferencia:** Existe a possibilidade do interesse de uma das partes em ter maior prioridade caso o bem seja negociado? (Tipo Booleano).

- **meio-comunicacao:** Existe a necessidade de estabelecer um meio de comunicação que será utilizado para notificações (e-mail, carta registrada, ambos)? (Tipo Booleano).

- **natureza_objeto_especializada:** A obrigação contratual exige mão de obra ou serviços de terceiros especializados? (Tipo Booleano).

- **permite_subcontratacao:** Existe a possibilidade da aceitação que o contratado utilize terceiros para executar suas obrigações? (Tipo Booleano).

- **complexidade_execucao:** A execução das obrigações é complexa e envolve múltiplas etapas? (Tipo Booleano).

- **riscos_externos:** Existe a possibilidade que o objeto do contrato seja prejudicado devido a riscos de fatores externos (greves, crises, pandemias)? (Tipo Booleano).

- **duracao_contrato:** O contrato possui tempo de duração relevante envolvendo a execução de obrigações contínuas que são passíveis de alta formalidade e proteção jurídica? (Tipo Booleano).

- **coleta_informacao:** O contrato necessita aborda a existência da coleta de informações relevantes como dados cadastrais ou documentais fornecidas pelas partes? (Tipo Booleano).

- **veracidade_informacao:** O contrato exige formalização jurídica robusta para validação de informações sobre todos os dados informados e suas dependências? (Tipo Booleano).

- **condicoes_uso:** O contrato deve prever a definição sobre a utilização sobre o objeto negociado além de associações e exigências específicas que devem ser respeitadas relacionadas ao uso do bem? (Tipo Booleano).

- **restricoes_objeto:** O contrato necessita determinar restrições e exigências sobre questões de utilização, ambientalismo, zoneamento e demais advertências específicas a respeito de continências sobre o objeto? (Tipo Booleano).

- **compromissos_danos:** O contrato deve estabelecer as diretrizes sobre a responsabilidade das partes por eventuais danos sobre o objeto? (Tipo Booleano).

- **risco_danos:** Existe risco potencial de danos ao bem, imóvel ou serviço? (Tipo Booleano).

- **peridiocidade:** O contrato necessita prever condições de prazo, execução continuada ou periódica e termos especiais para renovação? (Tipo Booleano).

- **transferencia_sucessao:** Existe a necessidade de definir obrigações e contenções sobre transferências de direitos e sucessão do objeto sobre as partes envolvidas? (Tipo Booleano).

- **obrigacoes_tributarias:** O contrato necessita definições formais sobre responsabilidades tributárias das partes envolvidas relacionadas a natureza do objeto? (Tipo Booleano).

- **natureza_juridica:** A natureza do contrato exige o englobamento de questões tributárias específicas a respeito de questões fiscais jurídicas? (Tipo Booleano).

2.3. **ESTRATÉGIAS DE INFERÊNCIA E REPRESENTAÇÃO DO CONHECIMENTO**

Tabela 1 – Estrutura do Sistema Especialista para Geração de Contratos Imobiliários

Característica	Descrição
Estratégia de Inferência	Encadeamento para frente (Forward Chaining) – As cláusulas do contrato são selecionadas progressivamente com base nas respostas do usuário.
Tipo de Raciocínio	Dedutivo – O sistema aplica regras pré-definidas para inferir quais cláusulas devem ser incluídas.
Representação do Conhecimento	Regras de Produção (IF-THEN) – A base de conhecimento do sistema é composta por regras que definem quais cláusulas devem ser aplicadas a cada contrato.

2.4. **DETALHES ADICIONAIS SOBRE O DESENVOLVIMENTO**

A implementação do sistema foi dividida em três partes principais.

2.4.1. **BASE DE CONHECIMENTO**

No sistema desenvolvido, a base de conhecimento foi estruturada com base em regras de produção e organizada para garantir flexibilidade, escalabilidade e precisão na seleção das cláusulas contratuais.

A base de conhecimento foi implementada utilizando a Shell CLIPS, que permite a modelagem do conhecimento por meio de fatos e regras de produção. A estrutura foi organizada em três partes principais:

- **Fatos (Facts) –** Representam as informações fornecidas pelo usuário, como o tipo de contrato e as características envolvidas.
- **Regras (Rules) –** Definem a lógica de seleção das cláusulas contratuais com base nos fatos informados.
- **Base Externa de Cláusulas (JSON) –** Contém o texto completo das cláusulas, separando a lógica de inferência do conteúdo textual dos contratos.

Os fatos representam os dados coletados a partir das respostas do usuário e são armazenados na base de conhecimento para serem utilizados pelo motor de inferência. Cada fato é modelado como um atributo que pode assumir diferentes valores.

No CLIPS, os fatos são declarados por meio de deftemplates, que funcionam como "estruturas" para armazenar múltiplos atributos. Exemplo da estrutura utilizada:

```
(deftemplate contrato
    (slot tipo-contrato)
    (slot objetivo-especifico)
    (slot partes-definidas)
    (slot tempo-duracao)
    (slot existe-pagamento)
    (slot penalidade-prevista)
    (slot aviso-previo)
    (slot localizacao-partes)
    (slot observacoes-finais)
    (slot ajustes-futuros)
    (slot confirmacao-legislacao)
    (slot exige-garantia)
    (slot tipo-garantia)
    (slot interesse-preferencia)
    (slot meio-comunicacao)
    (slot natureza-objeto-especializada)
    (slot permite-subcontratacao)
    (slot complexidade-execucao)
    (slot riscos-externos)
    (slot duracao-contrato)
    (slot coleta-informacao)
    (slot veracidade-informacao)
    (slot condicoes-uso)
    (slot restricoes-objeto)
    (slot compromissos-danos)
    (slot riscos-danos)
    (slot peridiocidade)
    (slot transferencia-sucessao)
    (slot obrigacoes-tributarias)
    (slot natureza-juridica)
)
```

Abaixo temos um exemplo de regra implementada no CLIPS:

No CLIPS, as regras são definidas com defrule, que verifica fatos e dispara ações para incluir novas cláusulas. Para adicionar fatos ao sistema, utilizamos assert, desde que a regra seja considerada válida.

A seleção das cláusulas e seus itens segue o modelo **"cláusula + itens separados"**, onde:

- O primeiro assert indica a cláusula a ser incluída.
- O segundo assert especifica quais itens desta cláusula serão adicionados.

Os nomes das regras seguem o padrão: **incluir-clausula-nome-da-clausula-(completo ou parcial n).** A avaliação da regra considera os fatos presentes e ausentes que são relevantes para a cláusula, inferindo assim a seleção apropriada.

Por padrão, no CLIPS, a conjunção lógica ‘E’ é assumida. Ou seja, ao escrever múltiplas condições dentro de uma regra, o sistema interpreta automaticamente que todas devem ser verdadeiras para que a regra seja ativada.

```
;CLAUSULA PRAZO
(defrule incluir-clausula-prazo-completo
    (contrato-valido)
    (contrato (partes-definidas sim))
    (contrato (tempo-duracao sim))
    (contrato (peridiocidade sim))
    =>
    (assert (clausula-incluida prazo))
    (assert (clausula-itens prazo completo))
)

(defrule incluir-clausula-prazo-parcial-1
    (contrato-valido)
    (contrato (partes-definidas sim))
    (contrato (tempo-duracao sim))
    (not (contrato (peridiocidade sim)))
    =>
    (assert (clausula-incluida prazo))
    (assert (clausula-itens prazo item1 item3))
)

(defrule incluir-clausula-prazo-parcial-2
    (contrato-valido)
    (contrato (partes-definidas sim))
    (contrato (peridiocidade sim))
    (not (contrato (tempo-duracao sim)))
    =>
    (assert (clausula-incluida prazo))
    (assert (clausula-itens prazo item2 item4))
)
```

2.4.2. **DESENVOLVIMENTO DO BACKEND**

Com a base de conhecimento pronta o backend foi implementado como responsável por enviar as respostas do usuário (respondidas em um formulário frontend) para o motor de inferência do CLIPS, e depois pegar a resposta do CLIPS (cláusulas e seus respectivos itens) e inserir dentro de um documento HTML modelando o contrato especificado e apresentá-lo no frontend.

2.4.2.1. **APP.PY**

O backend foi implementado utilizando uma API Flask em Python. Esse servidor local é responsável por receber as informações do frontend no formato JSON.

O JSON recebido é enviado para o motor do CLIPS, que executa o processo de inferência sobre a base de conhecimento. Como resultado, as cláusulas e itens selecionados são extraídos, convertidos novamente em JSON e enviados para o gerador do contrato em HTML.

Com o contrato modelado, a API retorna o HTML final para o frontend.

2.4.2.2. **MOTOR_CLIPS**

O backend, além da API Flask, também contém o código responsável pelo motor de inferência do CLIPS. Para isso, ele utiliza a biblioteca clipspy, que permite executar o motor de inferência CLIPS diretamente no Python.

Com clipspy, não é necessário ter o CLIPS instalado na máquina nem acessá-lo como um subprocesso, tornando a integração mais eficiente e simplificada.

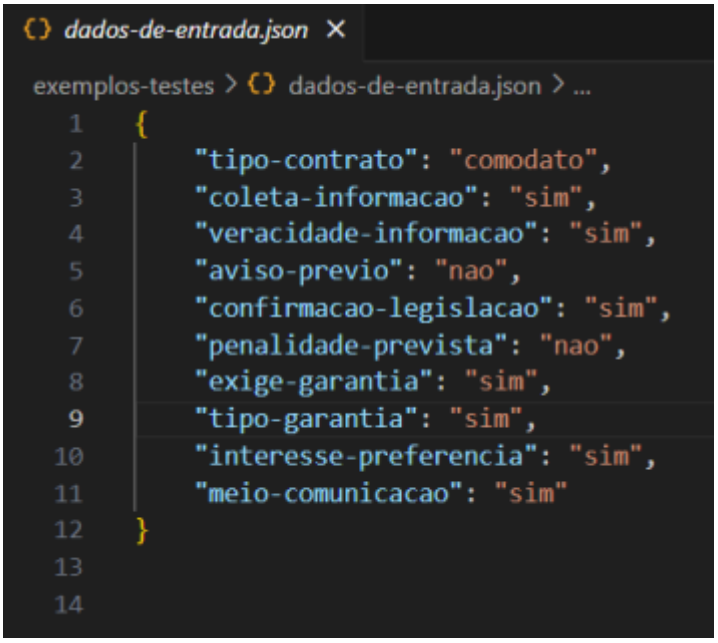
Neste trabalho, testamos todas as formas possíveis de integrar o CLIPS ao Python:

- **CLIPS como subprocesso** → Ineficiente, apresentou muitos problemas de compatibilidade e exige que o CLIPS esteja instalado na máquina.
- **Biblioteca pyclips** → Desatualizada e descontinuada no Python 2.
- **Biblioteca clipspy** → A única opção que funcionou corretamente para a integração do CLIPS com Python.

Optamos pelo clipspy, pois foi a única solução viável. No entanto, essa biblioteca não está atualizada para versões do Python 3.9+, e no Windows podem ocorrer problemas de compatibilidade.

O clipspy se mostrou 100% funcional utilizando Python 3.8.18 em Linux ou no ambiente WSL no Windows. Isso não é necessariamente um problema, já que em uma hospedagem real do sistema, um ambiente virtual poderia ser configurado com todas as dependências necessárias.

O motor_clips recebe da API as respostas do frontend em formato JSON:



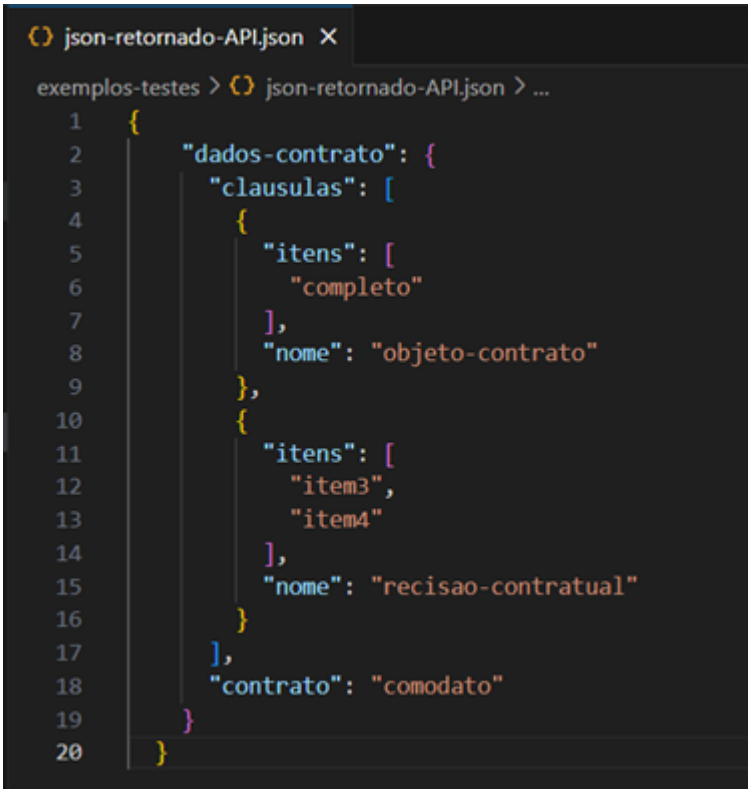
Esses dados do JSON representam os fatos dentro da base de conhecimento.

O motor_clips é responsável por receber esse JSON e, para cada item, convertê-lo para o formato adequado, conforme a estrutura da base de conhecimento.

Com os fatos montados no formato esperado, o motor de inferência pode processá-los corretamente e gerar as conclusões necessárias.

```
(contrato
    (tipo-contrato comodato)
    (coleta-informacao sim)
    (veracidade-informacao sim)
    (aviso-previo nao)
    (confirmacao-legislacao sim)
    (penalidade-prevista nao)
)
```

É executado a inferência pelo motor do CLIPS. O CLIPS retorna os fatos inferidos contendo as cláusulas e quais itens serão inseridos, esses fatos são novamente convertidos para um formato JSON no seguinte estilo:



Após o motor_clips realizar a inferência, ele retorna um JSON com os resultados para a API. O código responsável por modelar o contrato HTML pode ser encontrado em:

- **Implementacao → backend → gerador_html.py**

Esse código recebe o JSON com as respostas da inferência e, a partir disso, modela o contrato HTML.

Cada contrato foi convertido em um modelo HTML, permitindo ao usuário exportá-lo como um documento PDF ou DOCX. Os modelos podem ser encontrados na pasta:

- **BASE DE CONHECIMENTO -> MODELOS HTML**

Para cada tipo de contrato, são inseridos os respectivos cabeçalhos, rodapés e cláusulas obrigatórias. Em seguida, os dados do JSON são extraídos para identificar quais cláusulas e itens devem ser incluídos no contrato final.

O JSON retornado pelo motor_clips contém apenas:

- Os identificadores das cláusulas e itens;
- O tipo de contrato.

O backend, por sua vez, possui outro arquivo JSON que contém as descrições completas das cláusulas:

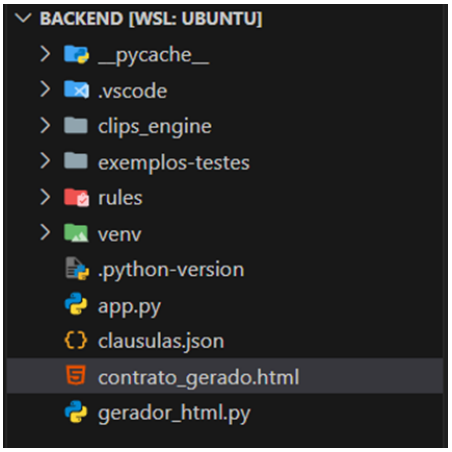
Implementacao → backend → clausulas.json

Esse arquivo inclui:

- O texto completo das cláusulas;
- Tags HTML embutidas para estilização.

O código gerador_html.py faz a relação entre os identificadores retornados pelo motor_clips e as cláusulas armazenadas no clausulas.json. Com isso, ele insere os elementos correspondentes no HTML que está sendo gerado.

Por fim, todas as informações são concatenadas para formar o documento final, que é retornado à API. Além disso, o código salva uma cópia do contrato HTML na pasta backend para conferência e validação.



O contrato é retornado pela API onde quem vai receber é o frontend.

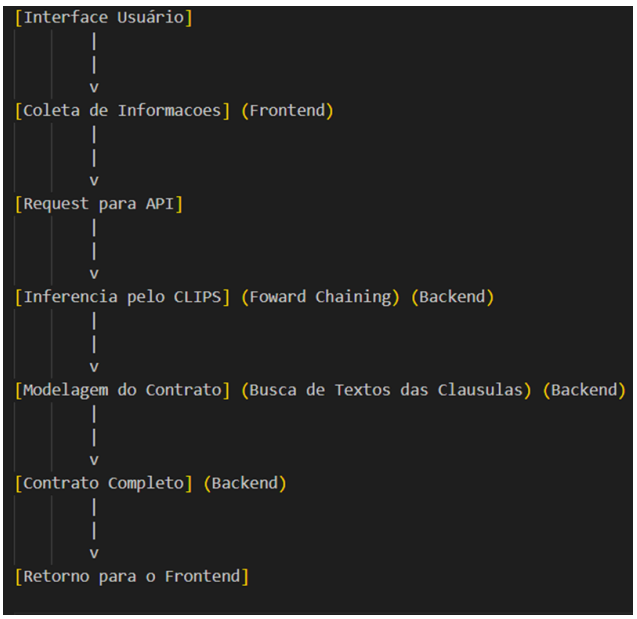
2.4.3. **DESENVOLVIMENTO DO FRONTEND**

Com o backend finalizado, implementamos o frontend utilizando JavaScript, HTML e CSS. O frontend consiste em um formulário interativo, onde o usuário responde a diversas perguntas. Cada pergunta corresponde a uma variável da base de conhecimento, que será avaliada nas regras do motor de inferência.

Fluxo de Funcionamento:

1. O usuário preenche o formulário;
2. O formulário envia uma requisição para a API, enviando as respostas no formato JSON esperado;
3. O backend processa os dados, executando a lógica de inferência;
4. A API retorna o documento HTML do contrato gerado;
5. Esse contrato é embutido no HTML da página via iframe, garantindo a preservação das tags de estilo e da lógica JS contida no modelo do contrato.

O usuário pode ajustar suas respostas e gerar novos contratos dinamicamente, com a opção de baixá-los em PDF ou DOCX.



3. PARECER DO ESPECIALISTA

PARECER TÉCNICO

Trata-se de análise técnica do software desenvolvido no âmbito acadêmico com a finalidade de criação automatizada de contratos jurídicos. Após a avaliação do sistema, verificou-se que o projeto atende satisfatoriamente ao seu propósito, permitindo a elaboração de instrumentos contratuais de maneira ágil e precisa. A interface é intuitiva e de fácil compreensão, facilitando sua utilização por profissionais da área jurídica.

No aspecto técnico-jurídico, o software se mostrou eficiente na geração de contratos personalizados, contemplando cláusulas essenciais e possibilitando ajustes conforme as necessidades específicas de cada caso. Dessa forma, o projeto cumpre sua proposta de oferecer uma ferramenta útil para a prática jurídica, demonstrando coerência entre a teoria e a aplicação prática na área do Direito Contratual.

Fernanda Garzon Oliveira

OAB/PR 118.849

4. LINKS EXTERNOS

https://github.com/EduardoGarzon/Expert-System **- Repositório do projeto**

https://drive.google.com/drive/folders/1B7COL-CZZHpdmJ_ZgkDhc84y9Ka0bEMy?usp=sharing **- Tutorial em Vídeo da Aplicação**

Obs: O manual de usuário e vídeo tutorial da aplicação encontram-se na pasta TUTORIAL.

5. **BIBLIOGRAFIA**

CLIPS. **A Tool for Building Expert Systems.** Disponível em: http://clipsrules.sourceforge.net. Acesso em: 10 mar. 2025.

DURKIN, J. **Expert Systems: Design and Development.** New York: Macmillan, 1994.

HAYES-ROTH, F.; WATERMAN, D. A.; LENAT, D. B. **Building Expert Systems.** Boston: Addison-Wesley, 1983.

JAVASCRIPT. **Free JavaScript training, resources and examples for the community.** Disponível em: <https://www.javascript.com/>. Acesso em: 10 mar. 2025.

MICROSOFT. **Visual Studio Code**. Disponível em: <https://code.visualstudio.com/>. Acesso em: 10 mar. 2025.

MITCHELL, T. **Machine Learning.** New York: McGraw-Hill, 1997.

PYTHON. **Python**. Disponível em: <https://www.python.org/>. Acesso em: 10 mar. 2025.