

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Explorador de Litígios Similares

**Um sistema de apoio à tomada de decisão para
advogados**

Yang Ricardo Barcellos Miranda

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Sistemas de Informação

Rio de Janeiro, julho de 2019



Yang Ricardo Barcellos Miranda

Explorador de Litígios Similares

Um sistema de apoio à tomada de decisão para advogados

Relatório de Projeto Final, apresentado ao programa de Graduação em Sistemas de Informação da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Gustavo Robichez de Carvalho

Rio de Janeiro

julho de 2019.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis” - José de Alencar

Agradecimentos

A PUC-Rio por conceder a bolsa de estudos integral que viabilizou a minha formação com ótimos professores durante esses anos.

Aos Programas de Inovação Tecnológica Legalite e Insurtech, que possibilitaram estar em contato com mentores e amigos fantásticos e contribuíram em vários aspectos para o meu crescimento pessoal e profissional.

Ao meu professor orientador, Gustavo Robichez de Carvalho e coorientadores Isabella Frajhof e Luiz Schirmer, pelo companheirismo e colaboração no desenvolvimento deste trabalho, assim como a todos os que estavam envolvidos nele antes de se tornar o meu projeto final de graduação.

A minha mãe, Vera Lúcia Alves de Oliveira e, minha madrinha, Regina Célia Reis de Oliveira, por todo suporte, amor e incentivo e que foram fundamentais ao longo desta jornada.

Por fim dedico ao meu pai, Ricardo Barcellos Miranda e avós, que contribuíram para a formação dos meus valores.

Resumo

Miranda, Yang Ricardo Barcellos. Carvalho, Gustavo Robichez de. eLis - Explorador de Litígios Similares: Um sistema de apoio a decisão para advogados. Rio de Janeiro, 2018. 46 p. Relatório Final de Projeto Final II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

O presente trabalho tem por objeto um sistema de apoio à tomada de decisão para advogados que permite a visualização de similaridades entre os processos judiciais julgados nos Juizados Especiais Cíveis do Tribunal de Justiça do Rio de Janeiro. A ferramenta é composta por um motor de busca de códigos de processos judiciais e outras funcionalidades que buscam auxiliar o profissional do direito na análise de documentos de processos judiciais. O processo de criação da base utilizou técnicas de ciências de dados para o entendimento e utilização dos dados disponibilizados pelo Tribunal de Justiça do Rio de Janeiro, que viabilizaram a criação de filtros de listas de dados como forma de auxílio ao processo de investigação de processos judiciais similares.

Palavras-chave

Sistema de Apoio a Decisão; Processos Judiciais; Visualização de Similaridades;

Abstract

Miranda, Yang Ricardo Barcellos. Carvalho, Gustavo Robichez de. eLis - Explorador de Litígios Similares: Um sistema de apoio a decisão para advogados. Rio de Janeiro, 2018. 46 p. Relatório Final de Projeto Final II – Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro

The objective of this study is a system developed to support lawyers on their process of decision making which allows the visualization of similar judicial proceedings judged by the Special Civil Courts of the Court of Justice of Rio de Janeiro. The tool is comprised of a search engine for judicial proceeding and other functionalities that seek to assist the legal professional on their analysis of similar lawsuits. The process of creating the database used data science techniques to understand and use the data made available by the Court of Justice of Rio de Janeiro, which enabled the creation of data list filters as a way to aid the investigation process of similar lawsuits.

Keywords

Decision Support System; Judicial Processes; Similarities Visualization

Sumário

1	INTRODUÇÃO	1
2	SITUAÇÃO ATUAL	4
3	OBJETIVOS	7
4	ATIVIDADES REALIZADAS	8
5	PROJETO E ESPECIFICAÇÃO DO SISTEMA	10
5.1	EXPLORAÇÃO DA BASE DE DADOS DO TJRJ	10
5.2	MAPEAMENTO OBJETO-RELACIONAL E API RESTFUL	11
5.3	MODELO DE AUTENTICAÇÃO COM A API RESTFUL	12
5.4	TRATAMENTO E PERSISTÊNCIA DE DADOS.	13
5.5	INTERFACE GRÁFICA	14
5.6	DESCRIÇÕES DE CASOS DE USO	15
5.6.1	CONSULTAR PROCESSO	15
5.6.2	ANALISAR PROCESSOS SIMILARES	16
5.6.3	AVALIAR PROCESSOS SIMILARES	17
5.6.4	AGRUPAR PROCESSOS SIMILARES	17
5.6.5	NAVEGAR PARA PROCESSO SIMILAR	18
5.6.6	NAVEGAR PARA SENTENÇA	19
5.6.7	LISTAR PROCESSOS SIMILARES	19
5.6.8	FILTRAR POR PARÂMETRO	20
5.6.9	VISUALIZAR GRUPO SIMILAR	21
5.7	OTIMIZAÇÃO DAS CONSULTAS	22
6	IMPLEMENTAÇÃO E AVALIAÇÃO	23
6.1	PLANEJAMENTO E EXECUÇÃO DE TESTES FUNCIONAIS	23
6.2	PLANEJAMENTO E EXECUÇÃO DE OUTROS TESTES	24
6.3	COMENTÁRIOS SOBRE A IMPLEMENTAÇÃO	25
7	CONSIDERAÇÕES FINAIS	26
	BIBLIOGRAFIA	28
	ANEXO I – MODELAGEM DE ENTIDADE E RELACIONAMENTO DOS DADOS DO TJRJ.	29
	ANEXO II – DIAGRAMA DE SEQUÊNCIA DO TRATAMENTO E PERSISTÊNCIA DOS DADOS.	30
	ANEXO III – DIAGRAMA DE CLASSES REFERENTE AO MAPEAMENTO OBJETO-RELACIONAL.	32
	ANEXO IV – DIAGRAMA DE SEQUÊNCIA DA OTIMIZAÇÃO DE CONSULTAS	33
	ANEXO V – FLUXO DE TELAS E CASOS DE USO ASSOCIADOS	34
	ANEXO VI – ÁLGEBRA RELACIONAL BÁSICAS PARA AS LISTAGENS DE PROCESSOS.	40
	ANEXO VII – MANUAL DE OPERAÇÕES DA APLICAÇÃO	42
	ANEXO VIII – CRONOGRAMA PROJETO FINAL 1	43

1 Introdução

Os tribunais de justiça brasileiros lidam com uma grande quantidade de processos. Somente no Rio de Janeiro, os Juizados Especiais Cíveis (JECs)¹, que são instâncias responsáveis por analisar processos de baixa complexidade, precisaram lidar com um total de 1,878,745 casos nos últimos cinco anos, ficando evidente a necessidade de modernizar e acelerar o processo de análise desses litígios de forma consistente. Apesar do alto número de demandas que chegam ao Judiciário, é verdade que a maior parte delas tratam de demandas repetitivas, narrando fatos idênticos ou muito similares entre si. Diante deste alto volume de processos, é exigido dos magistrados que julguem estes casos de maneira rápida e consistente.

Atualmente, para lidar com esse montante de processos no âmbito dos JECs, juízes leigos são contratados pelo sistema judiciário para compor a força de trabalho e auxiliar na tomada de decisão, elaborando projetos de sentença que são posteriormente confirmados ou revisados pelo magistrado. Estes projetos de sentenças, não entanto, não são redigidos de maneira consistente, mesmo em casos semelhantes, havendo a dispersão de questões iminentemente subjetivas (como é o caso da fixação do valor da indenização de danos morais), ou até mesmo objetivas (como o termo inicial da correção monetária).

Para lidar com esse problema, foi proposto por (SCHIRMER, ALMEIDA, et al., 2017) um modelo de jurimetria² para identificar a similaridade entre processos judiciais, que se valeu de duas técnicas de processamento de linguagem natural (NLP): o algoritmo Doc2Vec e a classificação de documentos para analisá-los. A identificação destes processos similares tem como objetivo recuperar a sentença proferida pelo magistrado em casos pretéritos, para que as mesmas possam ser utilizadas como uma minuta em novos casos semelhantes àquele analisado. Esta ferramenta busca dar apoio à tomada de decisão do magistrado, de forma que ele poderá avaliar como casos semelhantes foram julgados no passado, e oferecer uma decisão judicial consistente com este histórico, de forma que casos

¹ A Lei n. 9.099, de 26 de setembro de 1995 determina que os juizados especiais têm competência para a conciliação, o processamento, o julgamento e a execução das causas cíveis de menor complexidade (causas cujo valor não exceda a quarenta vezes o salário mínimo, por exemplo) e das infrações penais de menor potencial ofensivo, ou seja, as contravenções penais e os crimes para os quais a lei defina pena máxima não superior a dois anos. As turmas recursais, por sua vez, integradas por juízes em exercício no primeiro grau, são encarregadas de julgar recursos apresentados contra decisões dos juizados especiais.

² (NUNES, 2016) "... a disciplina do conhecimento que utiliza a metodologia estatística para investigar o funcionamento de uma ordem jurídica."

semelhantes sejam tratados de forma similar, garantindo, assim, a segurança e a previsibilidade jurídica para os jurisdicionados.

Para testar os resultados gerados pelo modelo de inferência de similaridades entre processos, foi solicitado o protótipo de uma ferramenta que realizasse a busca de processos judiciais similares entre si. Na ocasião, haviam três hipóteses de uso: (I) a identificação de fraudes ocorridas no âmbito dos Juizados Especiais Cíveis; (II) o auxílio na tomada de decisão do juiz; (II) o auxílio na tomada de decisão de advogados, sendo este último o foco da aplicação desenvolvida neste projeto.

Baseado na terceira hipótese, buscou-se compreender como a aplicação poderia ser adaptada ao contexto de apoio à decisão de advogados. Dada a necessidade de elaborar uma petição inicial, uma peça de defesa, ou um recurso, é frequente que os advogados utilizem modelos que já foram previamente elaborados em casos semelhantes. No entanto, achar estes modelos similares pode ser extremamente custoso, pois não é usual que escritórios de advocacia ou departamentos jurídicos invistam em recursos tecnológicos para facilitar esta busca.

Para realizar esta pesquisa, os advogados frequentemente dependem da sua própria memória, ou da lembrança de seus pares, para achar documentos jurídicos similares ao que precisa ser elaborado por eles. E, mesmo quando a memória permite que seja identificado o caso similar, esta lembrança se restringe a poucos casos, não permitindo que o advogado tenha conhecimento de todos os casos similares existentes.

Alguns dos meios tecnológicos de busca existentes que podem auxiliar o advogado na elaboração de suas peças são: (I) Jurisprudência Unificada³; (II) Consulta Processual⁴ e a plataforma JUSBrasil.

No entanto, essas ferramentas carecem de informações e métodos de busca elaborados com relação a decisão tomada pelo juiz ou que consolidem os processos similares entre si e quais foram as peças jurídicas apresentadas nos casos ou a decisão proferida de interesse.

Ou seja, não há um banco de dados que forneça aos advogados uma visualização que relacione a sentença, a petição inicial e a contestação (peça de defesa do réu) com outros processos similares, permitindo que o profissional

³ Disponível em <https://www2.cjf.jus.br/jurisprudencia/unificada/>

⁴ Disponível em <http://www.tjrj.jus.br/>

possa ter um visão geral e informada sobre o posicionamento jurídico de dado juiz, e as peças que foram apresentadas nestes casos.

Desta forma, o sistema objeto deste trabalho irá auxiliar os advogados privados, sejam os que trabalham em escritórios de advocacia, sejam os que trabalham em departamentos jurídico, na melhor definição da sua estratégia jurídica, a partir da busca de processos similares nas situações acima descritas. Isto é, a partir da identificação de processos similares, é possível ter conhecimento sobre (1) como determinado juiz julga certos assuntos, (2) quais foram as peças jurídicas apresentadas nos casos em que o juiz julgou favorável à sua causa, e (3) as peças jurídicas apresentadas em casos similares sem depender da lembrança humana. Estas informações são relevantes para que o advogado possa tomar a melhor decisão no caso concreto.

Para o desenvolvimento do sistema, os principais recursos computacionais, levando em consideração dispositivos, ferramentas CASE, IDEs, linguagens de programação, frameworks e bibliotecas, utilizados foram:

- Notebook Dell Vostro, 8 GB RAM, 500GB HD. Sistema Operacional Linux Mint 19.1;
- Macbook Air, 8GB RAM, 500GB SSD. Sistema Operacional MacOS Mojave 10.14.5;
- Editor de texto Visual Studio Code;
- Editor de texto Jupyter Notebook;
- Astah Professional;
- PlantUML;
- Docker Engine 18.09.2;
- Banco de dados PostgreSQL 11.4;
- Ferramenta de administração do Postgres pgAdmin4;
- Banco de dados in-memory Redis 5.0.5;
- Ferramenta de administração do Redis Commander;
- Nginx Reverse Proxy Server 1.17.0;
- Linguagem de programação Python 3.7.3;
- Linguagem de programação Javascript;
- Pandas Data Analysis Library;
- Web Framework Django 2.2.2;
- Django REST Framework 3.9.4;
- UI Library ReactJS 16.8.6;
- Redux;

Neste projeto foram aplicados diversos conhecimentos adquiridos nas diversas disciplinas, obrigatórias e eletivas, de programação, tendo como maiores referências as disciplinas de Programação Orientada a Objetos, Modelagem de Dados, Bancos de Dados (I, II, III, IV), Introdução à Ciência de Dados, Programação Distribuída e Concorrente, Inteligência Competitiva, Modelagem de Software e Qualidade de Software, Gestão de Serviços de TI.

2 Situação Atual

Dado que o projeto é uma extensão e readequação dos protótipos criados para o estudo da hipótese de auxílio a tomada de decisão dos juízes e da identificação de fraudes ocorridas no âmbito dos juizados especiais cíveis, serão descritas as características destes protótipos e mudanças na arquitetura necessárias para um sistema que possa ser modelado para um sistema que esteja mais adequado para ser implantado em produção.

Uma das características do modelo, é que o produto gerado pelo modelo é um conjunto de planilhas no formato CSV que descreve o código do processo de referência, o código do processo similar e o índice de similaridade calculado.

Analisar esse conjunto de dados nas planilhas é uma tarefa extremamente ineficiente, tanto pelo grande número de arquivos gerados pelo modelo, quanto pela necessidade de análise dos documentos que contribuíram para o índice de similaridade. Portanto, foi criada a interface do protótipo, representadas nas imagens abaixo, que buscam facilitar a visualização desses dados.



Figura 1 – Interface de Busca por Processo Judicial

A figura 1 demonstra a interface de busca, realizada a partir dos códigos identificadores da numeração de processos no formato definido pelo Tribunal de Justiça do Rio de Janeiro (TJRJ).



Figura 2 – Interface de Detalhe de Processos Judiciais Similares

Já a figura 2 demonstra uma versão da tela de detalhamento das similaridades, onde são disponibilizados (i) os códigos identificadores dos números do processo no formato do TJRJ e no formato estipulado pelo Conselho Nacional de Justiça (CNJ), (ii) os nomes dos personagens envolvidos, autor e réu, (iii) a Comarca⁵ e Serventia⁶ nos quais o processo aberto, e (iv) os *links* externos para os arquivos da petição inicial, da contestação e para a página de movimentos judiciais no portal do TJERJ, além do índice de similaridade criado pelo modelo. Com essas informações, o advogado poderia compreender melhor os detalhes do processo similar.

Dada a limitação do banco de dados modelado na ocasião, por utilizar planilhas pré-processadas, e que não refletiam a real possibilidade de uso dos dados em conjunto com as planilhas geradas a partir do modelo de similaridade, além de gerar um grande volume de dados replicados que poderiam ser simplificados, como demonstrado na figura 3, foi necessária uma exploração e

⁵ (CNJ, 2016) A comarca corresponde ao território em que o juiz de primeiro grau irá exercer sua jurisdição e pode abranger um ou mais municípios, dependendo do número de habitantes e de eleitores, do movimento forense e da extensão territorial dos municípios do estado, entre outros aspectos. Cada comarca, portanto, pode contar com vários juízes ou apenas um, que terá, no caso, todas as competências destinadas ao órgão de primeiro grau.

⁶ Serventias podem ser considerados cartórios que prestam serviços judiciais, extrajudiciais ou ambos.

nova limpeza nos dados utilizados na realização do treinamento. Esses dados são uma amostra desestruturada do banco de dados do TJRJ disponibilizada em planilhas no formato CSV, cujo volume está em torno de 60GB, com diversos conteúdos de colunas faltantes e dados que necessitavam de tratamento e compreensão em torno dos relacionamentos, bem como para ter sinergia com o contexto deste projeto.

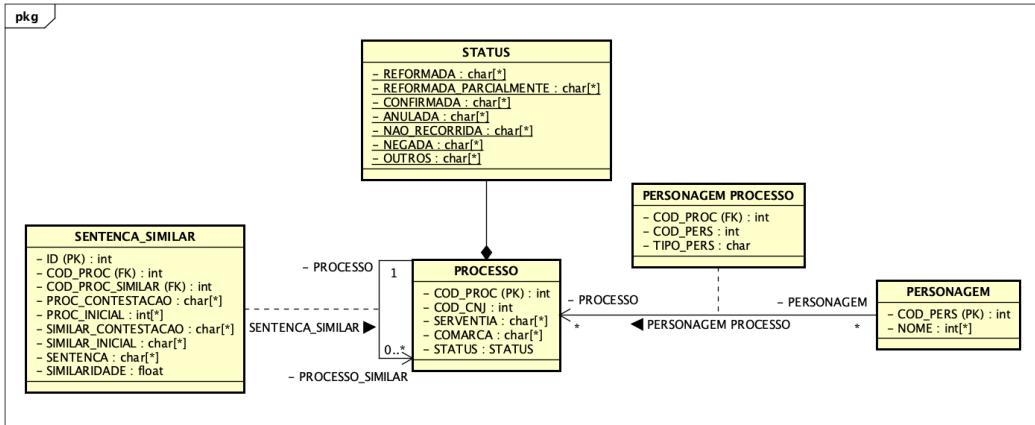


Figura 3 – Modelagem de Classes representando o mapeamento objeto relacional utilizado no protótipo.

Portanto, foi necessário explorar conceitualmente esses dados e criar um script para a limpeza e consolidação dos dados, o criando um banco de dados estruturado que daria suporte a interface gerada nesse projeto e em outros que fariam uso dessa mesma base de dados desestruturada.

A arquitetura do protótipo original era bastante simples, basicamente uma aplicação Django, utilizando o banco de dados SQLite e configurado como um serviço em um servidor com sistema operacional Ubuntu. Mesmo sendo simples, a implantação desse serviço era bastante manual, o que mostrou um ponto de melhoria em relação a este serviço, e outros que poderiam dar suporte.

Mesmo tendo sido desenvolvido um script, para realizar o cadastro dos registros era necessário ter acesso direto ao backend. Nesse aspecto, é interessante a criação de uma API que suporte o processo de persistência.

Visando minimizar o esforço de desenvolvimento e de implantação em máquinas distintas, buscou-se aplicar conceitos de DevOps⁷, implementando os

⁷ (MICROSOFT, 2019) DevOps é a união de pessoas, processos e tecnologias a fim de proporcionar a entrega contínua do valor para os clientes, sendo o termo composto por dev (*development* ou desenvolvimento) e ops (*operations* ou operações), unificando operações de desenvolvimento de software e TI para automatizar o ciclo de vida do aplicativo.

artefatos, quando possível, em containers Docker⁸ e scripts para a automação de tarefas de compilação e backup da solução desenvolvida, contribuindo para a produtividade e a criação de uma arquitetura orientada a microsserviços.

Uma solução em contêineres Docker possibilita implantar e conectar serviços dependentes entre si com maior facilidade, seja localmente, ou na nuvem, desacoplando esses artefatos e facilitando a manutenção dos mesmos.

3 Objetivos

A proposta desse trabalho é especificar uma documentação para a amostra de base de dados do TJRJ que possa suportar o desenvolvimento de outros trabalhos que se apropriem da mesma e aprimorar a arquitetura desenvolvida no projeto, adequando-o ao problema descrito na introdução, ou seja, criar um sistema de apoio à decisão de advogados, fornecendo a eles funcionalidades que auxiliem eles no processo de análise processos similares.

As funcionalidades em questão incluem a busca, análise, listagem, agrupamento e avaliação de processos similares, que é a proposta deste projeto para aplicar os dados gerados pelo modelo utilizado, ao contexto de uso dos advogados, de acordo com os problemas descritos na introdução.

Com relação ao protótipo original, existe o objetivo de aprimorar a arquitetura, projeto de banco de dados e visualização dos dados, que abrangia as funcionalidades de busca e análise de processos similares aplicados às hipóteses sob estudo no apoio à tomada de decisão de juízes e identificação de possíveis fraudes, na ocasião.

As funcionalidades de listagem, agrupamento e avaliação de processos similares propostas, buscam apoiar os usuários advogados de acordo com as motivações descritas na introdução.

A possibilidade de criar grupos de advogados auxiliaria na sua auto-organização, tornando desnecessário que ele crie mecanismos fora da plataforma para manter referências em relação a análise que está sendo realizada. Apesar de estar implementada para o uso individual, esta funcionalidade pode ser facilmente estendida para o uso em escritórios de advocacia, possibilitando compartilhar os processos similares agrupados.

⁸ Docker é uma tecnologia que fornece contêineres, ou seja, um artefato de software que empacota o código e suas dependências em um ambiente que pode ser facilmente reproduzido em diferentes infraestruturas.

No caso da avaliação de um processo similar, é um recurso que disponibilizaria um meio de comunicação fácil e rápido com os responsáveis pelo modelo de inferência de similaridades, para indicar possíveis inconsistências nos índices de similaridade, afim de melhorá-los.

4 Atividades Realizadas

Por ser um projeto que envolve a visualização de dados, o método aplicado durante o desenvolvimento buscou-se apoiar nas práticas de projetos de ciências de dados, como observado na figura 4.

Dentro desse processo, houve grande empenho na etapa de exploração e tratamento dos dados que serviriam como base para a visualização de processos similares a partir da interface web produzida, associando os dados tratados com os registros de processos similares gerados pelo modelo de inferência de processos similares. O segundo foco foi a implementação da interface em si.

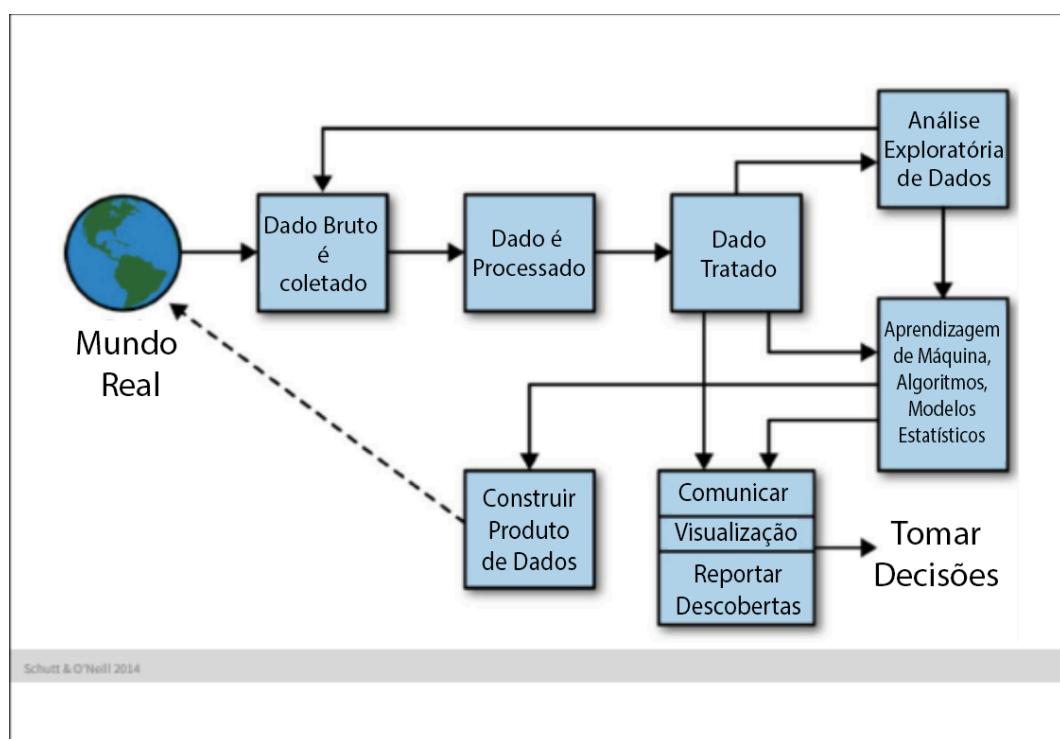


Figura 4 – Processo de Análise de Dados sugerido na Disciplina de Introdução a Ciência de Dados (adaptado)

A primeira tarefa foi criar uma base na qual será desenvolvido o projeto. Foram utilizados contêineres Docker para auxiliar e validar alguns conceitos associados a tecnologia, como a facilidade de realizar o desenvolvimento em diferentes dispositivos e infraestruturas, dado que foi utilizado dois computadores

distintos, além de ter sido feita uma implantação nos serviços da AWS, o serviço de computação em nuvem da Amazon.

Na arquitetura especificada, são necessários 4 contêineres Docker essenciais (Nginx, ElisApp, ElisDB e Redis) e outros utilizados para o suporte ao desenvolvimento e administração da solução:

- ElisApp: Contêiner lançado a partir da imagem Docker que é produto do empacotamento da solução desenvolvida. Na prática, é uma aplicação (Backend) implementada em Django que expõe uma API⁹ RESTful¹⁰ e uma *Single Page Application*¹¹ (Frontend) implementada em ReactJS, além de possibilitar executar um serviço do Jupyter Notebook que pode ser utilizado para interagir diretamente com o backend e auxiliar na criação e execução dos scripts Python utilizados no tratamento e cadastro dos registros persistidos no banco de dados através da API. A imagem está disponibilizada em: <https://hub.docker.com/r/yangricardo/elisapp>;
- ElisDB: Contêiner gerado a partir de uma imagem Docker do sistema gerenciador de banco de dados PostgreSQL. A imagem disponibilizada em <https://hub.docker.com/r/yangricardo/elisadb>, executa alguns scripts SQL gerados a partir de um script de backup do banco de dado, ou seja, seus esquemas, dados, visões, visões materializadas e funções implementadas nessa camada;
- Redis: Contêiner executado a partir da imagem padrão do Redis, banco de dados em memória, utilizado para realizar o cache, ou seja, criar um acesso rápido as informações que já tenham sido requisitadas;
- Nginx: Contêiner executado a partir da imagem padrão do Nginx e configurado para disponibilizar o servidor de proxy reverso¹² e disponibilizar os arquivos estáticos disponibilizados pelo Django;

⁹ Application Programming Interface, ou seja, uma interface de programação de aplicação que dá suporte ao desenvolvimento e integração de soluções de TI.

¹⁰ REST é um conjunto de princípios de arquitetura de software cliente/servidor que se utilizam do protocolo de comunicação HTTP para gerenciar recursos (dados) a partir de formatos, normalmente XML ou JSON, para prover serviços web de maneira padronizada. RESTful é a capacidade de um sistema implementar esses princípios.

¹¹ Single Page Application é uma aplicação web que reescreve o conteúdo dinamicamente, de acordo com as interações do usuário, minimizando a necessidade de recarregar totalmente uma página web acessada e atualizando assincronamente o seu conteúdo.

¹² (IBM) O servidor proxy reverso protege servidores HTTP, fornecendo um único ponto de acesso à rede interna.

- Redis Commander e PgAdmin: Contêineres Docker para a administração dos bancos de dados Redis e PostgreSQL, respectivamente;

Durante o desenvolvimento, foram realizados experimentos associados as tecnologias. Com relação ao Django, tecnologia de maior domínio até então, foram feitos apenas experimentos associados a integração com uma aplicação ReactJS e o seu uso em conjunto com Contêineres Docker. O código está disponibilizado no repositório GitHub <https://github.com/yangricardo/elisapp>.

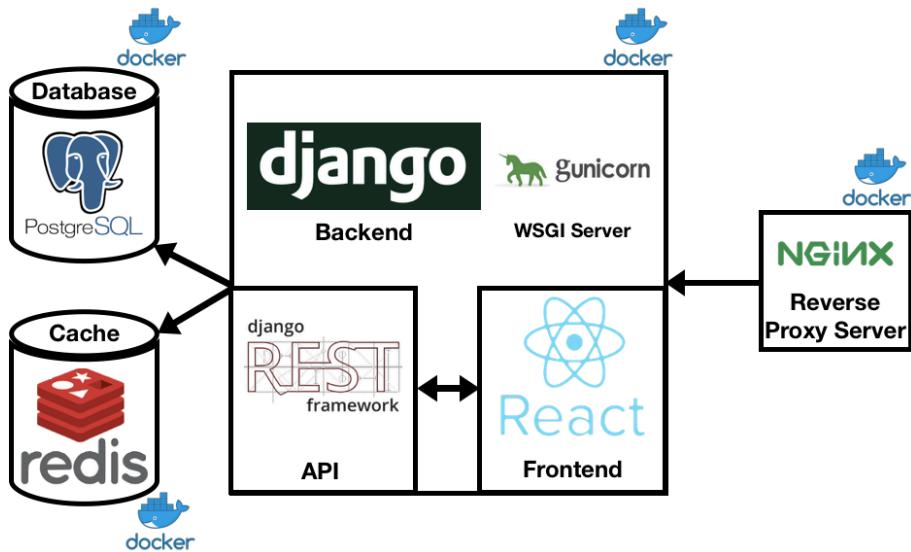


Figura 5 – Arquitetura Tecnológica

5 Projeto e Especificação do Sistema

5.1 Exploração da Base de Dados do TJRJ

O processo de exploração de dados ocorreu durante o período do projeto final 1, onde realizou-se uma engenharia reversa na amostra de dados do TJRJ a fim de criar um modelo de entidade relacionamento adequado ao contexto deste projeto, reunindo informações necessárias para o apoio à tomada de decisão dos advogados e adequar esses dados às formas normais¹³ possíveis de se aplicar em um projeto de banco de dados.

Ao explorar a base de dados, notou-se também a presença uma série de dados redundantes, faltantes e planilhas CSV com um volume considerável.

¹³ Conjunto de regras que visa organizar o projeto de um banco de dados, visando reduzir a redundância de dados e aumentar a integridade de dados e desempenho.

Essas planilhas estavam codificadas na especificação ISO¹⁴ /IEC¹⁵ 8859-1, conhecidos informalmente como CP1252 ou Latin1, e são utilizados para a codificação de caracteres do alfabeto latino.

O resultado deste estudo produziu uma planilha que seleciona alguns atributos candidatos a chaves primária, estrangeira e desconsiderou atributos com alta frequência de dados faltantes ou que não agregassem valor informacional ao contexto desse projeto.

Como artefato produto do processo de exploração, foi desenhado o Modelo de Entidade e Relacionamento (MER), disponibilizado no anexo I utilizando a notação de Crow's Foot (IE) existente na ferramenta CASE para modelagem Astah. Esse passo permitiu aprimorar a normalização das tabelas que serão mapeadas fisicamente no banco de dados e definir com maior precisão os tipos dos atributos. Também foram adicionados elementos de notas indicando proveniência de qual planilha CSV uma entidade foi gerada.

5.2 Mapeamento Objeto-Relacional e API RESTful

O mapeamento objeto-relacional do Django (Models) é utilizado para interagir com as tabelas relacionais persistidas no banco de dados PostgreSQL geradas a partir do MER (anexo I). Dado que as operações do sistema são, em grande parte, a execução de consultas complexas que envolvem diversas tabelas, para obter o detalhamento ou a listagem de processos similares, foram implementadas visões e visões materializadas no banco de dados, para auxiliar na obtenção dos resultados das consultas, de maneira otimizada, as tabelas relacionais e da tabela que contém os registros gerados pelo modelo de inferência de similaridades.

Com o mapeamento objeto-relacional, descrito no Diagrama de Classes do anexo III, utilizou-se as classes disponibilizadas pelo Django REST Framework, que provê uma biblioteca para rápida implementação de APIs RESTful e a disponibilização dos recursos especificados, com destaque para os Serializers, ViewSets, paginação, permissionamento e autenticação.

Os serializers possibilitam converter os bytes dos objetos em um formato que seja possível transportar via protocolo HTTP, como o JSON. Os ViewSets

¹⁴ ISO é a sigla para *International Organization Standardization*, ou Organização Internacional de Normalização, uma entidade não governamental que reúne 162 entidades de normalização ao redor do mundo.

¹⁵ IEC é a sigla para *International Electrotechnical Commission*, é a organização líder mundial que prepara e publica normas internacionais para todas as tecnologias elétricas, eletrônicas e relacionadas.

disponibilizam uma interface simples para disponibilizar os recursos nos pontos de acesso (endpoints) do servidor da API. Quando possível, utilizou-se de herança¹⁶, inclusive múltipla, para otimizar código e torna-lo mais legível e fácil de realizar a sua manutenção.

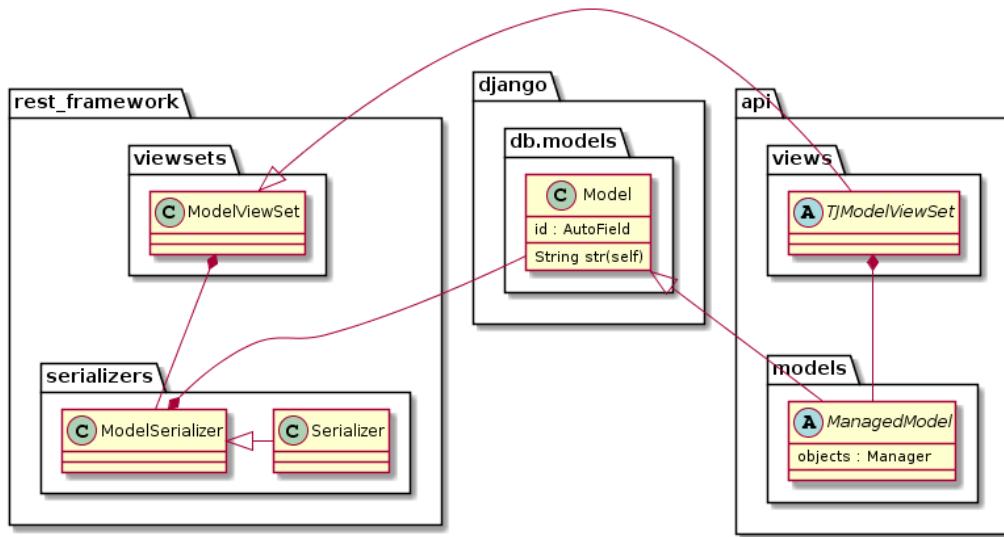


Figura 6 – Diagrama de Classes de Referência para a disponibilização dos Recursos da API RESTful

5.3 Modelo de Autenticação com a API RESTful

Para interagir com os recursos, ponto de acesso da API RESTful, seja via script ou interface gráfica, é necessário se autenticar com uma conta através do endpoint de Login, em caso de sucesso, será obtida uma chave de autenticação (token) a ser utilizada em todas as requisições HTTP com a API, com exceção dos recursos de registro e Login.

Os recursos associados as tabelas relacionais geradas a partir da exploração, tratamento e persistência dos dados do TJRJ, tem o seu acesso limitado às contas com nível de permissão de administrador da plataforma, definidos de acordo com o padrão do Django. Os grupos de processos similares possuem o seu acesso limitado à conta criadora.

¹⁶ Princípio do paradigma de programação orientada a objetos que permite o compartilhamento de atributos e métodos entre classes, possibilitando generalizar o comportamento do código e especializar o mesmo quando necessário. A linguagem Python possibilita a herança múltipla, recurso mais utilizado na implementação de ViewSets genéricas para selecionar os métodos HTTP possíveis de serem realizados por um recurso disponibilizado.

5.4 Tratamento e Persistência de Dados.

O processo de tratamento e persistência de dados pode ser visualizado no diagrama de sequência presente no anexo II, que demonstra o funcionamento genérico do script Python DataCleaning.ipynb, implementado através utilizando um Jupyter Notebook configurado para estar integrado ao backend Django em execução, utilizando aplicando o paradigma funcional e a biblioteca Pandas, como práticas eficientes no tratamento de dados com em Python.

Esse script foi desenvolvido buscando minimizar problemas com a integridade dos dados, consolidando no banco de dados deste projeto somente a os dados relativos aos processos judiciais que passaram pelo modelo de inferência de similaridades.

Algumas dessas planilhas são simples de tratar, pois tratam somente a definição de parâmetros referenciados por outras tabelas e possuem baixo volume de dados, apresentando baixa complexidade no seu processamento. Outras aumentam sua complexidade devido a fatores como um grande volume de dados ou a necessidade de realizar junções entre as tabelas ou filtros complexos para reduzir a quantidade de dados em memória sendo tratados e persistidos.

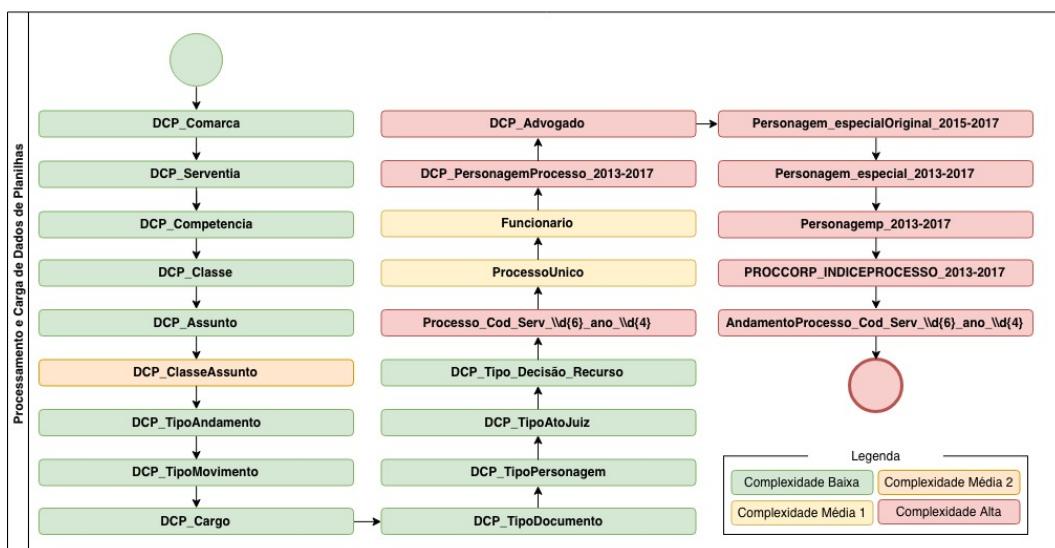


Figura 7 – Complexidade no Tratamento das Planilhas CSV.

Junto ao mencionado script acima, foi implementado o módulo utils.py, presente no pacote scripts da aplicação backend, que disponibiliza duas classes para auxiliar o desenvolvimento de tarefas repetitivas do script:

- TJData: Classe que contém diversos métodos estáticos que auxiliam na limpeza e preparação dos Pandas Dataframes que geram os objetos convertidos em JSON para a realização de requisições HTTP POST a API REST disponibilizada pelo backend;

- ElisAPI: Classe responsável por obter as credenciais de autenticação utilizados junto ao backend, ou seja, o token utilizado para realizar as operações HTTP nos recursos (endpoints) disponibilizados pela API.

Tanto a classe TJData quanto a ElisAPI, possibilitam realizar ações em paralelo com o uso de ThreadPools, um padrão de design de software que permite realizar tarefas concorrentemente, como forma de obter maior performance e agilidade no processamento de tarefas de limpeza, por meio da classe TJData.

Durante a persistência dos dados, como o ThreadPool implementado na classe ElisAPI executa requisições HTTP POST concorrentemente à API RESTful, utilizando a biblioteca Requests (Python), os objetos de retorno dessas requisições praticamente duplicam os dados usados no corpo da requisição, ao manter os dados de origem e os de retorno, que podem incluir os dados enviados completos no caso de requisições bem sucedidas, e as indicações de dados invalidados, seguindo o padrão dos serializers implementados via Django REST Framework.

Uma forma de evitar um vazamento da memória que essa duplicação dos dados em memória poderia ocasionar, foi apagar esses objetos da memória caso a requisição fosse considerada bem-sucedida, através do código HTTP 201. Ao manter as requisições malsucedidas por motivos de dados inválidos, foi possível cruzar os registros dos dados de origem e os de retorno afim de identificar os registros que não apresentaram problemas de integridade e, então, refazer as requisições HTTP POST para persistir os dados no banco de dados.

Em geral, esse script tratou somente os processos das comarcas de Bangu e Campo Grande entre os anos de 2013 e 2017, utilizando os respectivos códigos identificadores para selecionar os dados dos processos de interesse.

Uma lógica similar foi utilizada nas tabelas que faziam referência aos processos em si, como tabelas de personagens, advogados, andamentos e documentos, que utilizavam os códigos dos processos para selecionar esses dados específicos, reduzindo significativamente a quantidade de dados para a persistência do banco de dados deste sistema.

5.5 Interface Gráfica

O projeto das interfaces abaixo foi inspirado no processo de Design Baseado em Cenários, e como é citado por (BARBOSA e SILVA, 2010), é um processo que utiliza diferentes tipos de cenários como representação básica e fundamental durante as atividades envolvidas na concepção de IHC.

Um cenário representa uma história sobre pessoas executando uma atividade, e geralmente são escritos em linguagem natural, motivando todos os interessados a participarem e contribuir com decisões de design.

Dada a necessidade de tratar os cenários dos problemas dos advogados, como descrito na introdução, foram especificados os casos de uso presentes na seção 5.6 e na figura 6.

O anexo V apresenta um fluxograma de telas para descrever a sequência de execução dos casos de uso em relação às interfaces, além das versões de interfaces implementadas durante o projeto.

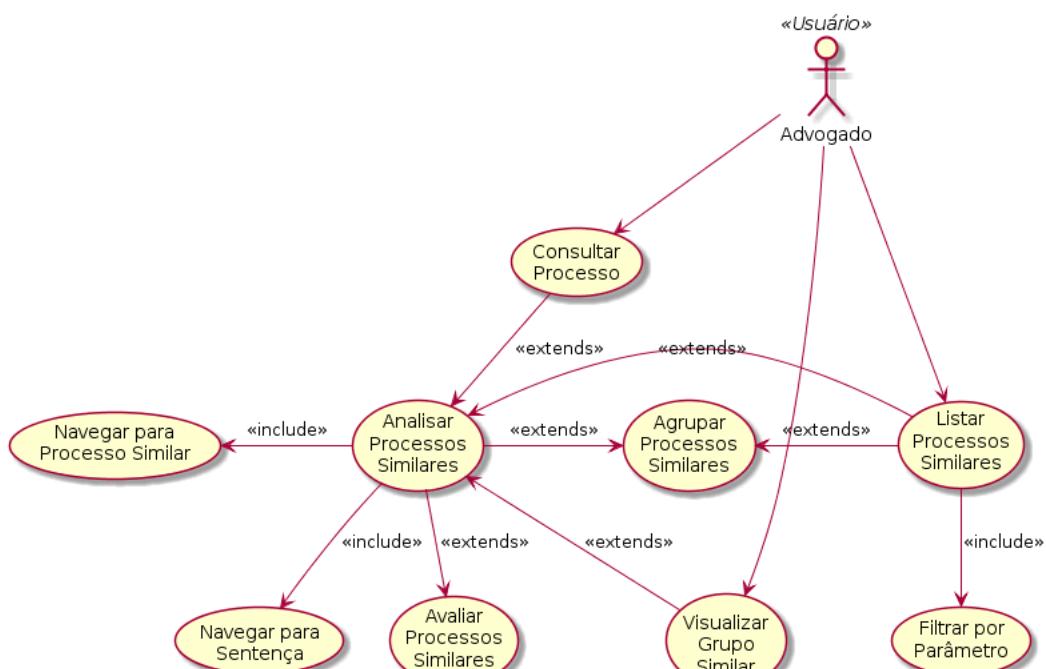


Figura 8: Diagrama de Casos de Uso

5.6 Descrições de Casos de Uso

5.6.1 Consultar Processo

- Nível: Primário
- Atores: Usuário
- Pré-condições:
 - O Usuário está autenticado no sistema;
 - O Usuário deseja consultar por um processo específico;
- Fluxo Principal:
 1. A página exibe o formulário de Consulta de Processo
 2. O Usuário indica ou mantém no seletor o formato de código de processo (TJRJ ou CNJ) a ser consultado;

- 3. O Usuário insere o código do processo de interesse;
- 4. O Usuário executa a consulta por processo;
- Fluxo Alternativo:
 1. O Usuário executa a consulta por processo;
 1. O sistema notifica o Usuário que não existem sentenças similares para o código do processo buscado;
 2. O Usuário executa o caso de uso **Consultar Processo**
- Pós-condições:
 - Fluxo Principal: O usuário executa o caso de uso **Analizar Processos Similares**;
 - Fluxo Alternativo: O usuário retorna caso de uso **Consultar Processo**;

5.6.2 Analisar Processos Similares

- Nível: Primário
- Atores: Usuário
- Pré-condições:
 - Usuário está autenticado no sistema;
 - O Usuário realizou um dos seguintes casos de uso:
 - **Consultar Processo**;
 - **Listar Processos Similares**;
 - **Visualizar Grupo Similar**;
- Fluxo Principal:
 1. O sistema exibe a *Interface de Análise de Processo Similares* com as informações do par de processos similares, o índice de similaridade e a lista de processos similares;
 2. O Usuário visualiza as informações de Comarca, Serventia, Classe, Assunto, Advogados, Personagens Autores e Réus, Juiz, Ato do Juiz, Texto de Sentença do Juiz do par de processos similares atual;
 3. O Usuário aciona o link para o documento de Petição Inicial, de Contestsção ou Movimentação de cada par de processos similares atual e é redirecionado para fora do sistema;
- Fluxo Alternativo:
- Pós-condições:
 - Fluxo Principal e Alternativo: As pós-condições desse caso de uso são refletidos de acordo com a ocorrência dos seguintes casos de uso:
 - **Avaliar Processos Similares**

- **Navegar para Sentença do Processo**
- **Navegar para Processo Similar**
- **Agrupar Processos Similares**

5.6.3 Avaliar Processos Similares

- Nível: Secundário
- Atores: Usuário
- Pré-condições:
 - O Usuário está realizando o caso de uso **Analisar Processos Similares;**
 - O Usuário deseja realizar uma avaliação de similaridade;
- Fluxo Principal:
 1. O sistema exibe a *Interface de Avaliação de Processos Similares*
 2. O Usuário atribui um valor de 1 a 5 a todas às seguintes perguntas:
 - *A similaridade está adequada considerando o texto da sentença?*
 - *A similaridade está adequada considerando o texto da inicial?*
 - *A similaridade está adequada considerando o texto da contestação?*
 4. O Usuário insere um comentário textual *sobre a taxa de similaridade*;
 5. O Usuário envia a avaliação;
- Fluxo Alternativo:

 1. O Usuário cancela a avaliação;

- Pós-condições:
 1. Fluxo Principal: O sistema registra a avaliação de similaridade;
 2. Fluxo Principal e Fluxo Alternativo: O Usuário retorna ao caso de uso **Analisar Processos Similares;**

5.6.4 Agrupar Processos Similares

- Nível: Secundário
- Atores: Usuário
- Pré-condições:
 - O Usuário está realizando um dos seguintes casos de uso:
 - **Analisar Processos Similares;**
 - **Listar Processos Similares**

- O Usuário deseja adicionar o par de sentenças similares em um agrupamento similar;
- Fluxo Principal:
 1. O Usuário seleciona o par de processos similares;
 2. O Usuário aciona a *Interface de Agrupamento de Processos Similares*;
 3. O sistema carrega os agrupamentos existentes associados ao usuário;
 4. O Usuário seleciona ou insere o nome dos agrupamentos de processos similares;
 5. O Usuário envia a requisição;
- Fluxo Alternativo:
 1. O Usuário cancela o agrupamento;
- Pós-condições:
 1. Fluxo Principal:
 - O sistema adiciona o par de processos similares aos grupos indicados;
 - O Usuário retorna ao caso de uso que estava realizando;
 2. Fluxo Alternativo:
 - O Usuário retorna ao caso de uso anterior;

5.6.5 Navegar para Processo Similar

- Nível: Secundário
- Atores: Usuário
- Pré-condições:
 - O Usuário está realizando o caso de uso **Analisa Processos Similares**;
 - O Usuário deseja analisar as informações de outro processo similar ao processo buscado
- Fluxo Principal:
 1. O Usuário seleciona um processo similar indicado na lista de processos similares ao processo buscado
 2. O sistema atualiza o conteúdo da página;
- Fluxo Alternativo:
- Pós-condições:
 - Fluxo Principal: O Usuário executa o caso de uso **Analisa Processos Similares**;

5.6.6 Navegar para Sentença

- Nível: Secundário
- Atores: Usuário
- Pré-condições:
 - O Usuário está realizando o caso de uso **Analizar Processo Similares**;
 - O Usuário deseja analisar outra sentença de um dos processos do par de processos similares;
- Fluxo Principal:
 1. O sistema exibe as páginas de sentenças do processo
 2. O Usuário seleciona uma das páginas de sentenças do processo disponíveis;
 3. O sistema atualiza o texto da sentença;
- Fluxo Alternativo:
- Pós-condições:
 - Fluxo Principal:
 - O sistema atualiza o conteúdo referente ao processo similar e sentença do processo buscado;
 - O Usuário continua o caso de uso **Analizar Processos Similares** para o par de processos atual;

5.6.7 Listar Processos Similares

- Nível: Primário
- Atores: Usuário
- Pré-condições:
 2. O Usuário está autenticado no sistema;
 3. O Usuário deseja realizar uma listagem baseada em parâmetros de busca sobre os processos similares;
- Fluxo Principal:
 1. O sistema apresenta a *Interface de Listagem de Processos Similares*;
 2. O usuário realiza arbitrariamente o caso de uso **Filtrar por Parâmetro** para os seguintes parâmetros:
 - Comarca;
 - Serventia;

- Ano;
 - Classe;
 - Assunto;
 - Nome do Personagem;
 - Advogado;
 - Juiz;
3. O usuário define o intervalo de índice de similaridades a ser realizada a busca;
 4. O usuário aciona a listagem de processos similares baseada nos parâmetros especificados;
 5. O sistema retorna a lista de processos similares que correspondem aos parâmetros especificados;
- Fluxo Alternativo:
 1. O sistema notifica que não houveram processos similares, caso não haja correspondências de acordo com os parâmetros;
 - Pós-condições:
 1. Fluxo Principal:
 - O sistema apresenta a dos processos com sentenças similares ordenados pela taxa de similaridade;
 - O usuário pode executar arbitrariamente o caso de uso **Analizar Processos Similares** com um par de processos presentes na lista;
 - O usuário pode executar o caso de uso **Agrupar Processos Similares**;
 2. Fluxo Principal e Fluxo Alternativo: O usuário retorna ao caso de uso **Listar Processos Similares**;

5.6.8 Filtrar por Parâmetro

- Nível: Secundário
- Atores: Usuário
- Pré-condições:
 - O Usuário está realizando o caso de uso **Listar Sentenças Similares** a partir da:
 - *Interface de Listagem de Sentenças Similares*;

- *Interface de Visualização de Grupos de Processos Similares;*
- O Usuário deseja filtrar a lista de processos similares com base em um dos parâmetros:
 - Comarca;
 - Serventia;
 - Ano;
 - Classe;
 - Assunto;
 - Nome do Personagem;
 - Advogado;
 - Juiz;
- Fluxo Principal:
 1. O sistema retorna a lista de opções disponíveis de acordo com o parâmetro;
 2. O usuário insere o texto referente a uma descrição possível de acordo com o parâmetro;
 3. O sistema filtra a lista de opções disponíveis de acordo com a entrada do usuário no passo anterior;
 4. O usuário seleciona uma das opções disponíveis restantes;
- Fluxo Alternativo:
 1. O sistema não localiza uma opção disponível de acordo com a entrada do usuário
 2. O usuário seleciona a opção sem filtro;
 3. O usuário reinicia o fluxo principal;
- Pós-condições:
 - Fluxo Principal e Alternativo: O Usuário retorna ao caso de uso anterior;

5.6.9 Visualizar Grupo Similar

- Nível: Secundário
- Atores: Usuário
- Pré-condições:
 - O Usuário está autenticado no sistema;

- O Usuário deseja visualizar processos similares adicionados a um grupo criado anteriormente por meio do caso de uso **Agrupar Processos Similares**
- Fluxo Principal:
 1. O sistema apresenta os grupos associados ao usuário via *Interface de Listagem de Grupos de Processos Similares*
 2. O usuário seleciona um grupo listado;
 3. O sistema apresenta os processos associados ao grupo;
- Fluxo Alternativo:
 1. O sistema não apresenta grupos criados, caso não existam;
- Pós-condições:
 - Fluxo Principal:
 - O usuário pode executar o caso de uso **Analizar Processos Similares** em relação a um par de processos listado;
 - O usuário pode remover um par de processos similares de se este tiver sido associado ao grupo;
 - Fluxo Principal e Alternativo: O usuário retorna ao caso de uso atual;

5.7 Otimização das Consultas

Devido a possibilidade de um grande volume de dados serem retornados, alguns ViewSets são programados para retornar os resultados de uma consulta de maneira paginada, limitando por acesso de acordo a página da consulta.

Outra otimização necessária, e descrita no diagrama de sequências, presente no anexo, foi a necessidade de criar um mecanismo para acelerar o acesso aos retornos das consultas, conhecido como cache, dado que mesmo com o uso das visões, em alguns casos é necessário consultar mais de uma visão materializada para gerar o retorno da consulta serializado via API.

Por esse motivo, foi utilizado o Redis, um banco de dados em memória para gerar o cache das consultas realizadas na API, otimizando o acesso às consultas realizadas anteriormente por processos similares de acordo com os parâmetros de busca utilizados.

Também foi desenvolvido um mecanismo de cache na camada do cliente,

utilizando o Redux, inspirado na arquitetura Flux¹⁷, e permite que os componentes de interface gráfica implementados em ReactJS possam gerenciar os estados de uma aplicação.

A base dessa arquitetura é o *Store*, um repositório central e imutável dos estados que serão observados pelos componentes de interface gráfica. Este é composto por diversos *Reducers*, repositórios de dados especialistas, que recebem e tratam informações recebidas via *Actions*, ou seja, eventos invocados pelos componentes de interface gráfica que irão evoluir os estados de acordo com a lógica implementada.

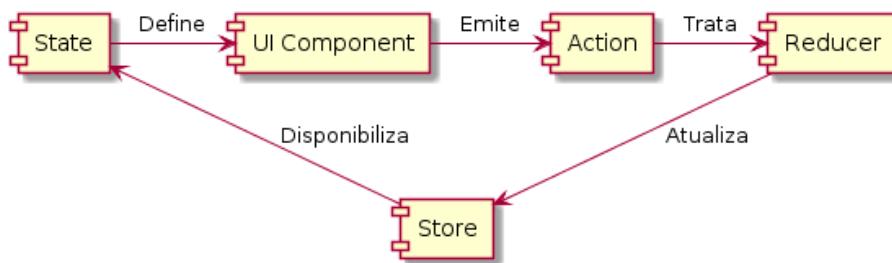


Figura 9: Estrutura e Fluxo de Dados do Redux

O uso dessa arquitetura possibilitou reduzir o número de chamadas à API, caso a consulta tivesse sido realizada em uma sessão, dado que são implementadas como *Actions* e as respostas são armazenadas no *Store*.

6 Implementação e avaliação

6.1 Planejamento e execução de testes funcionais

Durante a etapa de exploração e tratamento de dados, a própria execução do script de tratamento e persistência dos dados serviu para testar os pontos de acesso aos recursos da API implementada a partir das classes disponibilizadas pelo Django REST Framework (DRF).

Dado que não existem extensões muito complexas às classes disponibilizadas pelo DRF, ou seja, que alterem o comportamento padrão, foram realizados apenas testes para confirmar se os dados estavam sendo persistidos.

¹⁷ O Flux é uma arquitetura difundida pelo Facebook, muito similar ao padrão de projetos Observer, que implementa um fluxo de dados unidirecional entre os dados que são visualizados na interface gráfica e os eventos que irão evoluir os mesmos.

Alguns dos comportamentos padrões verificados e validados com o uso dos serializadores do DRF que foi possível verificar, é o retorno dos registros cadastrados caso a requisição HTTP Post fosse realizada com sucesso.

Caso contrário, o serializador retorna uma lista de dicionários, onde os dicionários vazios são os registros que não possuem falhas de integridade, sejam pela pré-existência da chave primária do registro ou alguma outra restrição definida no esquema da tabela ou do mapeamento objeto-relacional.

Para validar a pré-existência registro, um teste do tipo caixa preta a ser realizado é executar um bloco do notebook, ou seja, do script, e esperar que em uma segunda execução, esses registros retornem um HTTP Response com código de status diferente de 201 e indicando o motivo pelo qual os registros não foram cadastrados, como descrito anteriormente.

Dado que a primeira execução, em um banco de dados que não possua os dados cadastrados, tenha sido bem sucedida, as tentativas seguintes retornarão como um dos erros possíveis a falha devido ao uso de uma chave primária já existente, ou a referência a uma chave estrangeira inexistente, caso o registro de origem da chave primária não tenha sido realizado.

Como descrito na seção 5.3, as interações com os recursos da API ocorrem via autenticação Token, caso uma requisição não possua Token válido, é retornado um HTTP Response 403 (Bad Request), indicando que a requisição é inválida e indicando que as credenciais estão inválidas.

Para a realização dos testes descritos na camada de API, foi utilizada a classe ElisAPI, implementada no módulo utils.py do pacote scripts, e executado a partir de um notebook Jupyter integrado ao backend.

6.2 Planejamento e execução de outros testes

Foi planejado um questionário online¹⁸ a ser realizado com advogados e estudantes de direito, para: (I) conhecer o seu perfil como usuários de sistemas em geral; (II) apresentar e validar as interações e visualizações realizadas com o relação ao sistema desenvolvido; (III) conhecer outras possibilidades de como o sistema pode auxiliar o advogado em suas tarefas; (IV) descobrir se existe interesse do advogado em assinar uma plataforma desse tipo.

No entanto, até a entrega e apresentação deste projeto, não foram obtidas respostas suficientes para ter uma conclusão confiável em relação a solução

¹⁸ Formulário disponibilizado em <https://forms.gle/KHWaydJFNHGjvQtB7>.

devido a disponibilidade das pessoas para realizar as sessões de testes presenciais ou apenas o preenchimento dos testes online.

Mesmo com poucas respostas obtidas, foi possível identificar alguns pontos de melhoria com relação aos elementos de interface, como tamanho ou escolha dos ícones que refletissem uma funcionalidade, no caso, criar grupos de processos similares, onde os usuários não visualizaram de imediato a possibilidade de realizar essa ação durante a listagem de processos similares.

Também houve uma preferência por listar os processos similares listados ou selecionados a partir da interface de *Listagem de Processos* ou *Grupos Similares*. Da maneira como foi implementada até o momento do teste, a listagem apresentada é baseada na lista dos demais processos similares associados ao processo tido como referência.

Por outro lado, esses testes apresentaram o interesse dos advogados e alunos de direito pela presença de uma ferramenta como a apresentada em seu dia a dia, justamente para auxiliá-los na elaboração das petições tendo como base os resultados históricos de tomadas de decisão de jurisprudência, possibilitando a uniformização destas em um universo bastante massificado como o dos Juizados Especiais Cíveis.

6.3 Comentários sobre a implementação

Com relação às limitações do sistema, por mais que o PostgreSQL lide relativamente bem com campos do tipo texto, é provável que uma abordagem não relacional (NoSQL) para a gestão de banco de dados, como MongoDB ou ElasticSearch, ofereçam o retorno de consultas em um grande volume de informações mais eficiente, ainda mais considerando que este trabalho lidou com uma amostra bastante reduzida da amostra de dados do TJRJ, baseada nos registros disponibilizados pelo modelo de inferência de similaridade entre processos judiciais. Esse é um ponto de melhoria na camada da API.

A interface foi desenvolvida visando o seu uso em plataformas Desktop ou Notebooks, não visando o seu uso em plataforma mobile devido a quantidade de informações exibidas. No entanto, ainda existem melhorias a serem desenvolvidas com relação a tornar os componentes de interface mais responsivos em relação ao tamanho da tela. Essas serão executadas conforme maior domínio da biblioteca Material UI, que disponibiliza componentes em conformidade com o Material Design para aplicações desenvolvidas em ReactJS.

Além disso, como foi necessário investir muito tempo na exploração e tratamento dos dados, em relação a própria implementação da interface gráfica,

outro ponto de melhoria ocorrerá de acordo com a realização de testes com usuários para oferecer aprimorar os elementos disponibilizados para a visualização e as interações envolvidas.

7 Considerações finais

O processo de exploração de dados apresentou situações reais como por exemplo a falta de documentação em torno de base de dados e como ocorre a sua extração de informações e que tornou necessário realizar um extenso trabalho de engenharia reversa para a compreensão e organização de um modelo de dados não estruturado, como o do TJRJ.

A modelagem de entidade e relacionamento dessa etapa talvez seja a contribuição de maior impacto à curto prazo, dado que existem outras pesquisas ocorrendo e que se apropriam desse mesmo banco de dados disponível ou similares que venham a ser estudados em processos de análise de dados e construção de modelos de aprendizagem de máquina.

Excluindo o tratamento de dados, a implementação das funcionalidades era relativamente simples, por esse motivo buscou-se aprender sobre tecnologias bastante utilizadas no mercado, como o Docker, que facilita o empacotamento de soluções de software para serem executadas localmente ou em provedores de serviços em nuvem.

Essa abordagem possibilitou experimentar e aplicar uma abordagem de arquitetura de software baseada em microsserviços, APIs, que possibilita desacoplar as dependências dos projetos e tornar a sua manutenção mais efetiva. Este projeto poderia inclusive desacoplar facilmente o código da solução frontend implementada em ReactJS em um quinto serviço executado na mesma máquina ou externamente, ao contrário do que ocorre atualmente, com a aplicação sendo exposta através do servidor Django.

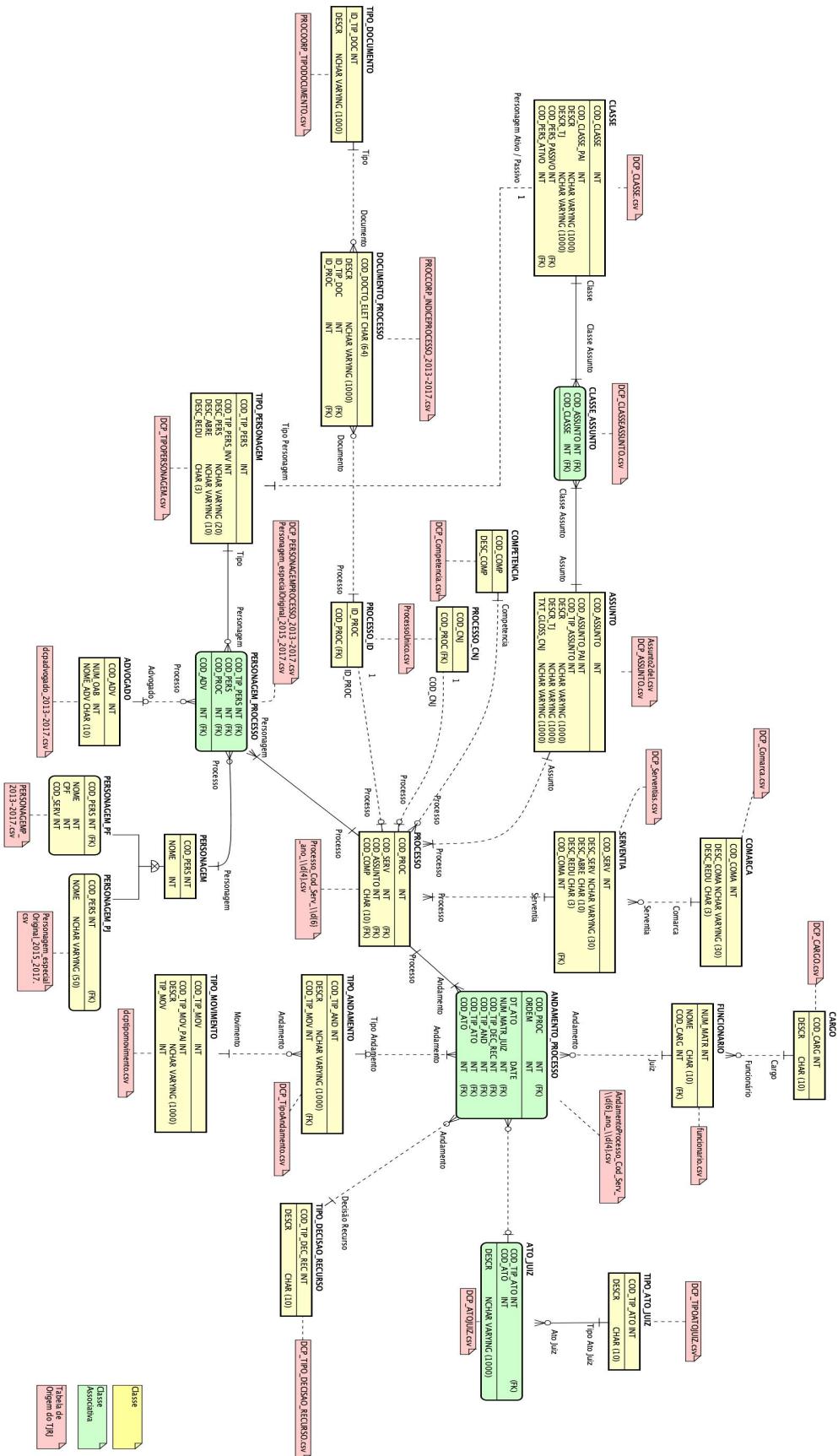
Sobre o aprendizado, foi possível aplicar e consolidar alguns conhecimentos com relação ao tratamento de dados, como a aplicação do paradigma de programação funcional, o projeto de banco de dados, na implementação de uma solução que seja orientada a APIs e no uso de tecnologias e práticas que facilitem a implantação em diversos tipos de plataformas, sejam locais ou em provedores de computação em nuvem, como o Docker, e de bibliotecas de desenvolvimento de aplicações web que vem sendo muito utilizadas no mercado, como o ReactJS e o Redux.

Por fim, aprendeu-se um pouco sobre um domínio muito específico, o do Direito, buscando apoiar os advogados em parte de seu trabalho, durante a elaboração de suas respectivas defesas judiciais nas competências dos Juizados Especiais Cíveis e que pode ser facilmente estendido para outros contextos, dependendo somente dos resultados de modelos de inferências de similaridades de processos judiciais e da disponibilidade dos dados que alimentarão o banco de dados desta interface.

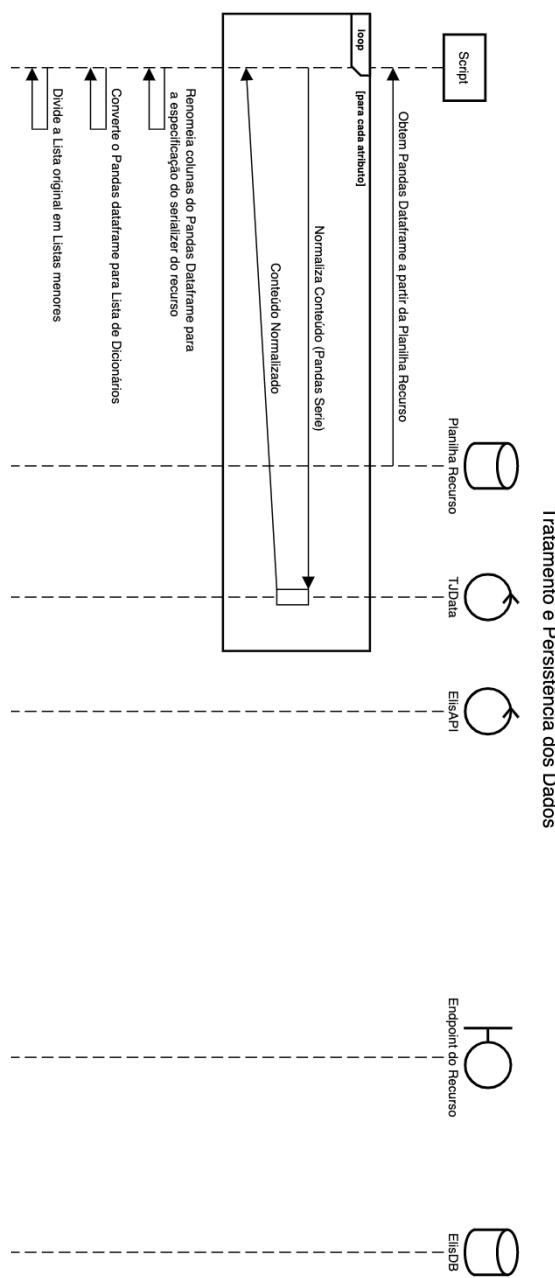
Bibliografia

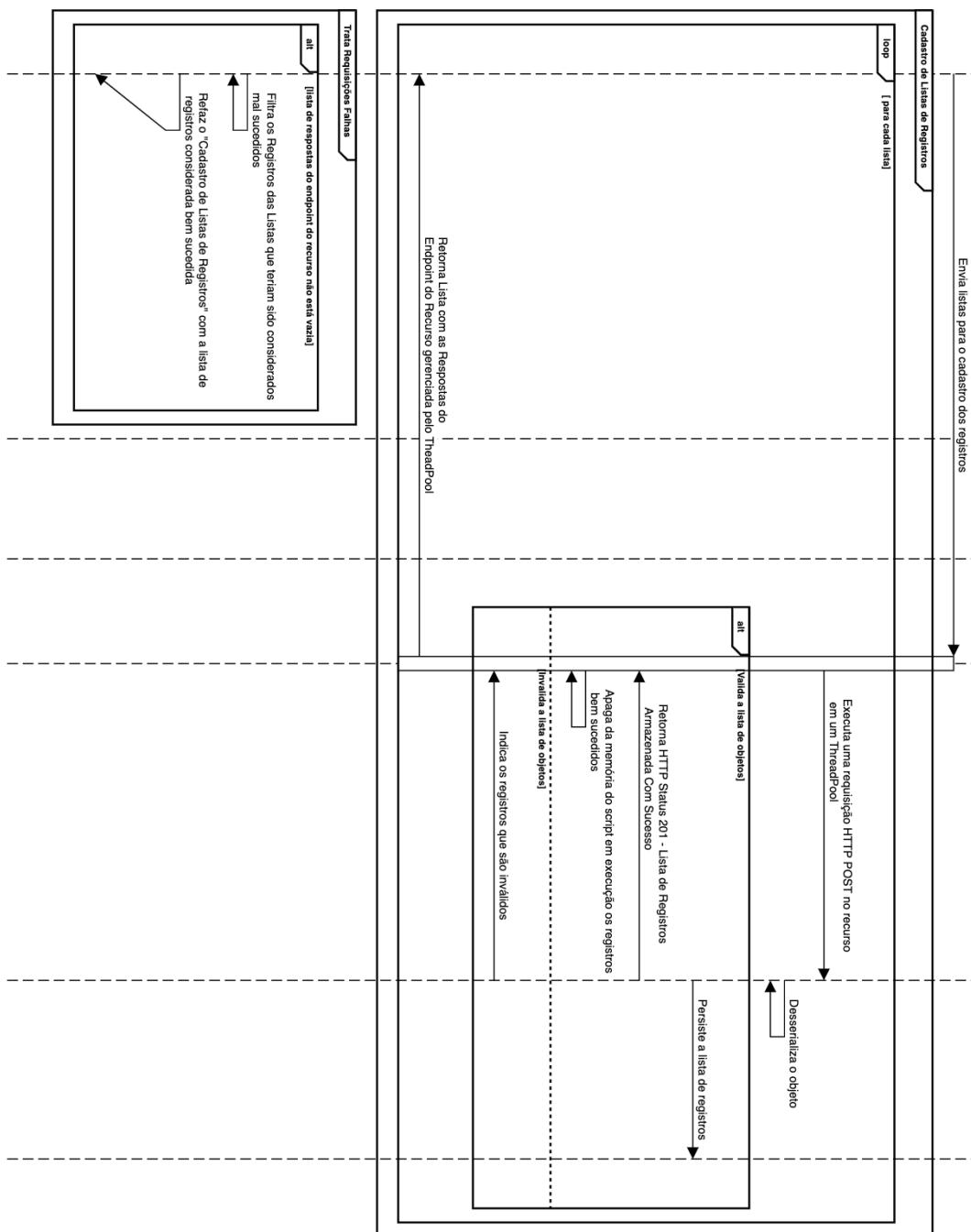
- BARBOSA, S. D. J.; SILVA, B. S. D. **Interação humano-computador**. 1. ed. Rio de Janeiro : Campus, Elsevier, 2010.
- CNJ. CNJ Serviço: Saiba a diferença entre comarca, vara, entrância e instância. **CNJ**, 2016. Disponível em: <<http://www.cnj.jus.br/noticias/cnj/82385-cnj-servico-saiba-a-diferenca-entre-comarca-vara-entrancia-e-instancia>>. Acesso em: 20 jun. 2019.
- CONSELHO NACIONAL DE JUSTIÇA. Justiça em números 2018. **Conselho Nacional de Justiça**, Brasília, p. 214, 2018. Disponível em: <<http://www.cnj.jus.br/files/conteudo/arquivo/2018/08/44b7368ec6f888b383f6c3de40c32167.pdf>>. Acesso em: 28 Novembro 2018.
- DEVMEDIA. Introdução a Web Services RESTful. **DevMedia**. Disponível em: <<https://www.devmedia.com.br/introducao-a-web-services-restful/37387>>. Acesso em: 21 jun. 2019.
- ELMASRI, R.; NAVATHE, S. B. **Sistemas de bancos de dados**. 6. ed. São Paulo: Pearson, 2010.
- GASANOV, P. Django + Docker + AWS EB. **Pavel Gasanov**, 2018. Disponível em: <<https://pogasanov.ru/posts/django-docker-aws>>. Acesso em: 01 mar. 2019.
- IBM. O que É um Servidor Proxy Reverso? **IBM Knowledge Center**. Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SSKTXQ_9.0.1/admin/config/st_adm_port_rvprxy_overview_c.html>. Acesso em: 25 jun. 2019.
- LARMAN, C. **Utilizando UML e padrões**: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007.
- MICROSOFT. O que é o DevOps? **Microsoft Azure**, 1 Junho 2019. Disponível em: <<https://azure.microsoft.com/pt-br/overview/what-is-devops/>>.
- NUNES, M. G. **Jurimetria**: Como a estatística pode reinventar o direito. São Paulo: Editora Revista dos Tribunais, 2016.
- SCHIRMER, L. et al. **Legal Text Analytics: a model for Special Civil Courts jurisprudence analysis**. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, p. 9. 2017.
- SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

Anexo I – Modelagem de Entidade e Relacionamento dos Dados do TJRJ.

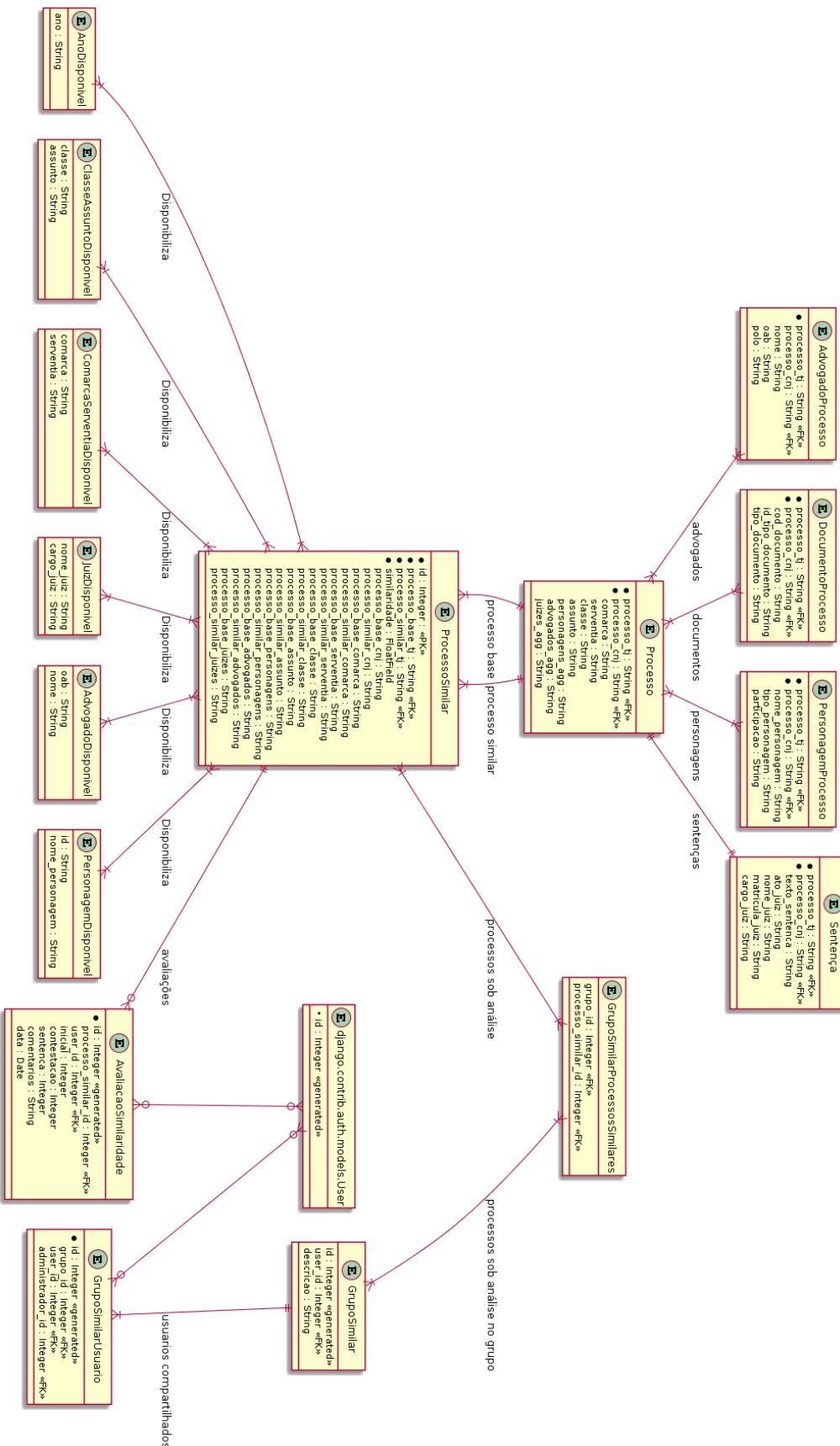


Anexo II – Diagrama de Sequência do Tratamento e Persistência dos Dados.

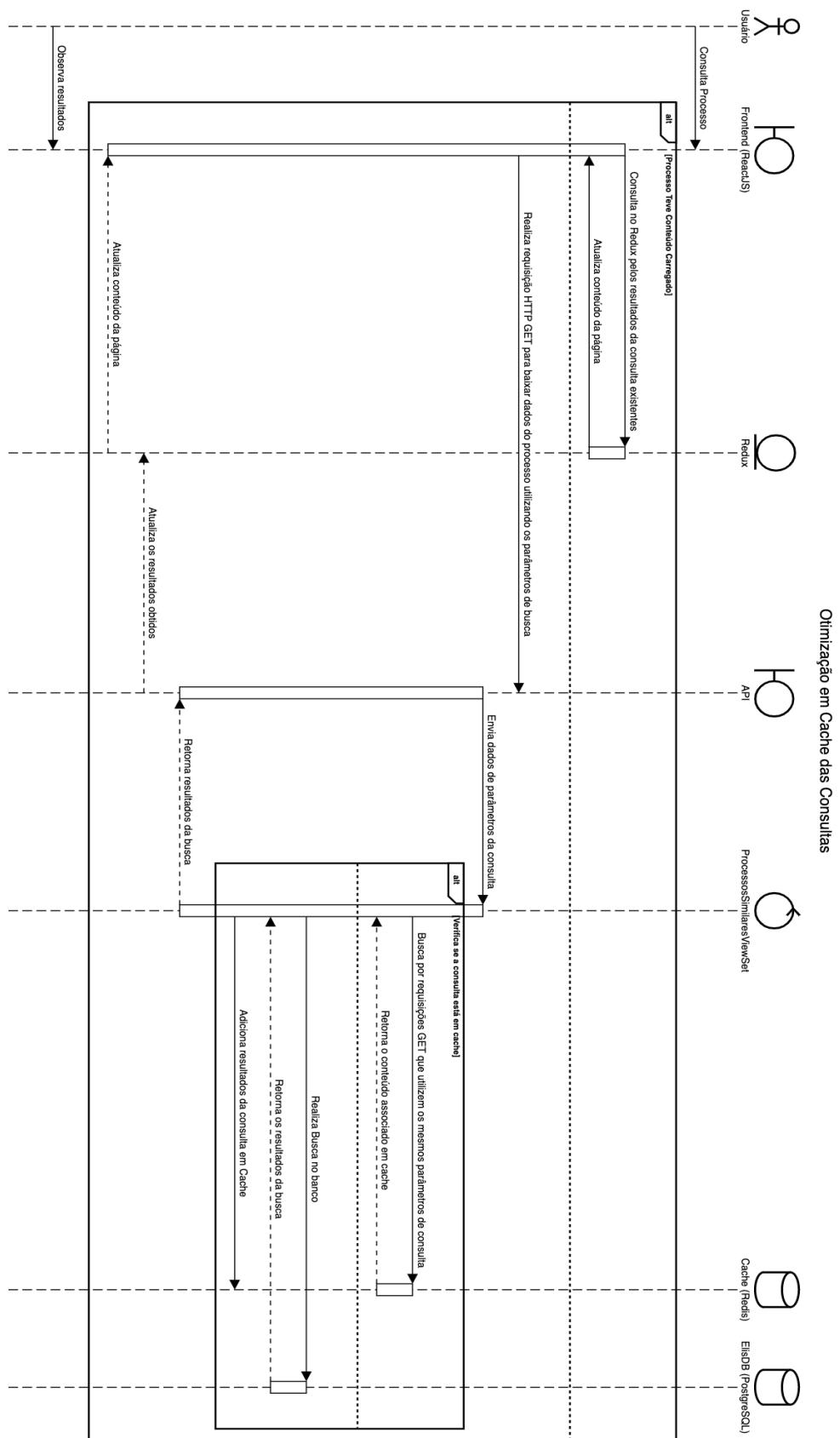




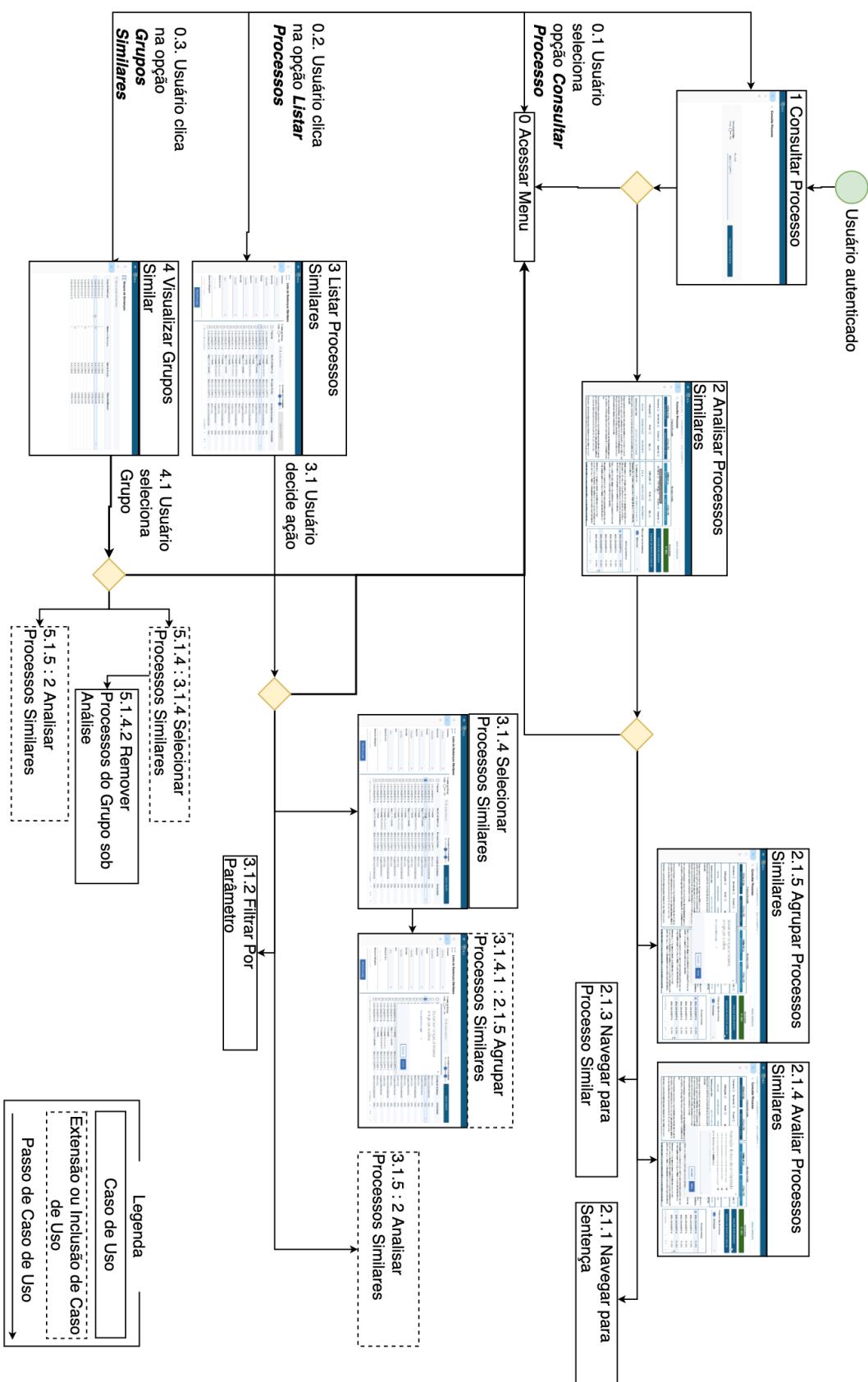
Anexo III – Diagrama de Classes referente ao mapeamento objeto-relacional.



Anexo IV – Diagrama de Sequência da Otimização de consultas



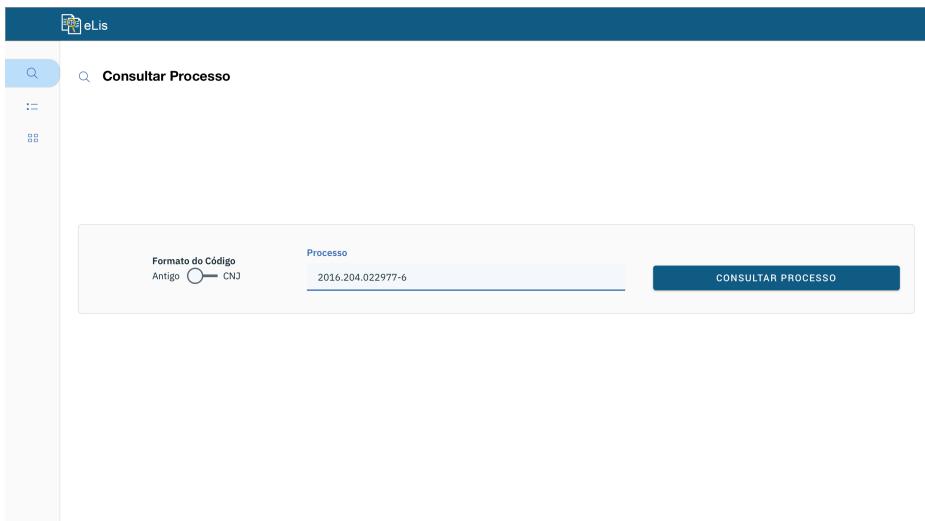
Anexo V – Fluxo de Telas e Casos de Uso Associados



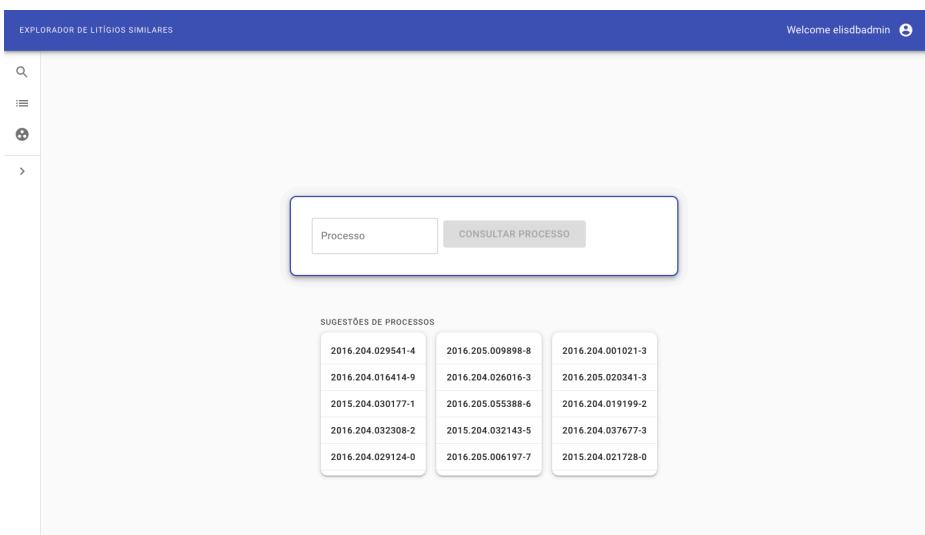
Interface de Consulta de Processo



Versão do Protótipo Original



Versão do Protótipo de Mockup



Versão Final Implementada

Interface de Análise de Processo Similares

The screenshot shows a search interface for similar legal cases. At the top, there's a logo for the Poder Judiciário do Estado do Rio de Janeiro (Poder Judiciário RJ) and a title 'Buscador de Processos Similares'. Below this, there are two tabs: 'Processo Buscado (Código Antigo)' and 'Processo Buscado (Código Atual)'. The 'Código Antigo' tab is selected, showing process numbers 2016.204.013565-4 and 2016.204.001912-6. The 'Código Atual' tab shows process number 0013602-67.2016.8.19.0204. To the right, there are tabs for 'Inicial', 'Contestação', and 'Movimentos' for both the buscado and similar processes.

Processo Similar (Código Antigo)	Processo Similar (Código Atual)	Similaridade
2016.204.001912-6	0001916-78-2016.8.19.0204	66.73%

Below the tabs, there are sections for 'Autor' (João Carlos de Pinha), 'Réu' (Banco Santander Banespa S.A.), and 'Sentença'. The 'Sentença' section contains a detailed text about a legal case involving a bank loan and its cancellation. To the right, a table lists 'Número de Sentenças Similares: 5' with entries 1 through 5, each with a date, process number, and similarity percentage. At the bottom, there are buttons for 'Sentenças Não Recorridas e Confirmada', 'Filtrar', 'Anterior', 'Próximo', and 'Nova consulta'.

Versão do Protótipo Original

This screenshot shows the 'Consultar Processo' page of the original prototype. It features a sidebar with navigation icons and a main content area divided into several sections:

- Processo Buscado:** Shows process numbers 2016.204.013077-2 and 2016.204.022977-6.
- Processo Similar:** Shows process numbers 2016.204.022977-6 and 2016.204.013077-2.
- INICIAL CONTESTAÇÃO MOVIMENTO:** Sub-sections for 'Sentença pelo Juiz' (Judge's Judgment) and 'Sentença pelo Réu' (Defendant's Judgment).
- Advogado, Autor, Réu:** Details for the lawyer, plaintiff, and defendant.
- INICIAL CONTESTAÇÃO MOVIMENTO:** Sub-sections for 'Advogado', 'Autor', 'Réu', and 'Movimento'.
- Filtrar por tipo de sentença:** A dropdown menu set to 'Reformada'.
- Similaridade:** A large green bar indicating a similarity of 91.98%.
- AVALIAR SIMILARIDADE:** A button to evaluate similarity.
- SALVAR EM GRUPO DE ANÁLISE:** A button to save to a group for analysis.
- Resultados de similaridade:** A table listing 7 sentences with their similarity percentages, starting with 2016.204.013077-2 at 91.98%.

Versão do Protótipo Mockup

This screenshot shows the prototype mockup's search interface. It includes a sidebar with navigation icons and a main search form:

EXPLORADOR DE LITÍGIOS SIMILARES

Search fields: Código RJ (2015.001.398623-7) and Código CNJ (0443319-23.2015.8.19.0001). Below the fields are dropdowns for 'COMARCA', 'SERVENTIA', 'CLASSE', 'ASSUNTO', 'JUIZ', 'ADVOGADO', 'AUTOR', and 'RÉU'.

Resultados:

- Caso 1:** Código RJ 2015.001.398623-7, Código CNJ 0443319-23.2015.8.19.0001. Includes sections for 'MOVIMENTOS' and 'OITIVAMENTOS'.
- Caso 2:** Código RJ 2015.204.021581-7, Código CNJ 2015.21701-60.2015.8.19.0204. Includes sections for 'ATO', 'INICIAL', 'CONTESTAÇÃO', and 'MOVIMENTOS'.

Resumo de Similaridade: Indice de Similaridade: 89.14%.

ACTIONS: Buttons for 'AVALIAR SIMILARIDADE' and 'ADICIONAR EM GRUPO'.

Lista de processos similares: A table listing 12 similar processes, such as 2015.204.021581-7 and 2015.205.041905-5.

Versão Final Implementada

Interface de Agrupamento de Processos Similares

Salvar sentenças similares
em grupo análise

Criar ou Escolher grupo ▾

Cancelar Salvar

Versão do Protótipo Mockup

Salvar em Grupo Similar

Adicione os processos similares em um Grupo Similar.

Grupos de Processos Similares

Pagamento Indevido X X | ▾

CANCELAR ENVIAR

Versão Final Implementada

Interface de Avaliação de Processos Similares

Avaliação da taxa de similaridade

A similaridade está adequada considerando o texto da sentença? [escala]

A similaridade está adequada considerando o texto da inicial? [escala]

A similaridade está adequada considerando o texto da contestação? [escala]

Comente sobre a taxa de similaridade

Digite aqui o seu comentário em texto livre

Cancelar Salvar

Versão do Protótipo Mockup

Avaliar Taxa de Similaridade

Deixe a sua opinião com relação a similaridade entre os processos
2015.001.398623-7 e 2015.204.021581-7

De 0 a 10, o quanto considera a similaridade adequada para o documento da Petição Inicial? 8

De 0 a 10, o quanto considera a similaridade adequada para o documento da Contesteção? 8

De 0 a 10, o quanto considera a similaridade adequada para os textos das sentenças? 6

Comentários

O texto possui inconsistências...|

CANCELAR ENVIAR

Versão Final Implementada

Interface de Listagem de Processos Similares

Processo	Decisão de Sentença	Processo Similar	Decisão de Sentença	Similaridade
2016.204.022977-6	Reformada	2016.204.022977-6	Reformada	80%
2016.204.013077-2	Reformada Parcialmente	2016.204.013077-2	Reformada Parcialmente	98%
2016.204.013565-4	Confirmada	2016.204.013565-4	Confirmada	90%
2016.204.022977-6	Negado o Provimento	2016.204.022977-6	Negado o Provimento	80%
2016.204.013077-2	Reformada	2016.204.013077-2	Reformada	98%
2016.204.013565-4	Reformada Parcialmente	2016.204.013565-4	Reformada Parcialmente	90%
2016.204.022977-6	Confirmada	2016.204.022977-6	Confirmada	80%
2016.204.013077-2	Negado o Provimento	2016.204.013077-2	Negado o Provimento	98%
2016.204.013565-4	Reformada	2016.204.013565-4	Reformada	90%
2016.204.022977-6	Reformada Parcialmente	2016.204.022977-6	Reformada Parcialmente	90%
2016.204.013565-4	Reformada Parcialmente	2016.204.022977-6	Reformada Parcialmente	90%
2016.204.022977-6	Confirmada	2016.204.013077-2	Negado o Provimento	80%
2016.204.013077-2	Negado o Provimento	2016.204.013565-4	Reformada	98%
2016.204.013565-4	Reformada	2016.204.022977-6	Reformada Parcialmente	90%
2016.204.022977-6	Reformada Parcialmente	2016.204.022977-6	Confirmada	90%
2016.204.013077-2	Negado o Provimento	2016.204.013565-4	Reformada	80%
1 - 18 de 40 sentenças similares	1 of 3 páginas	<	1	>

Versão do Protótipo Mockup

Processo Referência	Processo Similar	Similaridade
2016.205.031352-8	2016.205.040872-2	87.03
2016.205.040872-2	2015.204.035267-5	86.84
2016.205.040872-2	2016.204.001082-1	86.66
2016.205.038508-4	2016.205.040872-2	86.48
2016.205.040872-2	2016.205.038508-4	86.48
2016.205.040872-2	2016.205.040861-8	86.23
2016.205.040861-8	2016.205.040872-2	86.23
2016.205.040872-2	2015.204.021717-6	85.82
2016.205.040872-2	2015.204.038025-7	85.74

Versão Final Implementada

Interface de Listagem de Grupos de Processos Similares

A interface de usuário (UI) do protótipo mockup é baseada em um design clean com uma barra superior azul escuro contendo o logo "eLis". Abaixo, uma barra de navegação com ícones para busca, filtros e outras funções. O conteúdo principal é dividido em duas seções: uma lateral com uma lista de categorias e uma central com uma tabela de dados.

Categoria Lateral:

- CLARO
- PAGAMENTO INDEVIDO
- INDENIZAÇÕES
- OI

Tabela Central:

Grupo de Sentenças	Número de Sentenças	Data de Criação	Última Modificação
Grupo de Sentença 1	3	01/03/2018	01/03/2018
Grupo de Sentença 2	5	01/03/2018	01/03/2018
Grupo de Sentença 3	30	01/03/2018	01/03/2018
Grupo de Sentença 4	3	01/03/2018	01/03/2018
Grupo de Sentença 5	5	01/03/2018	01/03/2018
Grupo de Sentença 6	30	01/03/2018	01/03/2018
Grupo de Sentença 7	3	01/03/2018	01/03/2018
Grupo de Sentença 8	5	01/03/2018	01/03/2018
Grupo de Sentença 9	30	01/03/2018	01/03/2018
Grupo de Sentença 10	3	01/03/2018	01/03/2018

Versão do Protótipo Mockup

A interface de usuário (UI) da versão final implementada é similar ao protótipo, com uma barra superior azul escuro e uma barra de navegação. No topo, há uma barra azul com o nome da aplicação "EXPLORADOR DE LITÍGIOS SIMILARES" e o nome do usuário "Welcome elisdbadmin".

O lado esquerdo contém uma estrutura de navegação com ícones e uma lista de categorias:

- CLARO
- PAGAMENTO INDEVIDO
- INDENIZAÇÕES
- OI

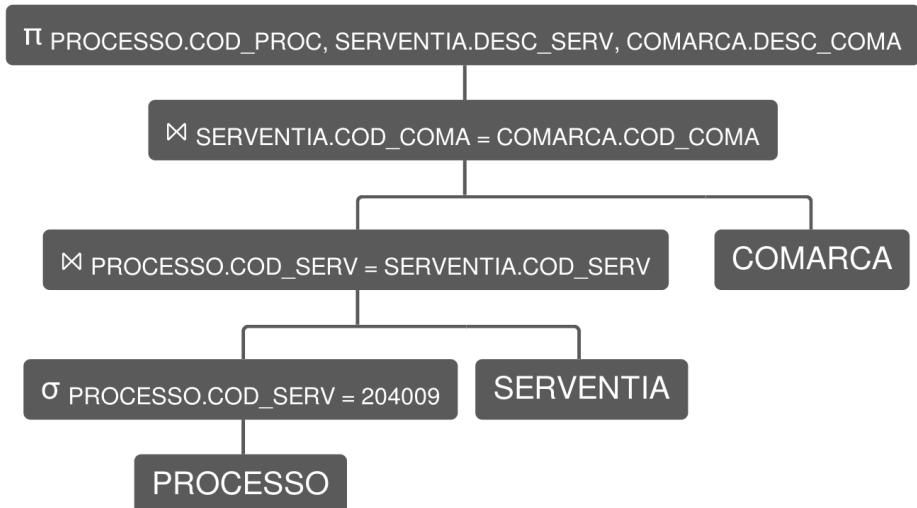
O lado direito exibe uma lista de processos similares para a categoria "OI".

Processo Referência	Processo Similar	Similaridade
2016.205.003658-2	2016.204.027241-4	91.53
2015.204.024854-9	2015.204.022110-6	91.47
2015.205.050950-0	2015.204.001710-2	91.33
2016.204.030375-7	2015.205.050950-0	91.3
2015.001.398623-7	2015.204.027588-7	88.1
2015.204.000175-1	2015.204.036788-5	91.32
2015.204.001710-2	2015.205.050950-0	91.33

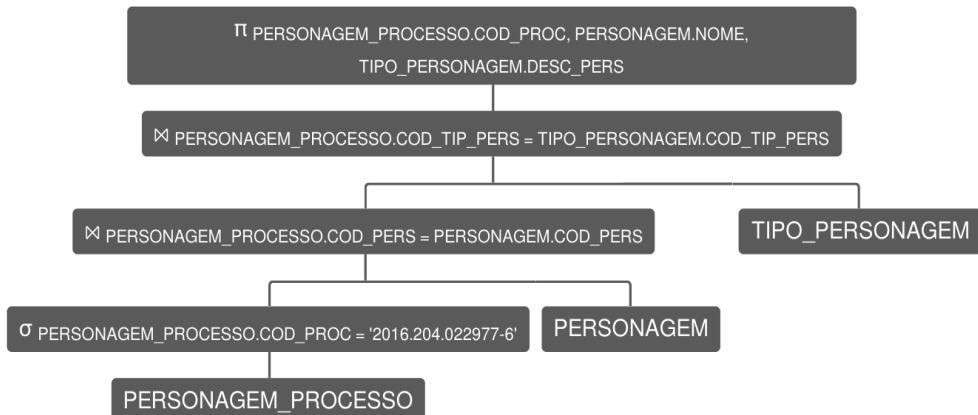
Versão Final Implementada

Anexo VI – Álgebra Relacional básicas para as listagens de processos.

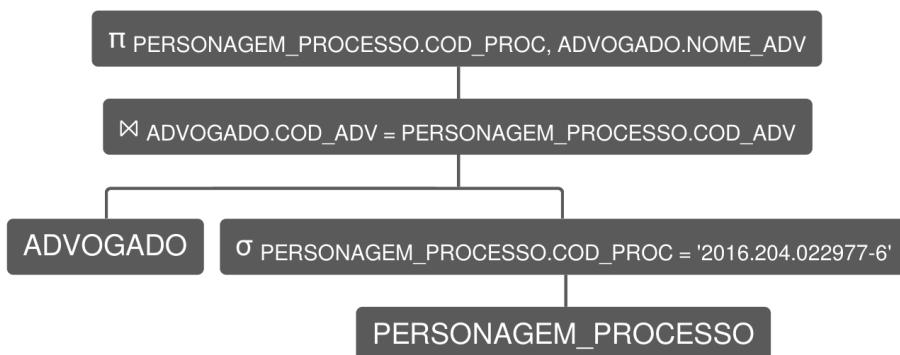
Listar processos de uma serventia específica



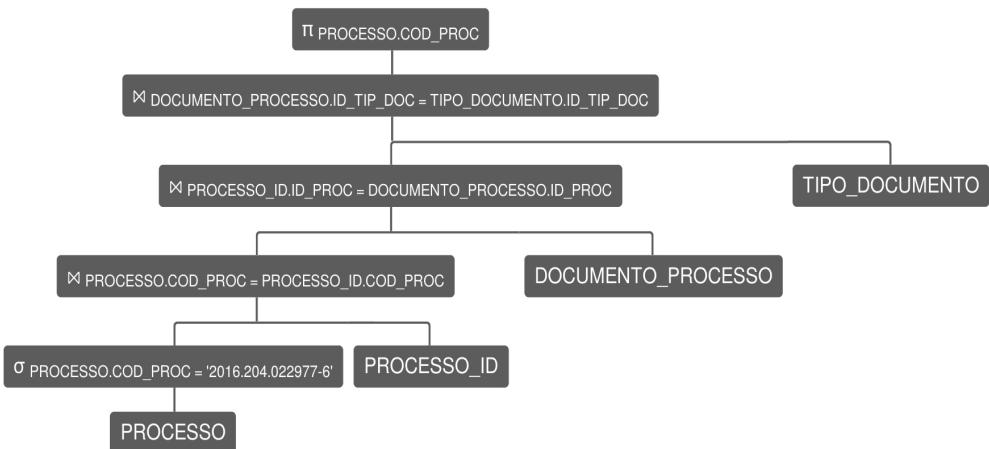
Listar personagens de um processo



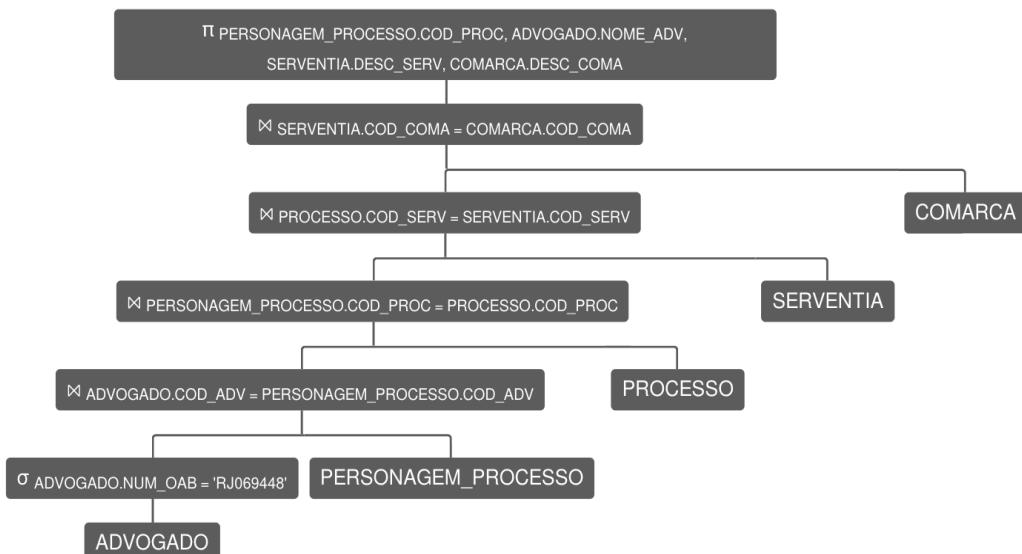
Listar advogados de um processo



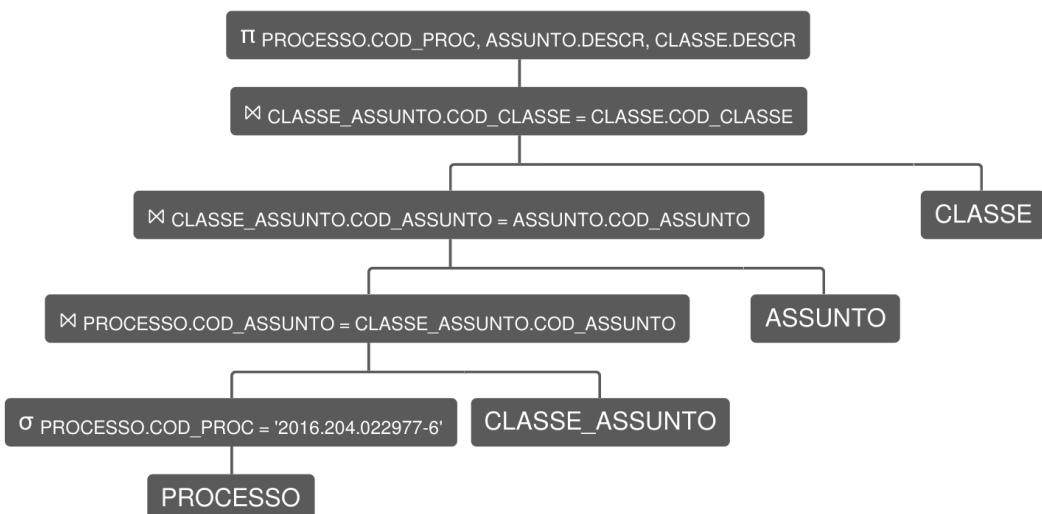
Listar documentos de um processo



Listar processos de um advogado



Identificar Classe e Assunto de um processo



Anexo VII – Manual de Operações da Aplicação

Para facilitar as operações relacionadas a configuração, desenvolvimento, implantação e backup, foi desenvolvido um script *Makefile* com as sequências de comandos de terminal. Para a sua utilização, digite *make* seguido de:

- **create-network:** Cria uma Docker Network, que permite conectar os containers presentes em diferentes arquivos docker-compose.
- **run:** Lança os containers em modo desenvolvedor de elisapp, elisdb (Postgres), Nginx, PgAdmin e Redis Commander.
- **run-prod:** Lança os containers em modo desenvolvedor de elisapp, elisdb (Postgres), Nginx.
- **bash:** Entra no modo terminal do container elisapp em execução.
- **pipenv-install:** Instala os pacotes python presentes no arquivo Pipfile, permitindo desenvolver a aplicação Django.
- **npm-install:** Instala os pacotes javascript presentes no arquivo package.json, permitindo desenvolver a aplicação ReactJS.
- **npm-update:** Atualiza os pacotes presentes no package.json.
- **collectstatic:** permite atualizar os artefatos estáticos, como arquivos HTML e CSS.
- **createsuperuser:** cria um usuário de nível administrador em uma aplicação django.
- **build:** comando para empacotar em uma imagem docker o código da aplicação.
- **build-db:** comando que cria a imagem Postgres que executa os scripts SQL contendo as definições dos esquemas e dados base para o uso da aplicação na sua primeira inicialização.
- **push:** comando para publicar no repositório docker as imagens *elisapp* e *elisdb*.
- **notebook:** Lança um Jupyter Notebook dentro de um container do elisapp em modo desenvolvedor (*make run*), para interação via código Python com o backend ou tratamento e persistência dos dados.
- **notebook-token:** Obtem o link para o Jupyter Notebook.
- **migrate:** Executa o comando Django para atualizar as definições de mapeamento objeto-relacional.
- **run-debug:** Executa em um container no modo desenvolvedor (*make run*) que possibilita o debug da aplicação Django via Visual Studio Code.

- backup-db-schema: Realiza o dump das definições de esquemas, sejam tabelas, visões, visões materializadas e funções, que viabilizam a execução do comando *make build-db*.
- backup-db-data: Realiza o dump dos dados, que viabilizam a execução do comando *make build-db*.
- npm-run-dev: Executa dentro do container elisapp, em modo desenvolvedor, um compilador contínuo da aplicação ReactJS. A cada compilação durante o desenvolvimento é necessário executar o *collectstatic* para atualizar os arquivos estáticos provisionados pelo servidor Django.

Anexo VIII – Cronograma Projeto Final 1

Esperada	Realizada	Tarefa
5/10/18	12/11/18	Estudo de viabilidade e integração com base de dados do TJRJ utilizada pelo modelo de inferência e identificação outros dados candidatos para o suporte a decisão.
5/10/18	9/11/18	Análise exploratória da base de dados do TJRJ.
10/10/18	10/11/18	Modelagem de Entidade e Relacionamento (MER) da base de dados do TJRJ.
11/11/18	11/11/18	Diagrama de Classes da base de dados do protótipo
11/11/18	11/11/18	Identificar similaridades entre o Diagrama de Classes do protótipo e o MER produto da análise exploratória da base dedados do TJRJ.
12/11/18	12/11/18	Verificar com orientador e coorientadora a validade de agregar algumas entidades a interface, como classe, assunto, ato, etc.
13/11/18	13/11/18	Validar os artefatos gerados com o orientador.
14/11/18	15/11/18	Reavaliar a modelagem de dados com base nas observações.
16/11/18	17/11/18	Esboçar consultas possíveis a serem consultas a serem realizadas a partir da modelagem de dados definida.
17/11/18	19/11/18	Definir casos de uso a partir das consultas esboçadas.
19/11/18	20/11/18	Producir protótipos de interface a partir dos casos de uso produzidos.

21/11/18	21/11/18	Validar com orientadores os protótipos de interface desenvolvidos.
22/11/18	22/11/18	Definir com orientadores o perfil dos usuários que a plataforma irá suportar.
23/11/18	23/11/18	Identificar a tabela que possui os textos de sentenças para atualizar o MER.
25/11/18	25/11/18	Avaliar as entidades necessárias para o contexto do sistema aplicado aos advogados.
25/11/18	25/11/18	Identificar complexidade das tabelas a serem tratadas e definir o fluxo de tratamento de dados.
28/11/18	27/11/18	Redigir introdução do relatório do projeto final.
27/11/18	26/11/18	Validar os artefatos com o orientador.
27/11/18	27/11/18	Validar cenários de consulta com orientador e coorientadora.
1/12/18	1/12/18	Revisar protótipos de interface com base nas observações dos orientadores.
2/12/18	2/12/18	Revisar casos de uso e produzir o fluxograma de telas associado aos casos de uso.
5/11/18	5/11/18	Definir arquitetura e requisitos técnicos do sistema
5/11/18	5/11/18	Produzir cronograma do projeto final 2
5/11/18	5/11/18	Consolidar arquivo do relatório
12/12/19	12/12/19	Setup inicial do projeto Django
14/2/19	18/2/19	Testes de integração entre Django e ReactJS
17/3/19	17/3/19	Padronização do Projeto base com Django, ReactJS, Docker e Postgres.
24/3/19	24/3/19	Padronização dos componentes ReactJS para Material UI.
26/3/19	29/3/19	Experimentação de componentes diversos do Material UI e integração da aplicação ReactJS com a API.
31/3/19	31/3/19	Padronização de requisições HTTP para autenticação por Token.
5/4/19	5/4/19	Experimentação de implantação na AWS.
6/4/19	7/4/19	Configuração e testes do Jupyter Notebook integrado ao backend Django.
7/4/19	16/4/19	Configurar servidor Nginx para requisições HTTPS.
19/4/19	19/4/19	Padronização das rotas de recursos da API.

22/4/19	22/4/19	Implementação da classe ElisAPI para auxiliar nas requisições a HTTP com a API.
23/4/19	23/4/19	Tratamento das tabelas de Comarca, Serventia, Competencia, Classe e Assunto.
24/4/19	24/4/19	Tratamento das tabelas de classe e movimento.
28/4/19	28/4/19	Verificar a necessidade de criar logs automatizados para a aplicação.
29/4/19	1/5/19	Implementar requests concorrentes no tratamento das planilhas.
1/5/19	1/5/19	Tratamento das tabelas de decisão de recurso, tipo ato juiz, ato juiz, e tipo de documentos.
2/5/19	13/5/19	Tratamento das tabelas de Processos e Andamentos de Processos
5/5/19	5/5/19	Integração com o Redis para Caching
5/5/19	5/5/19	Experimentação com o RabbitMQ para criar fila de mensagens para o cadastro de registros concorrentes.
18/5/19	18/5/19	Ajustes na organização do código ReactJS com relação a organização dos componentes de interface, Redux e Autenticação
19/5/19	20/5/19	Tratamento de tabelas de personagens (réus e autores) e advogados
20/5/19	20/5/19	Implementação da tela de busca por processo.
20/5/19	20/5/19	Experimentação de integração com o modelo de inferência de similaridades
21/5/19	31/5/19	Implementação da página de análise de processos similares. Incluindo implementação das visões materializadas associadas.
1/6/19	10/6/19	Ajustes no mecanismo de caching com Redux.
5/6/19	17/6/19	Implementação da Listagem de Processos Similares.
11/6/19	11/6/19	Implementação da Avaliação de Similaridade.
11/6/19	11/6/19	Implementação do script de backup do banco de dados.
12/6/19	12/6/19	Implementação do mecanismo de notificações/
13/6/19	13/6/19	Implementação da sugestão de processos disponíveis para consulta.
16/6/19	17/6/19	Criação de Grupo de Processos Similares

17/06/19	17/06/19	Implementação do componente de multiseleção, usado na criação de grupos e listas de processos similares
21/6/19	30/6/19	Implementação de filtros da listagem de processos similares, integrando com a API.
21/6/19	22/6/19	Refatoração das visões e visões materializadas.
10/7/19	10/7/19	Refatoração de interações com a API e uso do Redux.
13/7/19	13/7/19	Implementação da interface de grupo de processos.
14/7/19	15/7/19	Implementação de trigger no banco de dados para permitir a remoção de grupos de processos similares.
20/7/19	20/7/19	Experimentação do Selenium para teste automatizado de interface.
26/7/19	29/7/19	Revisão do texto do relatório e artefatos gerados.
24/7/19	24/7/19	Teste presencial com usuário.