Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

LIBELLUS: DESENVOLVIMENTO DE UM SOFTWARE PARA GERÊNCIA DE PETIÇÕES

LIBELLUS: SOFTWARE DEVELOPMENTE TO MANAGE COMPLAINTS

Ana Paula de Melo de Melo e Souza[1] Gabrielly Alves Cavassini[1]
Marcio Rodrigues Sabino[2]

RESUMO

Uma das muitas funções de um escritório de advocacia é a elaboração de uma peça processual a qual deve ter como base o Código de Processo Civil. De forma geral, os requisitos essenciais para estruturar uma petição inicial são: escolha do juiz ou tribunal a quem é dirigida, os nomes, estado civil, profissão, domicílio e residência do autor e do réu, entrevistas, o fato e os fundamentos jurídicos do pedido, o pedido com as suas especificações, o valor da causa, as provas com que o autor pretende demonstrar a verdade dos fatos alegados, o requerimento para a citação do réu, dentre outras. Utilizando a metodologia *Design Thinking*, a qual propõe soluções inovadoras com o foco nas necessidades do público-alvo, e a metodologia Scrum, eficiente para a organização e produção de *softwares* complexos, realizou-se uma pesquisa de campo com advogados especializados em causas trabalhistas, nosso público-alvo, com os quais foram detectados problemas quanto à falta de formalização das entrevistas, controle de prazos de entregas e envios de documentos. Este artigo tem como proposta o desenvolvimento de um *software* que otimize a elaboração de petições trabalhistas. Espera-se com este, apresentar ao cliente um produto que minimize o tempo e esforço do trabalho, podendo com isso agregar valor ao seu negócio.

Palavras-chaves: Advocacia; *Design Thinking*; Petições; Direito.

ABSTRACT

One of the many tasks of a law firm is the drafting of a legal case document, based on the Civil Procedure Code. In general, the essential requirements for a petition are: the indication of the judge or court to which it is directed; the names, civil status, profession, domicile and residence of the plaintiff and the defendant; interviews, statement of claim, basis for jurisdiction, reliefs, amounts of the claim, proofs that the plaintiff intends to use to demonstrate the truth of the claim, the requirement to notify the defendant, among others. Design Thinking methodology was used, which proposes innovating solutions, focusing on the needs of the target public as well as Scrum methodology, which is efficient to organize and develop complex software. A field survey was carried out with lawyers specialized in labor cases, our target group, with whom it was detected problems regarding lack of formalization in the interviews, difficulties to control deadlines and sending documents. This paper aims at developing of a software program to optimize prepation of labor petitions. It is expected to present a product that minimizes time and effort of the work and thus add value to the business.

Keywords: Advocacy. Complaint. Design thinking.

[1] Tecnóloga em Análise e Desenvolvimento de Sistemas– FATEC Arthur de Azevedo. Rua Ariovaldo Silveira Franco, 567 - Jd. 31 de Março, Mogi Mirim, SP. email: souza.anasd2@gmail.com.
[2] Docente da Fatec Arthur de Azevedo de Mogi Mirim.

Recebido em julho de 2019 e Aceito em novembro de 2020

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

1. INTRODUÇÃO

A rotina de trabalho de milhões de pessoas vem sofrendo mudanças desde a popularização dos computadores. O uso de *softwares* personalizados ou até mesmo os de massa, como o Word e Excel, vêm ajudando o trabalhador a realizar suas tarefas de forma mais eficiente e eficaz. Todas essassão ferramentas necessárias para agilizar as atividades dos mais diversos ramos, inclusive da advocacia.

Há algum tempo era necessário consultar as leis em livros. Hoje, já é possível encontrá-las em *sites* oficiais. O Diário Oficial – instrumento de comunicação no qual são divulgadas todas as informações do estado, inclusive as decisões judiciais sobre o andamento dos processos – também é muito importante para rotina de um advogado. Antigamente, para consultá-lo era preciso ir até o fórum ou ler no jornal; atualmente existem soluções informatizadas que permitem acompanhá-lo sem precisar sair do escritório, além de *softwares* que filtram as informações sobre os processos.

Após uma reunião com um escritório de advocacia, os funcionários relataram a ineficiência em muitos processos rotineiros. Desta forma, surge o questionamento: Como um *software* pode contribuir para a rapidez e agilidade na rotina deste escritório?

Sendo assim este artigo tem como proposta o desenvolvimento de um *software* que otimize a elaboração de petições trabalhistas.

2. MATERIAIS E MÉTODOS

Nesta seção, apresentam-se as metodologias e materiais explanadas que justificam os processos utilizados na elaboração, fundamentação e validação da pesquisa.

O detalhamento da metodologia utilizada com detalhes das fases e elementos desenvolvidos, podem ser encontrados no artigo “DESIGN THINKING E SCRUM: Utilização no desenvolvimento de Software para Gerência de Petições”, dos mesmos autores.

2.1 *Design Thinking*

Vianna et al. (2012) diz que os *designers* veem como um problema, aquilo que afeta a experiência emocional, cognitiva e estética bem como o bem-estar das pessoas. Assim, eles são capazes de identificar problemas. Uma vez que esses foram detectados, o *designer* precisa

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

mapear a cultura, o contexto e experiências pessoais do público. Esse processo permite entender as causas e implicações do problema a fim de gerar soluções mais assertivas.

De acordo com Brown (2010), o *design thinking* é baseado no processo dos *designers* de entender quais as reais necessidades humanas, os recursos disponíveis e integrá-los, criando produtos desejáveis, tecnológicos e viáveis financeiramente ao público. A ideia desta metodologia é aplicar esta técnica em diversos problemas por qualquer pessoa.

Segundo Vianna et al. (2012), o *design thinking* se dá por três etapas: ouvir (imersão), consiste no entendimento inicial do problema e de possíveis oportunidades; ideação, o objetivo dessa fase é gerar ideias através de atividades que estimulam a criatividade; prototipação (criar ou implementar), é onde as ideias se tornam tangíveis e podem ser usadas para validação da solução. As etapas não precisam ser seguidas de forma linear, pois o processo é versátil.

2.2 Engenharia de *Software*

De acordo com Sommerville (2007), *softwares* não são apenas programas de computadores, mas sim todos os dados de documentação e configuração associados os quais são necessários para que o programa funcione corretamente. Segundo Pressman e Maxim (2016), através de um conjunto de métodos e ferramentas, a engenharia de *software* possibilita o desenvolvimento de *softwares* complexos de alta qualidade e dentro do prazo. Ela impõe disciplina ao trabalho, permitindo que as pessoas se adaptem aos métodos mais convenientes as suas necessidade.

2.2.1 O processo de *software*

Para Sommerville (2007, p. 6), “um processo de software é um conjunto de atividades e resultados associados que produz um produto de software.” O autor ainda define quatro atividades fundamentais ao processo: especificação de *software* – o *software* e suas restrições são definidos pelos engenheiros e clientes; desenvolvimento de *software* – o *software* é projetado e programado; validação de *software* – o *software* é verificado para garantir que é o que o cliente deseja; evolução de *software* – é modificado para se adaptar às mudanças dos requisitos.

Segundo Pressman e Maxim (2016, p. 16), o processo é uma abordagem adaptiva que possibilita a escolha de um conjunto apropriado de ações e tarefas para que o projeto seja

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

realizado de forma eficiente dentro do prazo. Eles definem cinco atividades genéricas aplicáveis a todos os projetos: comunicação, planejamento, modelagem, construção e entrega.

2.2.6 Modelos ágeis

Os métodos ágeis introduzem uma nova visão no desenvolvimento de *software*, estando associados a uma nova forma de pensar e às mudanças culturais, possuindo enfoque nas pessoas e não em processos e, além de ter um conjunto de valores, princípios e práticas que possibilitam a adaptação de fatores e resposta rápida às constantes mudanças do mercado (PRIKLADNICKI; WILLI; MILANI, 2014).

Existem diversos modelos ágeis, sendo um deles o Scrum, o qual tem por objetivo auxiliar no gerenciamento de projetos complexos e no desenvolvimento de produtos. Oliveira (2018) defende que o Scrum é leve e simples de entender, consistindo basicamente em papéis, eventos e artefatos.

2.3 Desenvolvimento *web*

Os computadores e dispositivos servem para mostrar informações codificadas a partir de linguagens específicas que são chamadas de linguagens de marcação. Alguns elementos importantes para o desenvolvimento *web* deste trabalho foram:

- **MVC – *Model View Controller*:** trata-se de uma arquitetura de desenvolvimento utilizado para aplicações *web*. Esta divide-se em três camadas, que apesar de distintas, interagem umas com as outras e estabelecem uma relação de dependência. Entretanto, alterações realizadas em uma das camadas não causará problemas às demais, tornando mais fácil e segura a atualização de *layouts*, mudanças de regras de negócio e inclusão de recursos (BASTOS, 2011).
- **PHP:** Niederauer (2011) descreve que com o dinamismo e a necessidade de constantes atualizações nas páginas *web*, a linguagem PHP – ou *Personal Home Page* – possui boas características de interação com páginas HTML estáticas, conferindo dinamismo e praticidade.
- **HTML:** O acrônimo HTML (*HyperText Markup Language*) faz referência à linguagem de programação desenvolvida para o uso em navegadores com o objetivo de divulgação de documentos através da *Internet,* independentemente da plataforma. Através dessa

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

linguagem, todo o conteúdo do hipertexto pode se ligar a outros documentos na rede (SILVA, 2011).

- **JAVASCRIPT:** Trata-se de uma linguagem de programação para *web*, usada para a manipulação de objetos (CROCKFORD, 2008), (FLANAGAN, 2013). Conforme Goodman (2003), o principal objetivo do JavaScript é tornar os sistemas *web* mais interativos.
- **CSS:** *Cascanding Style Sheets* (Folha de Estilo em Cascata), conhecido como CSS, é uma linguagem utilizada para customizar o *design* dos elementos de uma página. Ao programar uma página *web* é definido que o HTML estruturará e marcará o conteúdo, enquanto o CSS terá a responsabilidade pelo visual do documento (SILVA, 2013).
- **POO:** Programação Orientada a Objetos – Uma das principais ideias do conceito do paradigma de POO é a economia de tempo usado para o desenvolvimento de um sistema, com a possibilidade de reutilização de códigos já criados (MANZANO e JUNIOR 2011).
- **Sublime:** De acordo com Otávio (2016), "é um editor de texto bastante fácil de usar, mas com muitos recursos e funcionalidades que podem ser adicionadas para complementar seu uso."

2.4 UML

Segundo Booch, Rumbaugh e Jacobson (2005), a *Unified Modeling Language* (UML) é uma linguagem padrão para estruturação de projetos de *software*. Pode ser utilizada para visualizar, especificar e construir a documentação de artefatos, abrangendo todas as visões necessárias para desenvolvimento e implantação dos sistemas.

Melo (2010) divide os diagramas UML basicamente em: estruturais, cuja função é mostrar as características do sistema que não mudam ao longo do tempo, e comportamentais, que mostram a resposta do sistema às requisições ou como ele evolui durante o tempo.

2.5 Banco de Dados

De acordo com Elmasri e Navathe (2011, p. 3, grifo dos autores), "um **banco de dados** é uma coleção de dados relacionados. Com **dados**, queremos dizer fatos conhecidos que podem ser registrados e possuem significado implícito."

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

2.5.1 Arquitetura e modelagem de dados

Para Silberschatz, Korth e Sudarshan (2006), Elmasri e Navathe (2011) e LucidChart (2018), um modelo de banco de dados define e mostra a estrutura lógica que o descreve, seus dados, suas relações, forma de armazenamento e de acesso, além de suas restrições. Podemos destacar os seguintes modelos:

- **Modelagem Conceitual:** É afirmado por Heuser (2009) que, nesta modelagem, deve-se construir um diagrama entidade-relacionamento, baseando-se nas necessidades e solicitações da organização que armazenará esses dados.
- **Modelo Lógico:** O modelo lógico é utilizado para descrever a estrutura do banco de dados, detalhando como é feito o armazenamento interno de informações, ou seja, define as tabelas e os nomes de suas colunas, de acordo com Heuser (2009).
- **Modelagem Relacional:** De acordo com Silberschatz, Korth e Sudarshan (2006), esta modelagem usa a representação de dados e suas relações por meio de tabelas, onde cada tabela representa um conjunto de entidades e cada uma de suas linhas representa uma ligação, ou seja, uma relação entre um conjunto de valores.
- **Modelo Físico:** Este é o modelo que detalha a estrutura física dos dados, como seus tipos e tamanhos. Esse detalhamento é realizado através de uma codificação em uma linguagem de programação de banco de dados (ALVES, 2014a).

2.5.2 Linguagem SQL

SQL, ou *Structured Query Language* – Linguagem de Consulta Estruturada – é uma linguagem de programação de banco de dados baseada em álgebra e cálculo relacional, e pela qual é realizada a modelagem física de um banco de dados (RAMAKRISHNAN e GEHRKE, 2011).

2.5.3 Sistema Gerenciador de Banco de Dados

Segundo Silberschatz, Korth e Sudarshan (2006), um sistema de gerenciamento de banco de dados, ou SGBD, pode ser conceituado como uma coleção de dados que se relacionam, e uma coleção de programas que permite acesso e manipulação desses dados. O

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

MySQL. trata-se de um SGBD de código aberto e livre, o qual Manzano (2011) afirma ser do tipo relacional e utiliza a linguagem SQL para manipular dados.

3. RESULTADOS E DISCUSSÃO

Neste capítulo será apresentado o desenvolvimento do *software* a partir das técnicas descritas no capítulo anterior. Os autores deste artigo escreveram um outro trabalho intitulado "DESIGN THINKING E SCRUM: Utilização do desenvolvimento de Software para Gerência de Petições", o qual apresenta com detalhes as metodologias *design thinking* e Scrum para o desenvolvimento deste *software*.

3.1 Cliente

O cliente deste estudo de caso é uma advogada que trabalha em um escritório de advocacia situado na cidade de Mogi Guaçu o qual foi fundado há onze anos, pelo advogado Mailson Luiz Brandão. Atua especialmente nas áreas trabalhista e cível, possui clientes de toda a região. Hoje conta também com a advogada Camila Pelegrini – quem nos apresentou as dificuldades de sua rotina e nos solicitou uma solução para otimizar seu trabalho.

Ela ofereceu o escritório para usar de base a criação do *software* podendo futuramente ser implementado em outros escritórios que compartilhem essas dificuldades. A advogada e seu chefe, dr. Mailson Luiz Brandão, não conhecem um *software* que faça o que eles precisam.

O processo de criação foi começado ao marcar uma entrevista. A equipe foi até o escritório para entender as necessidades do cliente e levantar os requisitos necessários começando a fase de ouvir do *design thinking*.

3.2 Fase ouvir

A etapa de ouvir foi iniciada pela entrevista com o cliente, na qual os problemas de sua rotina e necessidades foram expostos. De forma geral, os problemas observados foram:

- Necessidade de otimizar o processo de criar petições iniciais trabalhistas.
- Falta de formalização na entrevista com os clientes antes da montagem do processo.
- Dificuldade em controlar prazos.

Além da entrevista com o cliente, realizou-se a técnica denominada sombra, na qual um membro da equipe assistiu a todo o processo para melhor entender a dinâmica de uma entrevista

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

entre o advogado e seu cliente. De forma geral, isso permitiu que fosse entendido como o advogado formula questões, redige as respostas e como conduz o processo, dados importantes para a modelagem do sistema.

3.3 Fase criar

O processo de criação do *software* foi iniciado com a análise dos dados, obtidos na fase ouvir e uma pesquisa de *softwares* que tivessem a proposta de oferecer a funcionalidade de automatizar a geração de petições. Encontrou-se a aplicação *online* e paga PJUD (2018), entretanto, funcionalidades diferentes foram idealizadas para a criação do novo *software*.

Após identificados os problemas dos clientes e as necessidades não atendidas por *softwares* do mercado, foram levantados os requisitos funcionais e não funcionais. Com base neles foram elaborados os diagramas de classe, de caso de uso e de atividades.

Por meio de uma reunião de *brainstorm,* foram pensados, discutidos e desenvolvidos o nome Libellus para o sistema, assim como o seu logo apresentado na Figura 1.

Figura 1 Logo do software

Fonte: Próprio Autor, 2019.

Com o propósito de validar os requisitos e conseguir gerar ideias para chegar à melhor solução, foi organizado um *workshop* de cocriação com a presença do cliente. Assim, foi elaborada uma atividade para a visualização da petição de uma forma fragmentada a partir da primeira versão do diagrama de classes, permitindo perceber se algo estava faltando, entender o processo e até mesmo ajudar a ter ideias de como ficaria a tela para posteriormente ser prototipada. Obteve-se ideias e sugestões do cliente sobre o que o *software* pode fazer para melhorar o processo em devidos momentos – coisas que somente a equipe de desenvolvimento não poderia imaginar.

A partir dos diagramas desenvolvidos e, também, com a ajuda do *workshop* de cocriação a modelagem do banco começou a ser desenvolvida, iniciando-se pela modelagem conceitual – utilizando a modelagem Entidade-Relacionamento, e depois a modelagem relacional.

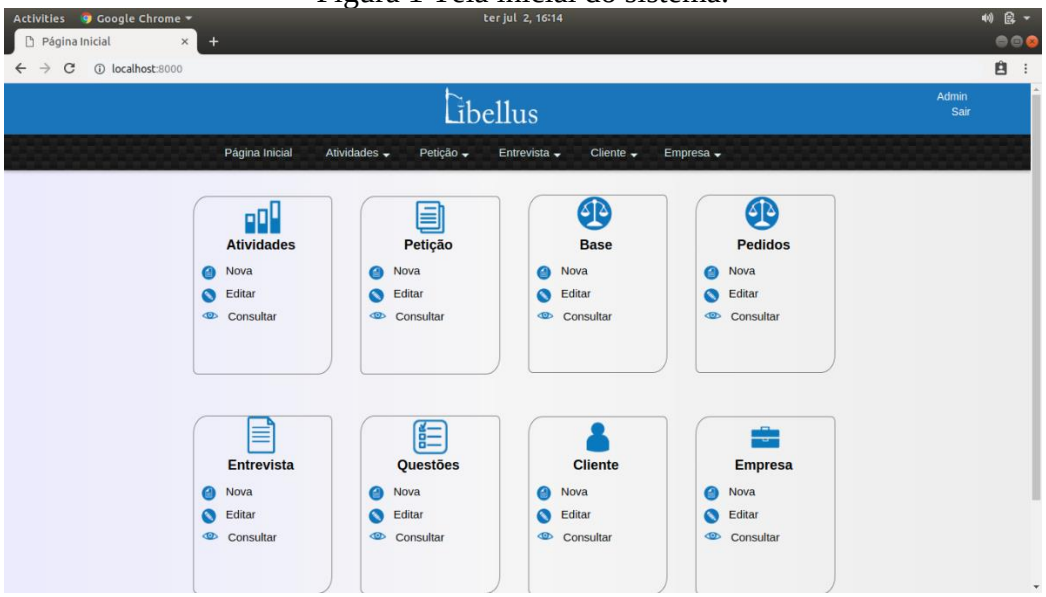
Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

3.4 Fase implementar

A terceira etapa do *design thinking* consistiu em tangibilizar as ideias e propostas das fases anteriores através de protótipos. Para um gerenciamento eficaz do projeto, foi utilizada a metodologia Scrum, designando papeis de *Scrum Master* para o orientador, especificando datas de reuniões e tarefas (*sprints)* e o acompanhamento do andamento do projeto por meio de um *task board*.

Para que todas as necessidades do cliente fossem atendidas, desenvolveu-se o *software* Libellus, cuja tela inicial apresenta-se na Figura 1. Nesta tela tem-se acesso rápido a todas as funcionalidades do sistema de uma forma mais visual.

Figura 1 Tela inicial do sistema.

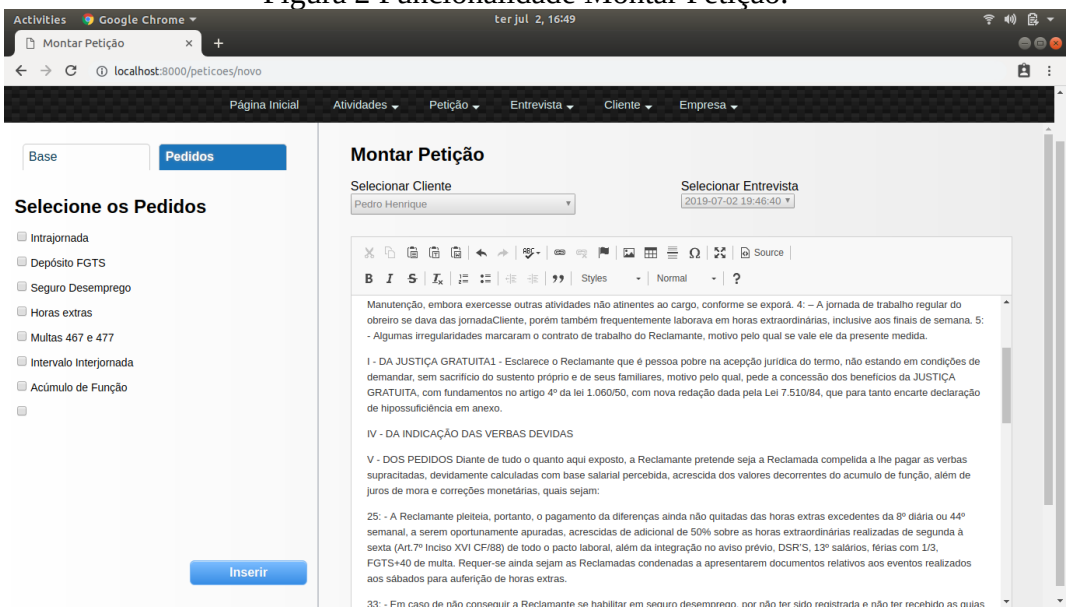


Fonte: Próprio Autor, 2019.

O sistema permite escrever petições trabalhistas a partir de dados cadastrados anteriormente como ilustrado na Figura 2. Essa funcionalidade preenche automaticamente os dados que são comuns em toda petição do que se diz respeito ao cliente, empresa e entrevistas. Os cadastros de base e pedidos se referem a partes comuns entre as petições, evitando o retrabalho de reescrevê-las a cada novo documento.

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

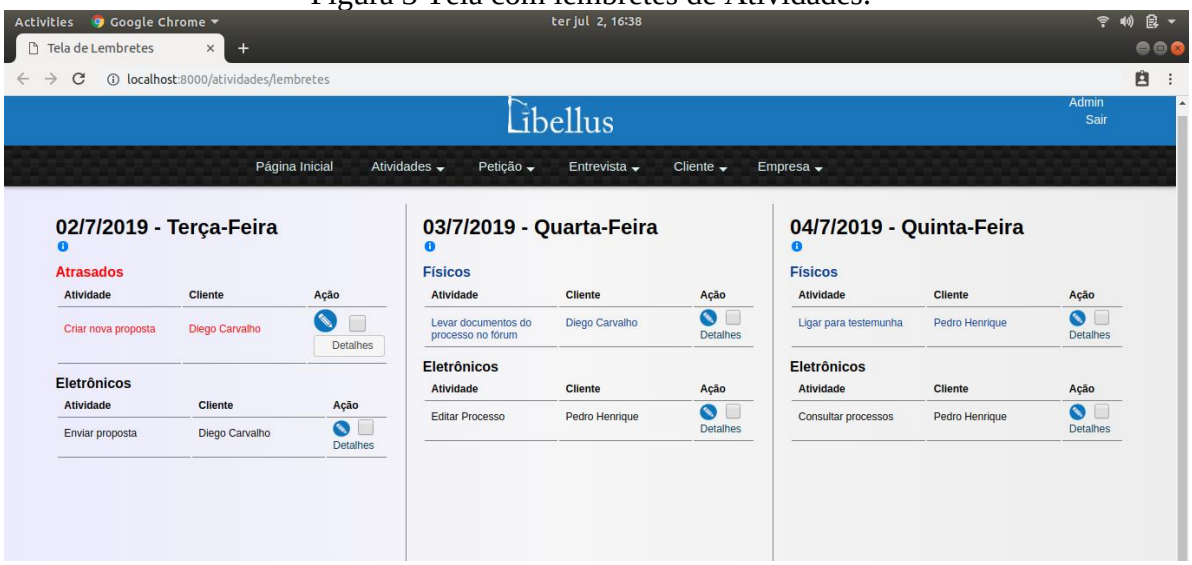
Figura 2 Funcionalidade Montar Petição.



Fonte: Próprio Autor, 2019.

O usuário pode obter lembretes organizados de acordo com a prioridade dos prazos das atividades a serem realizadas (FIGURA 3), tendo uma apresentação lógica e destacada. As atividades atrasadas são exibidas primeiro e sinalizadas em vermelho; o que se refere a processos físicos tem a próxima prioridade pois depende do horário de funcionamento do fórum e são destacados na cor azul; e por fim os processos eletrônicos na cor preta.

Figura 3 Tela com lembretes de Atividades.



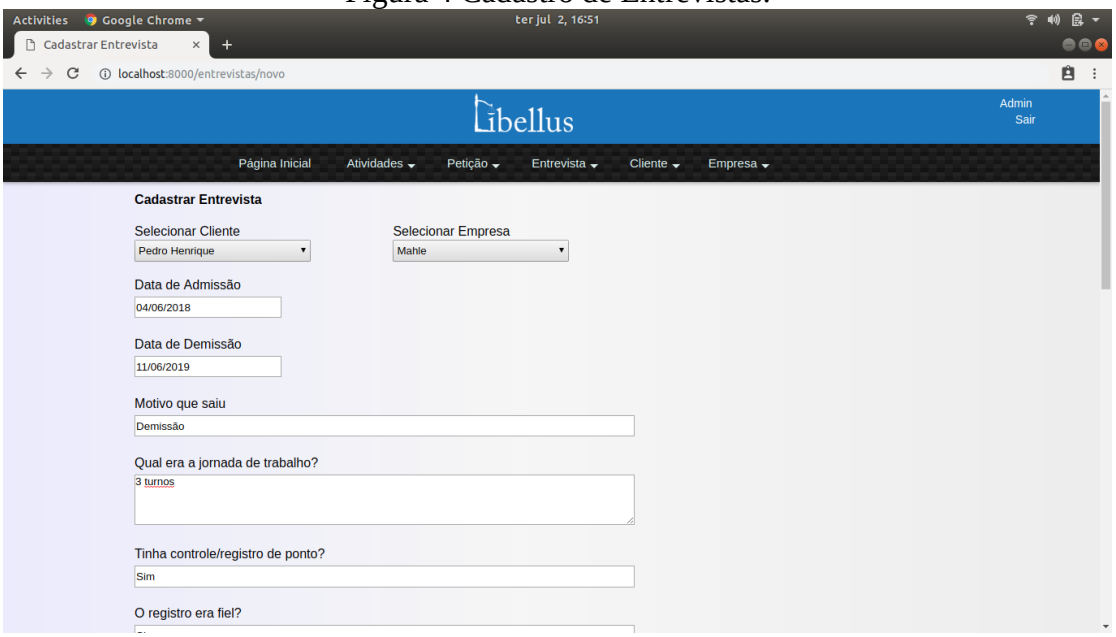
Fonte: Próprio Autor, 2019.

Na Figura 4, pode ser visto que o usuário pode cadastrar entrevistas a partir de perguntas registradas previamente no sistema. As respostas da entrevista são utilizadas ao montar as

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

petições inserindo informações automaticamente e servem como base de consulta para seu desenvolvimento.

Figura 4 Cadastro de Entrevistas.



Fonte: Próprio Autor, 2019.

Todas as informações podem ser gravadas no sistema ou exportada como um arquivo pdf, o qual pode ser impresso para a formalização do processo.

Utilizou-se para o desenvolvimento a combinação das linguagens PHP, HTML, biblioteca JavaScript e, também, CSS, utilizando o *framework* Laravel. O banco de dados foi criado utilizando o MySQL Workbench e emprega a linguagem SQL.

Nesta etapa foi realizado um teste com o cliente, utilizando um mesmo roteiro de criação de petição realizado em fases anteriores, para um teste com *software*. Como objetivo, buscou-se observar a eficiência do *software* no quesito diminuição do tempo.

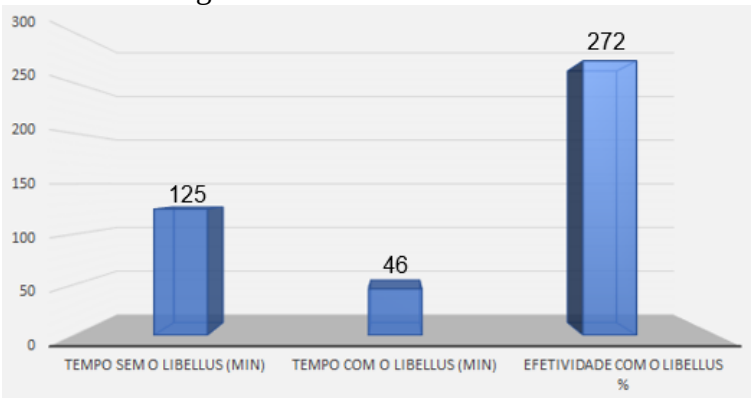
3.5 Estudo de Caso

O cliente deste estudo de caso foi um escritório de Mogi Guaçu, o qual compartilhou as principais dificuldades de sua rotina. Estas foram analisadas e, então, foi proposta uma solução na forma de um sistema informatizado o qual permite: escrever petições trabalhistas a partir de dados cadastrados; obter lembretes, organizados de acordo com sua prioridade, dos prazos das atividades a serem realizadas; acessar questões utilizadas em entrevistas; cadastros; listagens, edições; gravar as informações e possibilitar a sua impressão para a formalização do processo.

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

Realizou-se uma petição sem o *software* e outra com a sua utilização e, como pode ser visualizado na Figura 5.

Figura 5 Efetividade do sistema.



Fonte: Próprio Autor, 2019.

Anteriormente, o tempo necessário para a criação do documento de teste foi 2 horas e 5 minutos enquanto com a utilização da solução, o tempo foi reduzido para 46 minutos, sugerindo que a solução apresentada cumpre seu propósito de agilizar a elaboração de suas petições pela utilização dos dados previamente cadastrados no banco de dados, evitando retrabalho.

Em um comparativo, verificou-se que para elaborar uma petição pelo *software* concorrente, é necessário que o usuário digite diretamente no documento todos os dados de cliente e empresa, além de informações relacionadas à ação (PJUD, 2018). O Libellus possui uma base de dados para que haja cadastro e controle de clientes, empresas e entrevistas, para que a cada vez que uma petição seja criada, o usuário consiga selecionar os dados já cadastrados sem precisar digitá-los, mantendo histórico dos clientes, empresas e das petições, o que possibilita a reutilização e otimização dos dados.

O cliente apresentou um *feedback* positivo, além de algumas sugestões, obtidas por pontos identificados pelo usuário final.

Como forma de apresentação do funcionamento do *software* para a comunidade, foi elaborado um vídeo explicativo das funcionalidades desenvolvidas até o momento, o qual pode ser visualizado no link <youtu.be/5YMQes5df4g>, ou por meio do QR code da Figura 6:

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

Figura 6 Encaminhamento para o Vídeo explicativo do sistema.

Fonte: Próprio Autor, 2019.

4. CONCLUSÕES

A solução Libellus é uma ferramenta *web* que pode ser utilizada por mais de um usuário, e que entrega mais que somente a geração de petições, mas uma forma mais simplificada para o gerenciamento do escritório e de seus processos como um todo.

Ela oferece benefícios como produtividade e eficiência, contando com funcionalidades como: lembretes de atividades, que auxiliam no gerenciamento de prazos e entregas, e formalização de entrevistas, para garantir a autenticidade das informações fornecidas pelos clientes do escritório.

O *software* mostrou-se eficaz, reduzindo o tempo na elaboração de petições, além de verificar que o uso do sistema pode facilitar a rotina do escritório, diminuindo os problemas encontrados e contribuindo para a execução das atividades de forma mais eficiente e eficaz.

Está sendo realizado um planejamento para melhorias, novas funcionalidades e suporte técnico, sempre de acordo com as necessidades dos clientes e do mercado, para assim, transformar o sistema gerado em produto, para que o mesmo possa ser comercializado.

BIBLIOGRAFIA

ALVES, William Pereira. **Banco de dados**. São Paulo: Érica, 2014a.

BASTOS, Daniel Flores. **O que é model-view-controller (MVC)?** 2011. Disponível em: <https://www.oficinadanet.com.br/artigo/desenvolvimento/o_que_e_model-view-controller_mvc>. Acesso em: 01 nov. 2018.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**: guia do usuário. 6 ed. Rio de Janeiro: Elsevier, 2005.

BROWN, Tim. ***Design thinking***: uma metodologia poderosa para decretar o fim das velhas ideias. Rio de Janeiro: Elsevier, 2010.

CROCKFORD, D. **JavaScript**: *The Good Parts*. Sebastopol: O'Reilly, 2008.

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. 6ª ed. São Paulo: Pearson Addison Weasley, 2011.

FLANAGAN, D. **JavaScript**: O Guia Definitivo. 6ª ed. Porto Alegre: Bookman, 2013.

GOODMAN, Danny. **JavaScript**: A Bíblia de Ouro. 3ª ed. Rio de Janeiro: Campus/Elsevier, 2003.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. 6ª ed. Porto Alegre: Bookman, 2009.

LUCIDCHART. **O que é um modelo de banco de dados?** 2018. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-%c3%a9-um-modelo-de-banco-de-dados>. Acesso em: 13 abr. 2018.

MANZANO, José Augusto N. G.; JUNIOR, Roberto Affonso da Costa. **Java 7**: programação de computadores: guia prático de introdução, orientação e desenvolvimento. São Paulo: Érica, 2011.

MANZANO, José Augusto N. G. **MySQL 5.5 interativo**: guia essencial de orientação e desenvolvimento. São Paulo: Érica, 2011.

MELO, Ana Cristina. **Desenvolvendo aplicações com UML 2.2**: do conceitual à implementação. 3 ed. Rio de Janeiro: Brasport, 2010.

NIEDERAUER, Juliano. **Desenvolvendo Websites com PHP**. 2ª ed. São Paulo: Novatec Editora Ltda, 2011.

OLIVEIRA, Bruno Souza de. **Métodos ágeis e gestão de serviços de TI**. Rio de Janeiro: Brasport, 2018.

OTÁVIO, João. **Sublime Text IDE:** Introdução a melhor IDE para desenvolvimento. 2016. Disponível em: <https://www.devmedia.com.br/sublime-text-ide-introducao-a-melhor-ide-para-desenvolvimento/34117>. Acesso em: 31 out. 2018.

PJUD. **Conheça o PJUD**. Disponível em: <https://www.pjud.com.br/sobre-nos/#conteudo>. Acesso em: 20 dez. 2018.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de *software***: uma abordagem profissional. 8ª ed. Porto Alegre: AMGH, 2016.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de software**. Porto Alegre: Bookman, 2014.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistemas de gerenciamento de banco de dados**. 3ª ed. Porto Alegre: AMGH, 2011.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S; tradução de Vieira, Daniel. **Sistema de Banco de Dados**. 5ª ed. Rio de Janeiro: Elsevier, 2006.
SILVA, Maurício Samy. **HTML 5**. São Paulo: Novatec Editora, 2011.

Tekhne e Logos, Botucatu, SP, v.12, n.1, abril, 2021.
ISSN 2176 – 4808

SILVA, Maurício Samy. **JQuery:** A Biblioteca do Programador JavaScript. 3ª ed. São Paulo: Novatec, 2013.

SOMMERVILLE, Ian. **Engenharia de *software***. 8 ed. São Paulo: Pearson, 2007.

VIANNA, M. et al. ***Design thinking***: Inovação em negócios. Rio de Janeiro: MJV Press, 2012.