ALEXANDRE HENRIQUE DA SILVA

UMA ANÁLISE DAS METODOLOGIAS, TECNOLOGIAS E IMPLANTAÇÃO DE SISTEMAS WEB NO LEMAF

LAVRAS-MG
2020

ALEXANDRE HENRIQUE DA SILVA

UMA ANÁLISE DAS METODOLOGIAS, TECNOLOGIAS E IMPLANTAÇÃO DE SISTEMAS WEB NO LEMAF

Relatório de estágio supervisionado apresentado ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso para obtenção do título de Bacharel em Sistemas de Informação.

Orientador

Prof. André Grützmann

LAVRAS-MG
2020

ALEXANDRE HENRIQUE DA SILVA

UMA ANÁLISE DAS METODOLOGIAS, TECNOLOGIAS E IMPLANTAÇÃO DE SISTEMAS WEB NO LEMAF

AN ANALYSIS OF THE METHODOLOGIES, TECHNOLOGIES AND IMPLEMENTATION OF WEB SYSTEMS AT LEMAF

Relatório de estágio supervisionado apresentado ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso para obtenção do título de Bacharel em Sistemas de Informação.

APROVADO em 14 de Agosto de 2020
Prof. Dr. Mauricio Ronny De Almeida Souza - UFLA
Prof. Dr. Rafael Serapilha Durelli - UFLA
Márcio Greison de Azevedo - Zetta

Prof. André Gustmann
Orientador

LAVRAS-MG
2020

RESUMO

O processo de desenvolvimento de software é um conjunto de atividades e etapas que no seu todo, compõe a entrega de um produto com funcionalidades destinadas a atender às necessidades dos usuários finais. O desenvolvimento em si é uma dessas etapas. Diversos tipos de abordagens e metodologias podem ser adotadas para a elaboração, execução e implantação de um sistema. O presente trabalho tem enfoque na etapa de execução do projeto, buscando priorizar a destinada ao desenvolvimento. Para isso, o estagiário ingressou na rotina do Laboratório de Estudos e Projetos em Manejo Florestal, instituição voltada para o desenvolvimento de sistemas relacionados principalmente à preservação ambiental. Este relatório contém a descrição das rotinas, atividades e tecnologias utilizadas para o desenvolvimento dos projetos executados durante o período de estágio. Também será apresentada uma breve descrição dos projetos, com o intuito de contextualizar sobre a escolha do modelo de desenvolvimento e todas as partes que o compõem.

Palavras chave: Desenvolvimento Web, Scrum, Estágio.

ABSTRACT

The software development process is a set of activities and steps that, as a whole, make up the delivery of a product with features designed to meet the needs of end-users. The development itself is one of these steps. Several types of approaches and methodologies can be adopted for the development, execution, and implementation of a system. This work focuses on the execution stage of the project, seeking to prioritize the one intended for development. For that, the trainee entered the routine of the Laboratory of Studies and Projects in Forest Management, an institution focused on the development of systems related mainly to environmental preservation. This report contains a description of the routines, activities, and technologies used for the development of the projects executed during the internship period. A brief description of the projects will also be presented, to contextualize the choice of the development model and all the parts that compose it.

Keywords: Web Development, Scrum, Internship.

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização

Durante todo o período de graduação, o discente adquire o aprendizado e material teórico de seus professores e as experiências das disciplinas. Esse processo de aprendizado é a essência para que o estudante possa se graduar e formar a base para os próximos passos da sua vida profissional e acadêmica. Além de todas essas experiências obtidas durante o período de estudos, o graduando também é incentivado a realizar outras atividades, dentre elas o estágio é o que mais aproxima o graduando das experiências que existem no cotidiano do mercado de trabalho.

O presente relatório é uma análise das atividades desenvolvidas dentro do LEMAF - Laboratório de Estudos em Manejo Florestal, uma instituição voltada principalmente ao desenvolvimento de sistemas relacionados à preservação ambiental, manutenção da flora e benefícios naturais de todo o Brasil.

Os sistemas desenvolvidos pela instituição utilizam de tecnologias modernas e linguagens de programação. Para desenvolvê-los o LEMAF conta com diversos profissionais muito bem capacitados que com a cultura interna da empresa, se habituaram a compartilhar o conhecimento com os novos aprendizes, sejam eles bolsistas, estagiários ou desenvolvedores junior.

O estagiário atuou no desenvolvimento de dois sistemas e, durante o período, observou diversos pontos onde conceitos teóricos foram aplicados a prática. O desenvolvimento de softwares e sistemas é um processo que exige dos desenvolvedores esta mistura de prática e teoria para melhoria da qualidade dos processos de desenvolvimento, ocasionando na melhoria do produto final.

Para início dos trabalhos, o estagiário participou de diversos treinamentos que geraram certificações. Com a conclusão passou então a atuar no desenvolvimento de um sistema para uso interno dos colaboradores da empresa. O Sistema de Gestão de Contratos do LEMAF foi utilizado como parte do processo de aprendizagem sendo desenvolvido por uma equipe em treinamento sob orientação de colaboradores mais experientes. O sistema teve sua estrutura básica desenvolvida porém não foi concluído, sendo esta tarefa delegada a futuras equipes em treinamento.

O outro projeto de atuação durante o período de estágio foi o módulo de validação de

documentos do estado do Pará, um sistema legado desenvolvido para clientes de uso externo a empresa. Este projeto serviu como primeira experiência de trabalho com um produto de uso real. Também foi a primeira atuação em uma equipe de desenvolvimento utilizando o *framework* Scrum.

1.2 Objetivos

O propósito real do processo de estágio é fazer o aprendiz ingressar em um ambiente onde as tecnologias estudadas e conhecimentos obtidos durante a graduação são empregados para o desenvolvimento de produtos e soluções. Dentre diversos objetivos estabelecidos para o estágio, destacam-se:

- Adaptação às rotinas existentes no cotidiano de uma organização voltada para o desenvolvimento de Software;
- Contato com as tecnologias utilizadas para o desenvolvimento de Sistemas Web;
- Participação em experiências de projetos reais e observação de como os produtos são vistos pelos clientes e como as demandas são recebidas e tratadas pelos desenvolvedores;
- Utilização do conhecimento teórico obtido durante a graduação para a colaboração nos projetos;
- Vivenciar a escolha de carreira no mercado de trabalho e possivelmente seguir trabalhando;
- Verificar pontos em que a formação acadêmica pode ter deixado deficiências para então investir em melhorias;
- Estabelecer novos objetivos e caminhos a serem traçados no período posterior à graduação.

1.3 O LEMAF

Com sua localização no DCF - Departamento de Ciências Florestais (UFLA), o LEMAF - Laboratório de Estudos e Projetos em Manejo Florestal possui como principais objetivos: ensino, extensão e pesquisa. Fundado em 2004, o laboratório tem diversos projetos com convênios e parcerias relacionados à órgãos de esfera federal e estadual, e também com a iniciativa privada.

O LEMAF possui três áreas de conhecimento focais:

- Manejo Florestal
    - Administração da floresta para obtenção de benefícios econômicos, sociais e ambientais, respeitando-se os mecanismos de sustentação do ecossistema objeto do manejo (BRASIL, s. d.)
- Geoprocessamento e Sensoriamento Remoto
    - Tratamento de informações geográficas ou de dados georreferenciados por meio de softwares específicos e cálculos (FATEC JACAREÍ, 2016).

- Tecnologia da Informação
    - Soluções providas por recursos de computação que visam produção, armazenamento, transmissão, acesso, segurança e uso de informações (BRASIL, s. d.).

Na área de Tecnologia da Informação, a empresa subdivide seus colaboradores em diversas frentes de trabalho. Cada uma delas possui suas especificidades, sendo assim profissionais específicos e seus respectivos líderes. Essa divisão também acontece com os times de desenvolvimento. Cada time é alocado em uma tribo, que pode ser descrita como uma reunião de vários times e cada tribo tem como líder um(a) Gerente de Projetos. Os projetos são subdivididos e alocados em tribos distintas. Os principais cargos são:

- DBA - Database Administrator ou Administrador de Banco de Dados
    - A administração, gerenciamento e outras atividades relacionadas ao banco de dados e sistemas de bancos de dados são tarefas deste profissional. Cada tribo do LEMAF conta com um ou mais DBAs e os sistemas sob a responsabilidade deles variam de acordo com as tribos.
- Desenvolvedores
    - São os responsáveis pelo desenvolvimento dos sistemas, principal produto oferecido pela empresa. Diversos profissionais da área atuam na empresa, porém nem todos atuam nas mesmas frentes de trabalho ou com as mesmas tecnologias. São alocados em equipes de desenvolvimento, que possuem sua própria tribo. Os níveis de experiência e conhecimento de cada desenvolvedor lhe proporciona

uma classificação:

- Bolsistas - São aprendizes, executam pequenos projetos somente com o intuito de aprender. Trabalham com uma carga horária semanal de 20 horas com uma remuneração equivalente aos valores estipulados em contrato;
- Estagiários - São aprendizes que geralmente possuem mais experiência que os bolsistas. Trabalham com uma carga horária semanal de 30 horas, remuneração prevista em contrato mais benefício de vale transporte (opcional). Atuam em todos os tipos de projetos porém com menos responsabilidades que os desenvolvedores contratados no regime de trabalho *CLT*[1];
- Desenvolvedor Júnior - São desenvolvedores inexperientes que foram contratados para seus primeiros empregos ou que saíram do cargo de estagiário. Trabalham uma carga horária semanal de 40 horas e possuem todas as responsabilidades dos outros desenvolvedores. Porém como não possuem um conhecimento avançado, geralmente não oferecem suporte aos outros desenvolvedores e necessitam de auxílio em diversas situações por ainda estarem aprendendo.
- Desenvolvedor Pleno - São desenvolvedores mais experientes que os Juniores. Conseguem se suprir de acordo com suas necessidades sem contar com ajuda de outros, porém não conseguem oferecer suporte a outros desenvolvedores ainda.
- Desenvolvedor Sênior - São os desenvolvedores mais experientes dentro desta hierarquia. São referência em todos os projetos que trabalham e dão suporte aos outros times quando necessário.

- Gerente de Projeto
  - Gerencia os projetos da empresa que são alocados para suas tribos. Monitoram prazos e estimativas, e alocam os desenvolvedores para os projetos necessários, procurando organizar os times de

[1] Consolidação das Leis do Trabalho

> desenvolvimento da melhor maneira possível. São os líderes das tribos e são os responsáveis por delegar tarefas e instruções para os times de desenvolvimento.

Existem outros cargos que não foram citados, porém os mesmos estão relacionados especificamente ao *framework* adotado pela instituição para os processos de desenvolvimento: o Scrum**.**

1.4 O Estágio

A vaga de estágio é uma oportunidade disponibilizada pela empresa periodicamente. Neste processo, é oferecido um contrato com benefícios de bolsa para o estagiário e vale transporte, sendo o segundo um opcional. O estágio a que se refere este relatório possui uma carga horária de 30 horas semanais, tendo o estagiário autonomia para adequar seus horários de trabalho com seus afazeres do cotidiano.

Esta adequação se torna necessária devido ao fato de muitos estagiários serem alunos de graduação, possuindo a necessidade de cumprir com a carga horária de disciplinas obrigatórias e eletivas de seus cursos. Além disso, a flexibilidade de horários é facilitada pela localização favorável do LEMAF para os estudantes, já que essa instituição se localiza no campus da UFLA e possui vínculos com a universidade. Deste modo, é possível que o estagiário se dirija para aulas no meio do expediente em casos de necessidade.

O período de estágio teve duração total de 720 horas sendo divididos em diversas atividades. Nos primeiros dois meses, o estagiário foi submetido a treinamentos específicos das tecnologias que seriam utilizadas futuramente nos projetos. Em seguida passou por mais um período de aproximadamente seis semanas fazendo um processo chamado de imersão na tribo. Durante esse processo o estagiário atuou como assistente no suporte nível 3, apoiando nas correções e *bugs* reportados por clientes de sistemas em produção. Após o período no suporte nível 3, o estagiário passou então a atuar nos processos de desenvolvimento de sistemas.

1.5 Estrutura do trabalho

Os próximos capítulos possuem os seguintes conteúdos:

- O capítulo 2 traz uma descrição da metodologia utilizada pela empresa para seus processos de desenvolvimento, assim como uma descrição das ferramentas utilizadas para o desenvolvimento dos sistemas;
- Os projetos executados durante o período de estágio bem como as atividades desenvolvidas durante o período são descritos no capítulo 3;
- No capítulo 4 é feita uma análise do que foi obtido de conhecimento e o que foi ganho com o estágio.

2. METODOLOGIA E FERRAMENTAS UTILIZADAS NO LEMAF

Para desenvolver os projetos foram utilizadas linguagens de programação, tecnologias baseadas nestas linguagens além de uma metodologia ágil de desenvolvimento específica para optimização do processo de desenvolvimento. Nesta seção são apresentadas todos os conceitos sobre as ferramentas e metodologias utilizadas.

2.1 *Framework*

"Um *framework* é um conjunto de classes que incorpora um design abstrato para soluções para uma família de problemas relacionados" (JOHNSON; FOOTE, 1988, p. 2, grifo e tradução nossa). Este conceito pode ser aplicado ao se tratar de diversas temáticas, mas em todas elas ele acaba se referindo a um mesmo objetivo: uma sequência de passos, métodos e práticas a serem adotadas objetivando o alcance do melhor resultado. No contexto do desenvolvimento de software, os *frameworks* surgem, na maioria das vezes, como novas tecnologias. Estas novas tecnologias são caracterizadas por serem adaptações e melhorias das linguagens de programação, tornando-as mais simples de serem entendidas e utilizadas.

Assim, os *frameworks* são projetados com a intenção de facilitar o desenvolvimento de software, possibilitando que designers e programadores utilizem mais do seu tempo determinando as exigências do software em detrimento de detalhes de baixo nível do sistema (MACHADO NETO, 2018). No estágio em questão, além dos *frameworks* nas tecnologias para o desenvolvimento de sistemas, também foi utilizado um *framework* para o processo de desenvolvimento: o Scrum. Os *frameworks* diferem das bibliotecas de programação por possuírem um fluxo inverso de controle, já que ele determina quais objetos e métodos devem ser usados quando ocorrem certos eventos (JODAS, 2012).

2.2 Framework Scrum

"Scrum não é um processo ou uma técnica para construir produtos, sendo caracterizado como um *framework* no qual podem ser empregados vários processos ou técnicas" (SCHWABER; SUTHERLAND, 2013, p. 3). No LEMAF, o Scrum é utilizado com algumas adaptações. Alguns dos conceitos deste *framework* são mantidos e aplicados conforme a definição, porém outros são alterados para que se encaixem nos moldes e ideais da organização.

O Time Scrum é subdividido em três equipes, sendo eles: Product Owner, Time de Desenvolvimento e Scrum Master. Todos eles são caracterizados pela auto-organização e multifuncionalidade. Essa auto-organização confere uma característica mais autônoma para os times, já que os próprios podem analisar e escolher a melhor forma de realizar uma atividade e atingir o objetivo pretendido (SCHWABER; SUTHERLAND, 2013).

- Product Owner - PO
    - O *framework* Scrum estabelece como necessário a existência, uma figura central que teria o propósito de representar os interesses dos clientes e se comunicar com o Time de desenvolvimento de forma a repassar ideias e informações. Essa figura também realizaria a comunicação com os clientes, informando a eles possíveis dúvidas e ações realizadas pelos desenvolvedores. Nesse contexto, o Product Owner tem como objetivo fazer com que o valor do produto e o trabalho do Time de Desenvolvimento sejam maximizados. Os métodos utilizados para alcançar esse objetivo são diversos, podendo variar de acordo com as especificidades de cada organização, dos Times Scrum e dos próprios indivíduos (SCHWABER; SUTHERLAND, 2013).

      No estágio, os Product Owners eram caracterizados por tratarem de todas as regras do sistema e traduzirem as ideias dos clientes em formas de protótipos de baixa fidelidade e regras de negócio para que os desenvolvedores pudessem realizar suas atividades. Quando surgiam validações, dúvidas ou imprevistos em relação às regras, os Product Owners eram os responsáveis por desempenhar atividades pertinentes a essas situações para resolver os problemas. Em momentos nos quais essas problemáticas não conseguiam ser solucionadas, os Product Owners marcavam reuniões com os clientes para solucionar os problemas.
- Scrum Master
    - O Scrum Master é caracterizado por agir de forma a possibilitar o entendimento e a aplicação do Scrum. Desse modo, ele procura obter a garantia de que todos os componentes do Scrum - teoria, práticas e regras - sejam aderidos pelo Time Scrum (SCHWABER; SUTHERLAND, 2013). No LEMAF, o Scrum Master solucionava os imprevistos que surgiam de forma a

impedir que atrasos fossem gerados na Sprint. Esses imprevistos eram dos mais variados, podendo ser desde dúvidas quanto às regras dos sistemas, até mesmo problemas com equipamentos e servidores.

No período das atividades do estágio, o Scrum Master realizava ações para blindar o time de desenvolvimento de problemas externos e possíveis imprevistos. Assim, ele sempre objetivava a comunicação com os Gerentes de Projetos, bem como a aproximação de todos. Para isso, havia a realização de reuniões diárias de forma a ocorrer o alinhamento entre todos, sendo que os líderes imediatos eram informados de tudo que era feito nesse processo.

- Time de Desenvolvimento
    - O Time de Desenvolvimento é caracterizado por possuir um tamanho ideal, não podendo ser muito grande e nem muito pequeno. Para que seu tamanho seja considerado ideal, ele deve conseguir atingir dois requisitos: ser ágil e ser capaz de realizar parte significativa do trabalho dentro dos limites da Sprint (SCHWABER; SUTHERLAND, 2013). Durante o estágio os times de desenvolvimento se limitaram ao máximo de três desenvolvedores por squad, sendo compostos por desenvolvedores com níveis diferentes de experiência. Isso permitia que houvesse trocas de experiências e passagem de conhecimento para os desenvolvedores menos experientes.

"O coração do Scrum é a Sprint, um time-boxed de um mês ou menos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado" (SCHWABER; SUTHERLAND, 2013, p. 8). Geralmente nos projetos do LEMAF, as Sprints tem duração de quinze dias úteis, podendo serem realizadas com uma duração menor. Cada uma das Sprints trata de uma série de funcionalidades que podem ser novas, melhorias ou correção de bugs do sistema em questão. A Sprint possui suas próprias rotinas e rituais de início e fim, podendo variar de acordo com a dimensão dos sistemas, disponibilidade dos membros do time, tamanho do projeto, entre outros fatores.

- Reunião Diária
    - No Scrum são realizadas reuniões diárias que objetivam auxiliar o Time de Desenvolvimento a sincronizar e harmonizar a execução de atividades e a estabelecer planos para um período de 24 horas futuros. Nesse contexto, é estabelecido a realização de um "time box" com

duração de 15 minutos (SCHWABER; SUTHERLAND, 2013). Durante as reuniões diárias cada integrante do time tem um pequeno espaço de tempo para falar sobre as atividades que estão trabalhando e quais serão as próximas a serem executadas. Isso trouxe benefícios para a organização, já que possibilitou o alinhamento das ideias de todos, bem como possibilitou que todos se mantivessem informados sobre o que está sendo feito e o que ainda falta para conclusão da Sprint.

- Revisão da Sprint
    - No momento de Revisão da Sprint todos os envolvidos e interessados nela se comunicam para haver o entendimento do que foi feito na Sprint (SCHWABER; SUTHERLAND, 2013). Na Revisão é apresentado tudo que foi feito durante o prazo da Sprint e o que foi entregue, havendo uma comparação com aquilo que foi demandado. No decorrer do estágio houve situações nas quais havia a apresentação de novas demandas em funcionamento no sistema. Apesar disso, esse é um evento opcional, nem sempre ocorrendo nas Sprints observadas no estágio.
- Retrospectiva
    - No momento de realizar a retrospectiva da Sprint, o Time Scrum realiza uma análise de si mesmo, de forma a possibilitar a criação de um plano que oportunizará melhorias para uma próxima Sprint (SCHWABER; SUTHERLAND, 2013). No período de estágio esse evento servia principalmente para apresentar possíveis pontos de melhoria e fatores que se tornaram problemáticos e atrasaram a entrega das demandas. Nessa reunião de retrospectiva, os membros do time possuíam total liberdade para falarem o que achavam de positivo, o que não gostavam e o que poderia melhorar em relação aos processos executados. Dessa forma, o planejamento da próxima Sprint englobava esses pontos considerados na retrospectiva.

2.3 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados que possui como características o código-fonte aberto e o suporte multiplataforma. Ele é caracterizado por seguir o padrão SQL99, com seu gerenciador de transações garantindo algumas propriedades, como: “atomicidade, consistência, isolamento e durabilidade” (TONINI, 2010, p.). É o principal SGBD utilizados nos sistemas de banco de dados do LEMAF e durante o estágio foi utilizado para o desenvolvido e manutenção dos banco de dados dos sistemas. É utilizado dentro da instituição muito pelo fato de contar com uma documentação completa e de fácil acesso na internet, além de ser um SGDB disponibilizado de forma gratuita que conta com todas as funcionalidades que as aplicações exigem, além do suporte a funções de dados geográficos.

2.4 Java

Em 1993 a Internet passou a suportar *WWW*, oportunizando a evolução dos websites. Assim, ela passou a ser compostas tanto por elementos textuais, quanto por elementos gráficos. Nesse contexto, a linguagem Java foi adaptada, permitindo a criação do recurso de Applet (MURTA; WERNER; BARROS; 1998). “Os applets são pequenos programas carregados através da World Wide Web e executados por um navegador na máquina do usuário” (LEMAY E PERKINS, 1996, apud OGEDA, 1998, p. 7).

A grande vantagem da linguagem Java em relação a outras linguagens é que os programas desenvolvidos em Java são independentes de plataforma, ou seja, os programas Java podem ser movidos facilmente de um computador para outro (OGEDA, 1998). Segundo Gonçalves (2008), Java ganhou a simpatia de muitos desenvolvedores no mundo inteiro por ser uma tecnologia muito versátil e oferecer tantos recursos interessantes. No contexto do desenvolvimento dos sistemas do LEMAF, Java é a linguagem *backend* mais utilizada nos projetos desenvolvidos pela empresa. Porém esta linguagem raramente é usada no desenvolvimento Web em sua forma pura, sem nenhum *framework*.

2.5 Spring Boot

O Spring é um *framework* criado com o intuito de simplificar a programação de aplicações Java, sendo fundamentado no padrão de inversão de controle e injeção de

dependências. Isso faz com que o desenvolvedor não precise se preocupar com a instanciação de novos objetos, delegando ao *framework* que instancie o objeto apenas quando for necessário (PICHETTI, 2015).

Para as aplicações desenvolvidas durante o período do estágio, a tecnologia escolhida para o desenvolvimento *backend* foi o Spring Boot. Segundo Pichetti (2015), o Spring Boot é um projeto que engloba todos os módulos já existentes no Spring Framework, mas sua configuração é feita através de classes Java e anotações. Os objetivos do Spring Boot incluem o reaproveitamento dos módulos já existentes, o aumento da produtividade, bem como o melhoramento do entendimento do *framework*.

A visualização panorâmica dos módulos é necessária para que haja a compreensão da abrangência desse *framework*, com o Spring conseguindo englobar todas as necessidades de uma aplicação corporativa (WEISSMANN, 2014). Para os sistemas e projetos desenvolvidos dentro do LEMAF de forma geral, o Spring boot oferece diversos módulos que auxiliam e dão suporte consistente a diversos pontos críticos para a qualidade dos sistemas, como segurança e desempenho, por exemplo.

2.6 JavaScript

JavaScript é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, sendo usada como uma linguagem de script para páginas Web. Apesar disso, o JavaScript também pode ser utilizada em vários outros ambientes sem browser (MOZILLA, 2019). Ele é a principal linguagem de programação utilizada como base para o desenvolvimento de aplicações Web, bem como é a linguagem base para diversos *frameworks* utilizados no desenvolvimento *frontend*. Após o surgimento de novas ferramentas, como o Node.Js, o JavaScript passou a ser executado tanto em browsers, quanto utilizado para o desenvolvimento de aplicações desktop.

2.7 HTML

HTML (abreviação de Hypertext Markup Language) é uma linguagem de marcação utilizada na estruturação de páginas web. Sua sintaxe é bastante simples e baseada em *tags*, que representam os diversos elementos de uma página, desde imagens a links (DevMedia, 2019).

2.8 Vue.js

O Vue.js é um *framework* progressivo de JavaScript utilizado para construir interfaces com o utilizador e que permite que a integração com projetos que já utilizam bibliotecas de JavaScript seja facilitada (CALHAU PINTO, 2017). Isso ocorre porque, ao contrário de outros *frameworks* monolíticos, o Vue.js foi projetado para ser adotável de forma incremental e pode ser facilmente integrado com outras bibliotecas ou projetos existentes (MENDES, 2018). Essa tecnologia foi escolhida para o desenvolvimento *frontend* dos projetos em questão executados no estágio.

O maior diferencial do *framework* Vue.js encontrado pelo estagiário quanto a outros *frameworks*, é o fato dos scripts gerados conterem trechos de Html, Css e Javascript no mesmo arquivo, o que difere bastante de outros padrões de desenvolvimento conhecidos, fazendo com que o tempo para desenvolvimento seja menor, visto que todas as *tags* do html, assim como os estilos do Css e os *scripts* do Javascript possam ser escritos de uma só vez, sem a necessidade da escrita em arquivos distintos. Porém a manutenção em arquivos do Vue.js tende a ser prejudicada pois a leitura se torna difícil devido a quantidade de informações escritas em um mesmo arquivo com três tipos de linguagens diferentes.

3. ATIVIDADES DESENVOLVIDAS NO ESTÁGIO

Este capítulo contém uma descrição das atividades desenvolvidas pelo estagiário durante a execução de dois projetos. Após os treinamentos realizados pela empresa, o primeiro projeto a ser executado foi o Sistema de Gestão de Contratos do LEMAF. Sua duração foi de aproximadamente seis semanas e auxiliou o estagiário como um teste prático. Será apresentada uma análise do desenvolvimento deste sistema e da utilização de suas tecnologias em paralelo ao que foi estudado durante o período de graduação.

3.1 Sistema de Gestão de Contratos do LEMAF

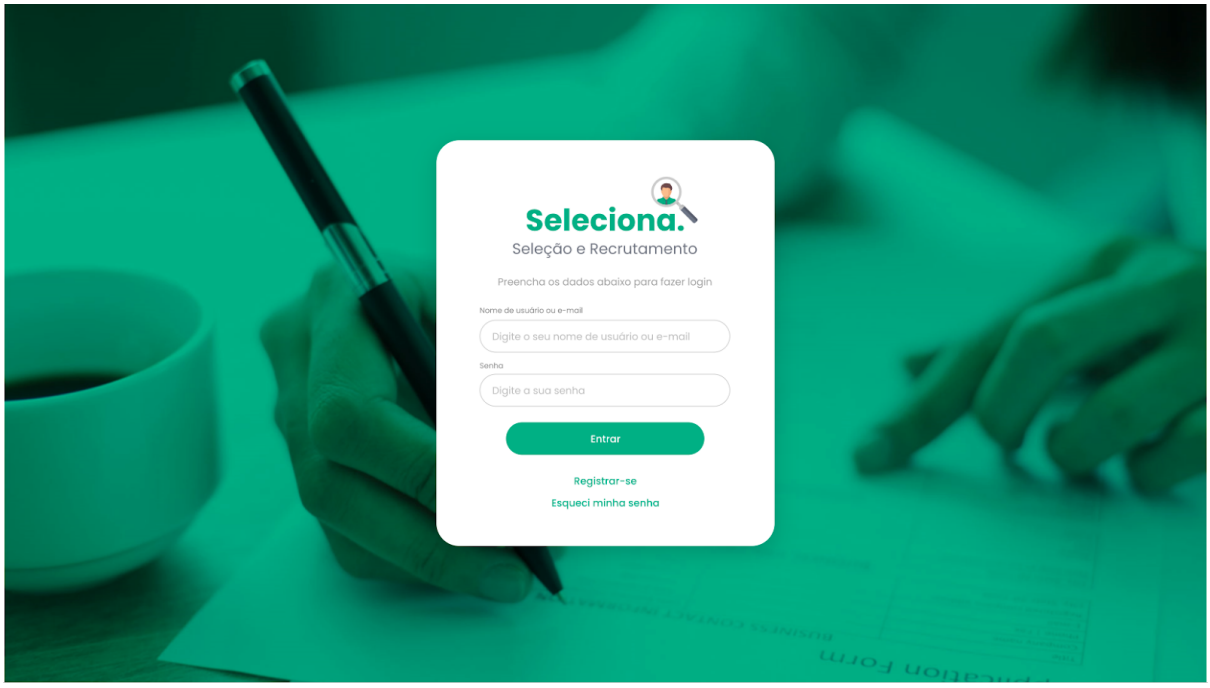
O Sistema de Gestão de Contratos do LEMAF foi descrito como sendo um sistema novo para utilização interna, com o objetivo de gerir os contratos existentes dos colaboradores e também de gerenciar novos contratos e processos seletivos. Considerado um projeto pequeno, o time de desenvolvimento contou com um Scrum Master, um estagiário e um desenvolvedor júnior.

Para adaptar os novos desenvolvedores as rotinas e ritos do Scrum, o Scrum Master sugeriu que tudo desse *framework* poderia ser adaptado neste pequeno projeto. Portanto, diversos elementos do Scrum foram utilizados durante a execução das demandas como reuniões diárias, um quadro para gerenciar as atividades e estimativas de tempo.

As regras e histórias de usuário foram passadas em um documento de regras contendo mais detalhes sobre o que era necessário conter na primeira entrega. Com os integrantes do time com as ideias alinhadas, atividades foram geradas a partir das regras e análise das telas do protótipo de alta fidelidade conforme a Figura 1, que é uma representação da primeira tela do sistema, a tela de login. Essas atividades foram dispostas no quadro de atividades do sistema Gitlab[2], e dessa forma eram feitos os controles do que já havia sido feito e do que ainda era necessário.

[2] https://about.gitlab.com/

Figura 1 - Protótipo da tela Inicial do Sistema de Gestão de Contratos



Fonte: LEMAF (2020)

O sistema seria utilizado pelo setor de recursos humanos do LEMAF e seu desenvolvimento seria dividido em pequenas entregas, sendo utilizado o sistema de Sprint do Scrum, e cada Sprint seria correspondente às demandas de uma entrega.

3.1.1 Tecnologias utilizadas no projeto

Para seguir um padrão coerente ao treinamento, optou-se pelas seguintes escolhas de tecnologias para o desenvolvimento da aplicação:

- Vue.js para o *frontend*
- Spring Boot para o *backend*
- SGBD PostgreSql para o banco de dados

Para o *frontend* foi incorporado o *framework* Vuetify[3], destinado a aplicações responsivas em Vue, baseado no *Material Design*[4]. Ele possui uma boa gama de componentes e uma documentação sólida (VUETIFY, 2018). O *framework* foi escolhido para manter a coerência com o protótipo de alta fidelidade, que foi desenvolvido utilizando o mesmo *Material Design.*

[3] https://vuetifyjs.com/en/
[4] https://material.io/design

3.1.2 Criação do banco de dados

Para o novo sistema era necessário a criação de um banco de dados, visto que as informações deveriam possuir um lugar para serem armazenadas e consumidas quando necessário. Embora o papel de controle dos *scripts* relacionados ao banco seja do DBA, como parte do processo de estágio essa responsabilidade foi passada ao estagiário.

Para dar início às atividades o estagiário começou pela análise das demandas nas regras de negócio. No banco de dados, inicialmente, foram levantadas as tabelas necessárias e quais atributos deveriam estar presentes. Como sendo um banco novo, as suas configurações tanto para criação, quanto para sua estrutura, deveriam estar presentes juntamente com as configurações do *backend* da aplicação, e disponibilizadas em forma de *migrations*. Dessa forma, um novo desenvolvedor que iniciasse seus trabalhos no projeto poderia criar seu próprio banco de dados localmente somente com a execução das *migrations* do sistema.

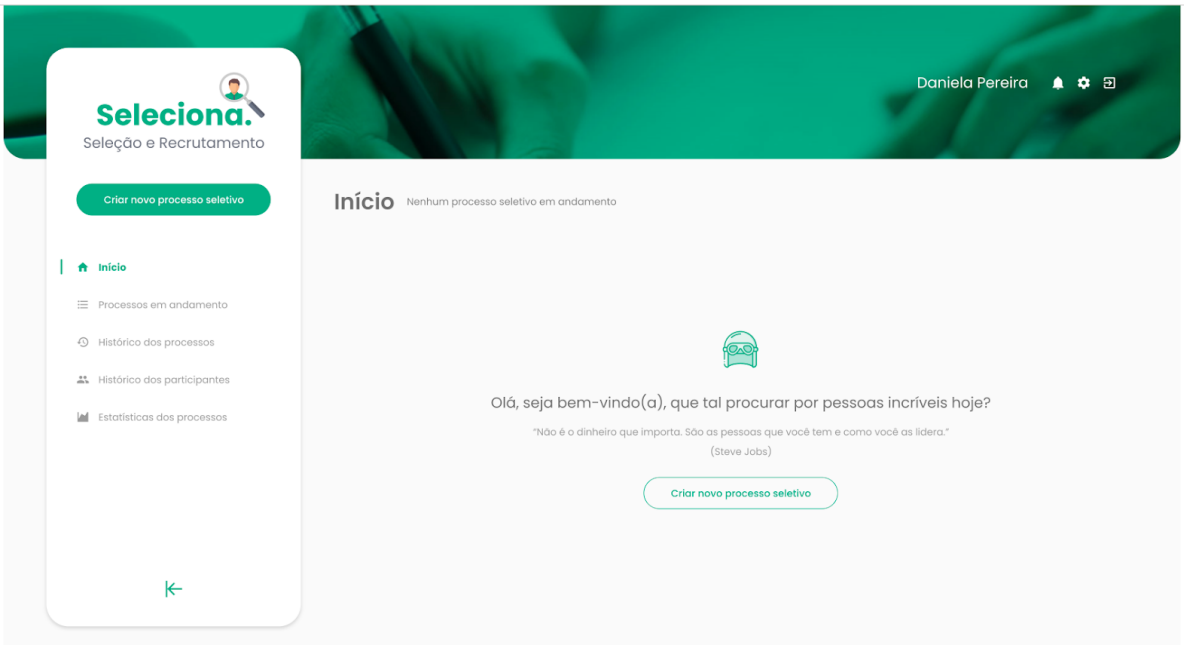
Notou-se que inicialmente, as primeiras tabelas que deveriam existir para o funcionamento do sistema, seriam as tabelas para armazenar dados de usuários. Utilizando dos conceitos aprendidos nas disciplinas de banco de dados, foi modelado um banco via diagramas *ER*[5]. Através destes diagramas ficou evidente de maneira visual, quais relações deveriam existir entre as tabelas, onde deveria existir chave estrangeira, onde deveria existir campos não nulos, entre outros fatores.

Foram criados então, os scripts SQL para geração das primeiras tabelas do banco. Conforme a Figura 1, observa-se a necessidade do armazenamento de dados de um usuário com email e senha. Ambos os campos deveriam ser salvos nas tabelas sendo ambos atributos do tipo de dados *String*, com o detalhe de que a senha deveria passar por um tratamento e por questões de segurança ser criptografada.

Uma das regras do sistema é que após o login do usuário, a tela com as informações internas do sistema, deveria ser exibida, trazendo consigo informações pessoais do usuário logado, conforme exemplificado na Figura 2.

[5] Entidade Relacionamento

Figura 2 - Tela com informações de usuário



Fonte: LEMAF (2020)

Para trazer as informações da pessoa relacionada a um usuário, notou-se a necessidade da criação de mais uma tabela para armazenar os dados desta entidade. O motivo de não armazenar os dados de pessoa e usuário na mesma tabela é que como existiam dados sensíveis na tabela usuario como as informações de login, e pessoa seria uma tabela consultada constantemente, seria mais viável a criação de duas tabelas distintas porém com uma relação de “um para um” entre ambas. Desta forma, quando fosse necessário consultar as informações de uma pessoa, nenhuma informação de usuário seria retornada tornando assim mais seguro as buscas no banco de dados.

Com as entidades pessoa e usuário criadas, e com uma relação entre elas, surgiu mais uma necessidade; garantir a integridade dos dados e que cada usuário existiria somente se uma pessoa existisse no banco. Para isso diversos *inserts* foram criados, populando assim o banco de dados com informações.

Após garantir a integridade das primeiras tabelas, tornou-se necessário a criação da tabela para guardar o perfil do usuário logado no sistema. Esta seria mais uma tabela simples, com poucas informações. Porém se diferenciava das outras pelo fato de ter uma relação com usuário de “muitos para muitos”, visto que um usuário poderia ter mais de um perfil e um perfil pertenceria a mais de um usuário. Isto para as estruturas de banco de dados, exigiria a

criação de uma tabela auxiliar para armazenar as informações de relação de usuário para com um perfil.

Com as estruturas de perfil e da tabela auxiliar que simbolizava a relação de perfil e usuário criadas, mais *inserts* no banco foram criados, tudo isso para garantir a integridade da estrutura de banco e verificar se as informações inseridas referentes a perfil, iriam persistir na base de dados.

3.1.3 Desenvolvimento do *Backend*

O *backend* da aplicação foi desenvolvido utilizando o *framework* Spring Boot. Para auxiliar na tarefa de inserir as configurações na aplicação, o sistema Spring Initializr[6] foi utilizado. Seu propósito é simplificar a tarefa de configurar uma nova aplicação Spring disponibilizando via interface gráfica, diversos elementos para se inserir nas configurações do projeto. Conforme a Figura 3, o sistema Spring Initializr oferece via interface, uma série de opções para a escolha de dependências que existirão no projeto. Além disso disponibiliza opções para a escolha da linguagem de programação que será utilizada. Para o projeto, a linguagem escolhida foi o Java e algumas dependências básicas foram escolhidas, como um plugin para conexão com o banco de dados Postgres e a biblioteca JPA por exemplo.

O Spring Boot possui diversos módulos em forma de biblioteca que ao serem inseridas nas configurações de um projeto novo, são baixadas e importadas no *build* da aplicação. Para que isso aconteça é necessário inserir na aplicação um gerenciador de dependências. O Spring conta com duas opções mais conhecidas e utilizadas pelos desenvolvedores: o Maven e o Gradle. Para o propósito do sistema foi utilizado a primeira opção. “O Maven controla muito bem as dependências de um projeto. Com o uso do Maven, é possível configurar, no arquivo Pom.xml (Project Object Model), todas as dependências da aplicação e dos testes” (ROCHA, 2014, p. 67). O arquivo Pom.xml é escrito em linguagem de marcação *XML*[7] e neste arquivo está contida a lista de configurações das dependências que estarão presentes no projeto.

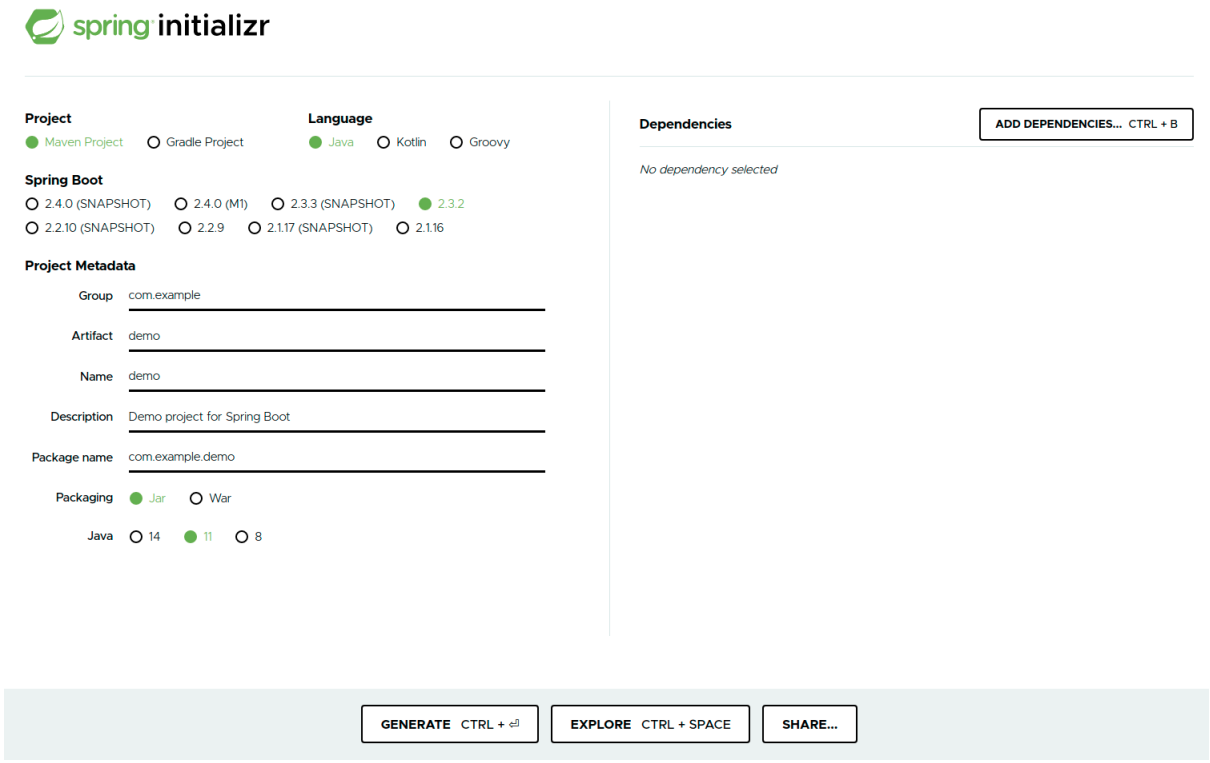
Ao final da escolha de todas as configurações no Spring Initializr um arquivo no formato Zip foi gerado com opção de download. Nele estavam todas as configurações feitas para o novo projeto, além do nome e do *package* que também foram pré configurados.

---

[6] https://start.spring.io/

[7] Extensible Markup Language

Figura 3 - Tela de configuração de um novo projeto no Sistema Spring Initializr



Fonte: Spring Initializr (2020)

Dentro deste arquivo Zip estavam todos os diretórios necessários para a execução de um projeto Java, e no primeiro nível do diretório estava o arquivo Pom.xml. A Figura 4 representa uma das dependências existentes dentro do arquivo Pom.xml do projeto e como é sua sintaxe, ressaltando que o mesmo arquivo foi gerado de forma automática pelo sistema Spring Initializr.

Figura 4 - Exemplo de dependência no arquivo Pom.xml

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
```

Fonte: LEMAF (2020)

3.1.4 Conexões e consultas no banco de dados

As conexões com banco, consultas e outros tratamentos relacionados foram todas feitas através do Hibernate implementado pela biblioteca JPA. “O objetivo do Hibernate é facilitar a construção de aplicações Java dependentes de bases de dados relacionais e particularmente, facilitar o desenvolvimento das consultas e atualizações dos dados” (PEREIRA, 2007, p. 23). Para conectar ao banco de dados é necessário a existência de configurações para que o Hibernate saiba qual a localização do banco e suas credenciais de acesso. Tudo isso foi inserido em um arquivo de configurações gerado juntamente com as configurações básicas do Spring Boot conforme consta na Figura 5, que é um trecho do arquivo de configuração da aplicação *application.properties*.

Figura 5 - Exemplo de configuração para conexão com o banco de dados

```
#datasource
spring.datasource.url = jdbc:postgresql://localhost:5432/sistema-contratacao
spring.datasource.username = postgres
spring.datasource.password = postgres
spring.datasource.driver-class-name = org.postgresql.Driver
spring.datasource.dbcp2.validation-query= = SELECT 1
```

Fonte - LEMAF (2020)

3.1.5 Mapeamento das entidades do banco

Com tudo configurado, o *backend* da aplicação precisava buscar os dados do banco de dados e disponibilizá-los. Para este propósito o hibernate faz através de estruturas pré configuradas o chamado **mapeamento objeto relacional**. As anotações do Java são utilizadas como forma de definir o que cada atributo do banco se tornará em forma de objeto do Java. “As anotações são divididas em duas categorias, as de mapeamento lógico (descrevendo o modelo de objeto, a associação entre duas entidades etc.) e as de mapeamento físico (descrevendo o esquema físico, tabelas, colunas, índices, etc.)” (JBoss, tradução nossa).

Para a entidade mapeada no *backend* ser transformada em objeto no Java, ela precisa ser identificada pela anotação Entity. Isso faz com que o hibernate entenda que aquela classe será equivalente a tabela referenciada por ela no banco de dados. Portanto, os atributos

existentes na classe devem existir também no banco, assim como seus tipos devem ser os mesmos ou equivalentes aos tipos das colunas da tabela no banco de dados. Para as tabelas criadas no banco de dados do sistema foi necessário inicialmente verificar os tipos de dados que ali constavam e a cardinalidade das relações, pois cada um destes detalhes gerou a necessidade da inserção de anotações especiais.

Para a tabela ‘pessoa’, foi criada a classe ‘Pessoa’ com suas respectivas anotações referenciando a tabela no banco e seus atributos. A anotação Column foi inserida em cada um dos atributos da classe e possuía dentro dos seus parâmetros a coluna referenciada por este atributo. Também foram criadas as classes ‘Usuario’ e ‘Perfil’, referenciando no banco as tabelas com seus respectivos nomes.

O relacionamento entre estas entidades foi talvez um dos pontos mais diferentes identificados. Utilizando os conceitos de orientação a objetivo e polimorfismo, um objeto usuário seria teoricamente, uma extensão de pessoa pois todo usuário só existiria se uma pessoa existisse, sendo pessoa uma classe pai e usuário uma classe filha. Porém com a estrutura do hibernate, essa relação deixou de ser feita com os conceitos de orientação a objetos e passou a ser feita com as relações entre as tabelas do banco e suas chaves estrangeiras. Como a cardinalidade verificada na estrutura do banco entre as tabelas *usuario* e *pessoa* é uma relação de “um para um”, no mapeamento do objeto na aplicação essa mesma informação deveria existir, sendo representada pela anotação OneToOne no atributo “usuario” dentro da classe ‘Pessoa’. Dessa forma o hibernate conseguiu mapear nas classes, todos os dados de ‘Pessoa’ e ‘Usuario’ presentes no banco. Para a relação entre ‘Perfil’ e ‘Usuario’ a anotação utilizada foi um pouco diferente. Como a cardinalidade da relação entre ambas tabelas é uma relação de “muitos para muitos”, a anotação utilizada foi a ManyToMany.

Com todos os mapeamentos feitos e todas as entidades referenciadas no banco assim como suas respectivas colunas, tornou-se necessário fazer novamente a mesma verificação feita na criação das tabelas e verificar se os dados eram realmente persistidos nas tabelas. Porém desta vez a inserção deveria ser feita através da própria aplicação. Para isso as operações de consulta e inserção ou alteração de dados no banco foram feitas através de uma interface identificada com a anotação Repository. No Spring Boot essa interface precisa ser uma extensão de uma outra classe com os métodos já implementados do hibernate. Para o projeto a classe utilizada foi a JpaRepository da biblioteca Jpa, previamente configurada nas dependências do Pom.xml. Com isso as consultas ao banco são feitas de forma simples e

prática, sem a necessidade de escritas de consultas SQL. Para se conectar a entidade do banco a Repository faz suas consultas utilizando das classes equivalentes às tabelas previamente anotadas com Entity.

3.1.6 Disponibilização das rotas

Para buscar, alterar, remover ou inserir dados no banco de dados de uma aplicação desenvolvida com Spring Boot, toda aplicação necessita de um controle. Para isso existem as rotas disponibilizadas em forma de serviços. Os serviços são disponibilizados pela classe Controller, uma das classes presentes no Spring Boot e que tem a função de disponibilizar os métodos implementados para serem consumidos por aplicações externas ao *backend*. Para identificar a classe Controller é necessário identificá-la pela anotação Controller e dentro de cada Controller estão os métodos que serão disponibilizados e consumidos através de rotas.

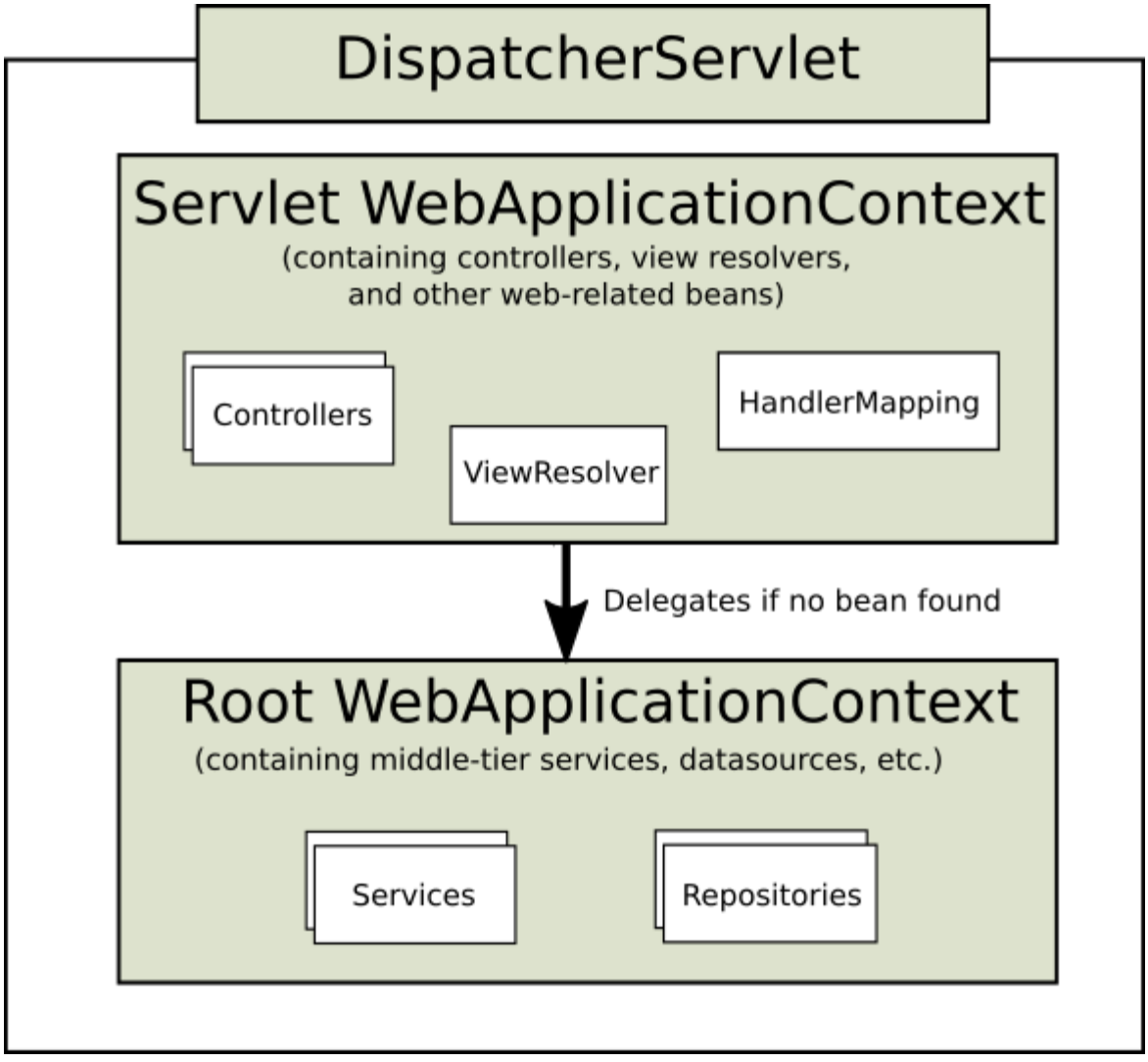
Para a aplicação disponibilizar uma rota para ser acessada através de uma requisição, um método deve ser anotado com uma anotação especial que informa qual será o endereço para consumo deste serviço. No projeto, para a consulta dos dados de uma pessoa, foi necessário a criação da Controller para buscar dados referentes a entidade pessoa. Uma rota com a anotação Get foi inserida no método que executava as funções para a busca no banco de dados. Dentro das propriedades desta rota foi informado o endereço pelo qual ela seria consumida. Destaca-se que para conseguir consumir estes dados a requisição feita para a rota em questão deve ser do tipo *Get*, pois a mesma foi anotada com a anotação que identifica este tipo de requisição.

Para um usuário fazer login no sistema, a rota criada foi a “/login” e inserida no método “login” da Controller ‘UsuarioController’. A requisição do tipo *get*, enviava dados para a aplicação através de um objeto, este objeto era lido através de um biblioteca para fazer seu mapeamento na aplicação *backend*, e então uma busca no banco pelas informações deste usuário era feita. Se ao menos um registro fosse retornado, significava que o usuário existia no banco e seus dados estavam corretos.

Para a classe Controller se comunicar com a entidade do banco, ela pode passar por uma classe específica para completar o fluxo, anotada com a anotação Service. Sua função é processar e tratar os dados que vem ou que vão ao banco de dados. Todo esse fluxo é baseado na arquitetura MVC presente em diversos *frameworks* como o Spring MVC. Segundo Apple (2011, apud JODAS, 2012), o MVC é composto de vários outros padrões de projeto básicos

que trabalham em conjunto para definir a separação funcional e caminhos de comunicação, característicos do MVC. Na Figura 6, um modelo representando o *framework* Spring MVC que serviu como base para a criação do *framework* Spring Boot. No modelo MVC cada uma das classes tem uma única função e são todas baseadas em um fluxo padrão de utilização dos usuários finais.

Figura 6 - Modelo de organização do Spring MVC baseado na arquitetura MVC



Fonte: Spring (sem data).

Para cadastrar um novo dado de usuário no *backend*, primeiramente foi necessário a criação do método *cadastrar* com sua respectiva rota "/cadastrar" que ao ser acessada fazia a Controller 'UsuarioController' ler os dados do objeto recebido via requisição do tipo *post*, identificada pela anotação Post. Se os dados lidos fossem válidos então a classe

‘UsuarioService’ era chamada para fazer o controle dos dados e inseri-los nos seus devidos lugares. Dentro do objeto enviado pela requisição, continham informações de uma pessoa, de um usuário e um perfil, todas as três entidades presentes no banco de dados e mapeadas via *hibernate.* Na Service, era feita a distinção de qual Repository seria chamado. Para salvar os dados de pessoa, a Repository ‘PessoaRepository’ utilizou o método “save()” da biblioteca JPA para salvar os dados de uma pessoa no banco de dados. Na sequência os dados de usuário e perfil foram salvos respectivamente, isso para não quebrar o fluxo, lembrando que no banco de dados um registro de pessoa poderia existir, porém um usuário só existiria se existissem dados de uma pessoa que o representasse.

3.1.7 Criação do *Frontend*

Com o *backend* pronto e os serviços disponíveis para serem consumidos através das rotas, a próxima etapa foi a criação das telas para exibição dos dados. As aplicações *frontend* configuradas no *framework* Vue.js, utilizam de uma estrutura para gerenciamento de dependências semelhante a estrutura do Spring Boot com o Pom.xml, porém no *frontend* existe um arquivo chamado package.json, escrito no formato *JSON*.

A ideia do package.json é semelhante a ideia do Pom.xml; manter em um arquivo com linguagem simples somente para interpretação, uma relação de todas as dependências que estarão presentes no projeto. Dessa forma, configurar o mesmo projeto em um ambiente diferente se torna uma tarefa menos complicada, visto que as aplicações *frontend* também possuem um gerenciador de dependências.

Para a execução da aplicação *frontend* foi utilizado ambiente Node.js. Como um ambiente de execução JavaScript assíncrono orientado a eventos, o Node.js é projetado para desenvolvimento de aplicações escaláveis de rede (Node). As dependências do projeto são baixadas no ambiente quando a ferramenta Npm[8] é executada. Npm, é em primeiro lugar, um repositório online para a publicação de projetos Node.js de código aberto e segundo é um utilitário de linha de comando para interagir com o referido repositório que auxilia na instalação de pacotes, gerenciamento de versões e gerenciamento de dependências (Node).

Mantendo o padrão do *framework* Vue.js escolhido como tecnologia para o desenvolvimento do *frontend*, foi utilizado um conceito de *components* para a divisão entre os

[8] Node Package Manager

arquivos que compõem a aplicação, conceito este vindo do próprio *framework* Vue. Uma *component* seria basicamente um arquivo que pode representar uma tela do sistema, uma tabela com informações, um botão que pode emitir uma ação, por exemplo. A ideia das *components* é a reutilização das mesmas em diversas etapas do sistema que necessitarem.

Para desenvolver uma *component* é necessária a criação de um arquivo com extensão Vue, a extensão padrão do Vue.js. A estrutura inicial para uma nova *component* é um arquivo contendo um *template,* os *script* e o *style.* Na Figura 7 está a configuração inicial de um *component* do Vue.

Figura 7 - Exemplo da estrutura padrão do Vue.js

```
<template>
 </template>

<script>
export default {
 name: 'HelloWorld',

 data: () => ({}),
};
</script>

<style scoped>

</style>
```

Fonte: LEMAF (2020)

No template estão todas as *tags* do Html e é onde os elementos da *component* são criados e organizados de acordo com o ideal. Vale ressaltar que propriedades próprias do Vue.js existem e servem para funcionalidades exclusivas do *framework*, são as chamadas diretivas.

Diretivas são atributos especiais com o prefixo v-. Espera-se que os valores atribuídos

às diretivas sejam uma simples expressão Javascript (Vue.Js).

- v-if: um operador condicional do JavaScript utilizado no template.
- v-bind: utilizado para atualizar um atributo html reativamente.
- v-on: utilizado para o gerenciamento dos eventos.

Para a navegação entre as telas e páginas do sistema, o Vue.js também possui uma ferramenta própria para o propósito, o Vue Router. Com essa ferramenta é possível fazer a troca de páginas através de url de forma centralizada e organizada, tudo isso através de configurações no arquivo ou arquivos de rotas. A Figura 8 representa o arquivo *router* da página de login do sistema. No caso quando acessado o endereço da aplicação *frontend* via *browser* juntamente da rota "/login", o usuário era redirecionado para a tela da *component* 'Login' (Figura 11).

Figura 8 - Exemplo de rota do Vue Router (Rota da tela de login do sistema)

```
import Login from "./pages/Login.vue"

export default [
  {
    name: 'login',
    path: '/login',
    component: Login
  }
]
```

Fonte: LEMAF (2020)

Para a estilização dos elementos das telas e *components*, foi utilizado o *framework* Vuetify. O mesmo forneceu diversas classes css e *components* já implementados e pré estilizados, o que tornou mais simples a tarefa de organizar as estruturas css. Dessa forma, a tarefa dos desenvolvedores passou a ser mais focada nas funcionalidades mais importantes para o sistema, como integridade dos dados, confiabilidade e segurança das informações. Além disso, o Vuetify ajudou no conceito de **usabilidade**, foco de estudo durante a graduação. As ações do usuário na tela eram realizadas de forma dinâmica, sem necessidade

de recarregamento de páginas, somente com a troca de informações na tela. Esse é um dos princípios da reatividade e uma das características do *framework* Vue.js. A Figura 9 contém um exemplo de como é feita a chamada de uma component já implementada do Vuetify.

Figura 9 - Exemplo de component do Vuetify

```
<v-flex
      mb-5
      xs12
    >
      <h2 class="headline font-weight-bold mb-3">What's next?</h2>

      <v-layout justify-center>
        <a
          v-for="(next, i) in whatsNext"
          :key="i"
          :href="next.href"
          class="subheading mx-3"
          target="_blank"
        >
          {{ next.text }}
        </a>
      </v-layout>
</v-flex>
```

Fonte: LEMAF (2020)

Para o tratamento dos dados e processamento de funções, o Vue.js tem sua estrutura para armazenamento de scripts. Neste trecho do arquivo Vue estão todos os scripts de controle dos dados divididos em pequenos métodos, que servem para fazer o controle do chamado ciclo de vida da instância Vue. Cada instância Vue passa por uma série de etapas em sua inicialização - por exemplo, é necessário configurar a observação de dados, compilar o *template*, montar a instância no DOM, atualizar o DOM quando os dados forem alterados. Ao

longo do caminho, ocorrerá a invocação de alguns gatilhos de ciclo de vida, oferecendo a oportunidade de executar lógicas personalizadas em etapas específicas (Vue.js).

Com as estruturas das telas desenvolvidas foi necessário fazer a integração e consumo dos dados vindos da aplicação *backend*. Para isso o ambiente Node.js disponibiliza uma ferramenta chamada Axios. Basicamente sua função é se comunicar com outras aplicações consumindo serviços através de rotas. Toda comunicação é feita através da troca de mensagens no formato Json.

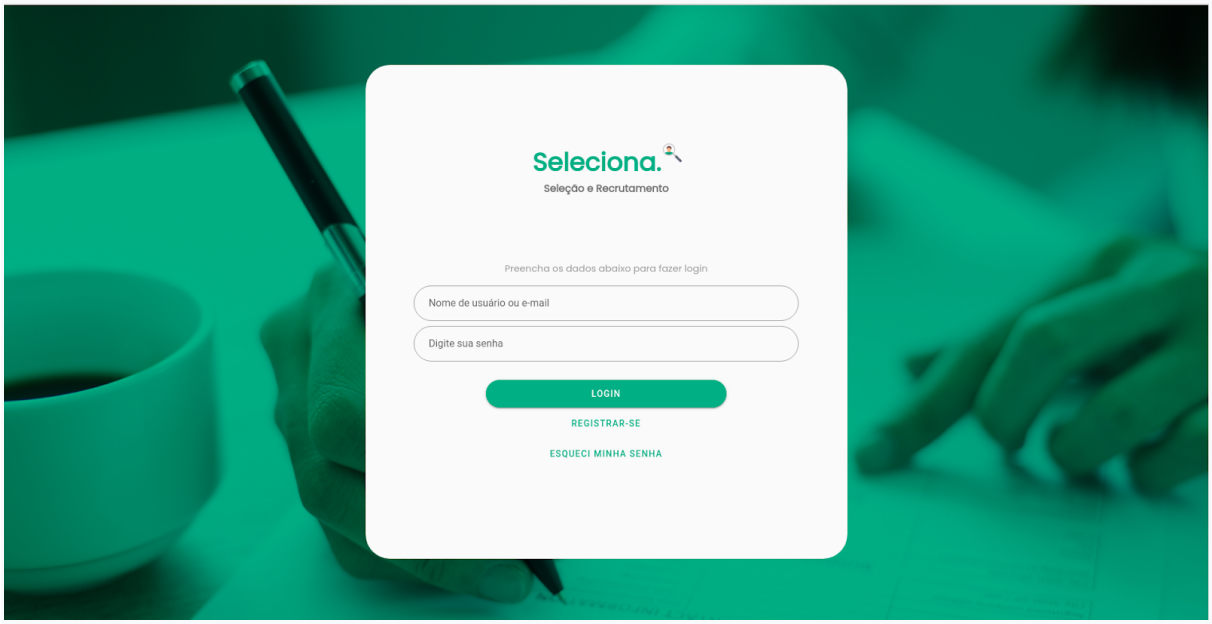
Figura 10 - Exemplo de configuração do Axios

```
import { http } from './config.js'

export default {

  listar:() => {
     return http.get('perfil/')
  },

  autenticar:(perfil) => {
     let email = perfil.email,
        senha = perfil.senha;
     return http.post('login/', { email, senha });
  }

}
```

Fonte: LEMAF (2020)

Como o escopo da Sprint se limitou a criação das telas de login, assim como suas funcionalidades, o projeto não teve continuidade, sendo o estagiário alocado para outro projeto. A Figura 11 é o exemplo da tela inicial do sistema em execução.

Figura 11 - Tela inicial do sistema



Fonte: LEMAF (2020)

3.2 Sistema Módulo de Validação de Documentos do Estado do Pará

O próximo projeto executado foi uma demanda relacionada ao Sistema Módulo de Validação de Documentos do Estado do Pará. Diferentemente do projeto anterior, este foi uma demanda solicitada por usuários externos da empresa, sendo um produto vendido pelo LEMAF. O time de desenvolvimento contou com dois desenvolvedores experientes, um Product Owner além do estagiário. Serão relatadas todas as atividades, rotinas e processos do Scrum presentes na etapa de planejamento e execução desta demanda.

O Módulo de Validação de Documentos é um sistema desenvolvido para o Estado do Pará e tem como finalidade, validar os documentos vindos de outros módulos presentes dentro de um sistema chamado Projeto de Regularização Ambiental (PRA), que também foi desenvolvido pelo LEMAF.

Antes da execução da demanda, o sistema contava com três perfis, cada um com sua respectiva finalidade dentro da aplicação:

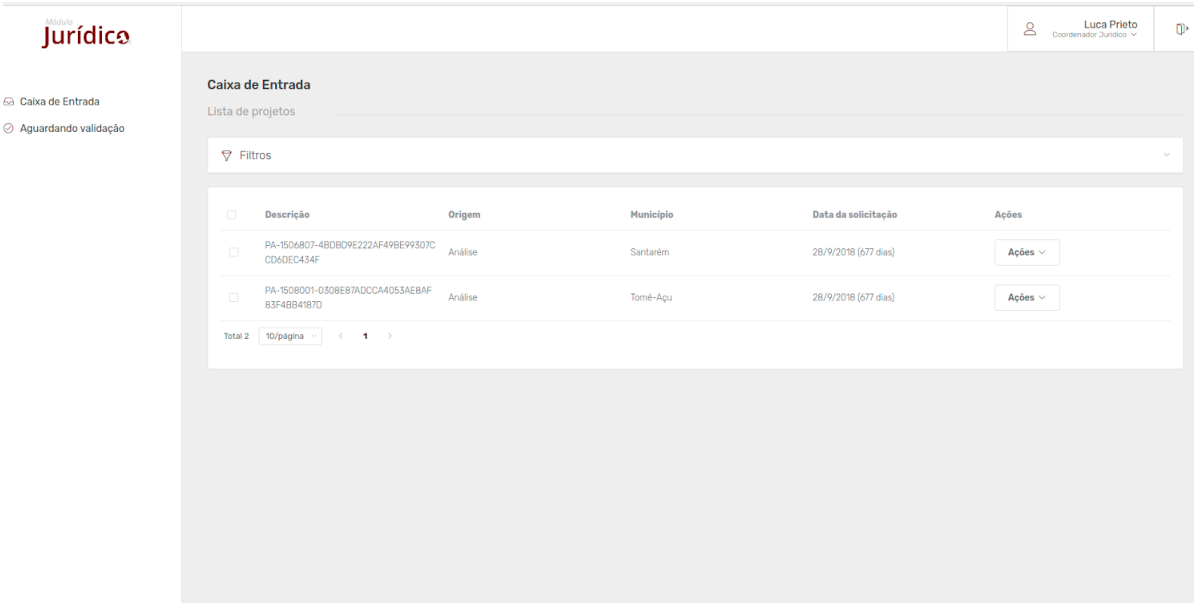
- Perfil coordenador: o perfil com mais permissões do sistema, com propósito de vincular consultores a novas solicitações, analisar os documentos recebidos no sistema e validar os pareceres dados pelos consultores;
- Perfil consultor: o usuário consultor é um perfil com função de analisar os documentos que a eles foram vinculados e emitir um parecer favorável ou não;

- Perfil assinante: um perfil com o propósito de assinar os documentos digitalmente.

Com o fluxo completo do sistema, o documento teria duas condições de saída, dependendo do parecer do consultor e coordenador os quais o analisaram nas etapas do sistema:

- Se os pareceres de consultor e coordenador fossem favoráveis e o documento fosse aprovado, o mesmo seguiria para que o assinante inserisse sua assinatura digital;
- Se os pareceres de consultor e coordenador não fossem favoráveis o documento retornaria ao seu módulo de origem com as devidas considerações e sem assinatura.

Figura 12 - Tela inicial do perfil coordenador



Fonte: LEMAF (2020)

3.2.1 Workshop

No workshop foi apresentado para todos os integrantes do *squad* a ideia do produto final. A apresentação foi feita com a demonstração do funcionamento do sistema no seu estado atual, e como o mesmo ficaria ao final da demanda, encaixando as novas funcionalidades e regras de negócio. Todo o workshop foi apresentado por um Product Owner que como parte de suas atividades, buscou captar todas as necessidades dos clientes e repassá-las da forma mais detalhada a equipe de desenvolvimento.

O problema apresentado descrevia que em certos cenários, algumas solicitações estavam ligadas a consultores e coordenadores que já não estavam mais ativos no sistema por

problemas externos. Com isso os projetos permaneciam parados dentro do módulo e assim não poderiam dar continuidade no fluxo do Projeto de Regularização Ambiental. Para solucionar este problema foi demandada a criação do perfil de Administrador com privilégios únicos, que davam permissão ao perfil de visualizar todas as solicitações existentes dentro do módulo, com opção a de alterar os seus consultores e coordenadores.

Além do workshop de apresentação da demanda a todos do time, também foi feito um outro workshop somente com os membros menos experientes. Esta outra apresentação descrevia regras específicas, fluxos e módulos externos a aplicação em questão, visando dar a todos os componentes do time, uma visão geral quanto aos módulos presentes no Projeto de Regularização Ambiental. Importante destacar que como uma grande quantidade de informação foi apresentada diretamente ao time de desenvolvimento, muitas dúvidas ficaram no ar. A ideia que os desenvolvedores tiveram do sistema final, sofreu mudanças com o decorrer do tempo.

3.2.2 Refinamento

Na etapa de refinamento, as atividades foram levantadas e subdivididas de acordo com seus impactos. Primeiramente cada regra do sistema foi separada e se tornou uma história de usuário, sendo composta assim por diversas atividades menores. A demanda se tornou muito mais complexa do que aparentava pois envolvia a quebra do fluxo padrão do sistema, o que era um problema pois alterações feitas sem planejamento dentro da aplicação poderiam gerar *bugs* em outras etapas não relacionadas a funcionalidade.

Durante essa etapa, o líder do squad o Scrum Master, em conjunto com o Product Owner, fizeram a leitura da documentação dos requisitos com os integrantes do time. As dúvidas que surgiram foram respondidas imediatamente para garantir que impedimentos do tipo não surgiriam durante a execução da Sprint. Simultaneamente a isso, um sistema Taiga[9] foi utilizado para guardar todas as atividades em forma de um *card* contendo de forma resumida, uma descrição do que era necessário ser feito na atividade e quanto ela demandaria de esforço pelos desenvolvedores.

[9] https://www.taiga.io

3.2.3 Planning

Na *planning* ou planejamento, os desenvolvedores analisaram cada uma das atividades levantadas na etapa anterior e estimaram a quantidade de horas que levariam para a conclusão das atividades. Como estudado nas disciplinas relacionadas a qualidade de software, diversas métricas podem ser usadas nessas estimativas. Geralmente opta-se por utilizar a quantidade de esforço que o time de desenvolvimento imagina que gastará para entregar uma atividade. Porém para o sistema em questão foi utilizada a estimativa de tempo em horas.

Para estimar, cada desenvolvedor utilizou de ferramentas diferentes sendo todas voltadas para o mesmo propósito. O papel do estagiário nesta etapa foi de simples observação e notou-se que as estimativas mudavam de acordo com o nível de experiência dos desenvolvedores. Se os valores escolhidos pelos desenvolvedores na estimativa fossem muito divergentes, era feita uma discussão sobre o motivo da escolha e quais pontos poderiam aumentar e diminuir o valor da estimativa final. Se um problema real fosse identificado, então a estimativa teria seu valor determinado considerando este problema, se não a estimativa considerada seria a que apresentasse o valor menor. O processo de planejamento serviu ao estagiário principalmente como forma de entender como os desenvolvedores mais experientes enxergavam as demandas. Notou-se que as partes eram tratadas unicamente como partes e que cada uma deveria funcionar independente das outras. Portanto, na estimativa era considerado somente o esforço em uma única atividade por vez.

Com a *planning* concluída, a Sprint teve seu início, o que foi estimado em aproximadamente cerca de duas semanas para o primeiro conjunto de demandas. Observou-se também que para o desenvolvimento de todas as funcionalidades do novo perfil seriam necessárias mais Sprints.

3.2.4 Reuniões diárias

Nas reuniões diárias os membros do time de desenvolvimento tinham um pequeno espaço de tempo para falarem sobre as atividades que estavam trabalhando e o que pretendiam fazer ao concluírem. Vale destacar que eram reuniões de curta duração que serviam para fazer a aproximação do time, sendo uma forma de incentivar o diálogo entre todos. Um fato de destaque é que em meio às reuniões, quando um dos membros do time informava sobre uma atividade que estava sendo impedida por algum problema, soluções

surgiam quase que de imediato. Isso auxiliava na mudança das estratégias abordadas para a continuidade das atividades que seriam realizadas durante o dia.

Em paralelo ao que foi estudado sobre este rito presente no Scrum, pode-se analisar os impactos positivos de fazer os membros do time estarem sempre alinhados quanto a todas as atividades que estavam trabalhando. Com estas informações, os integrantes sabiam quais atividades ainda iriam demandar muito esforço e quais seriam entregues com menos esforço, mesmo que não estivessem sendo executadas.

3.2.5 Revisão

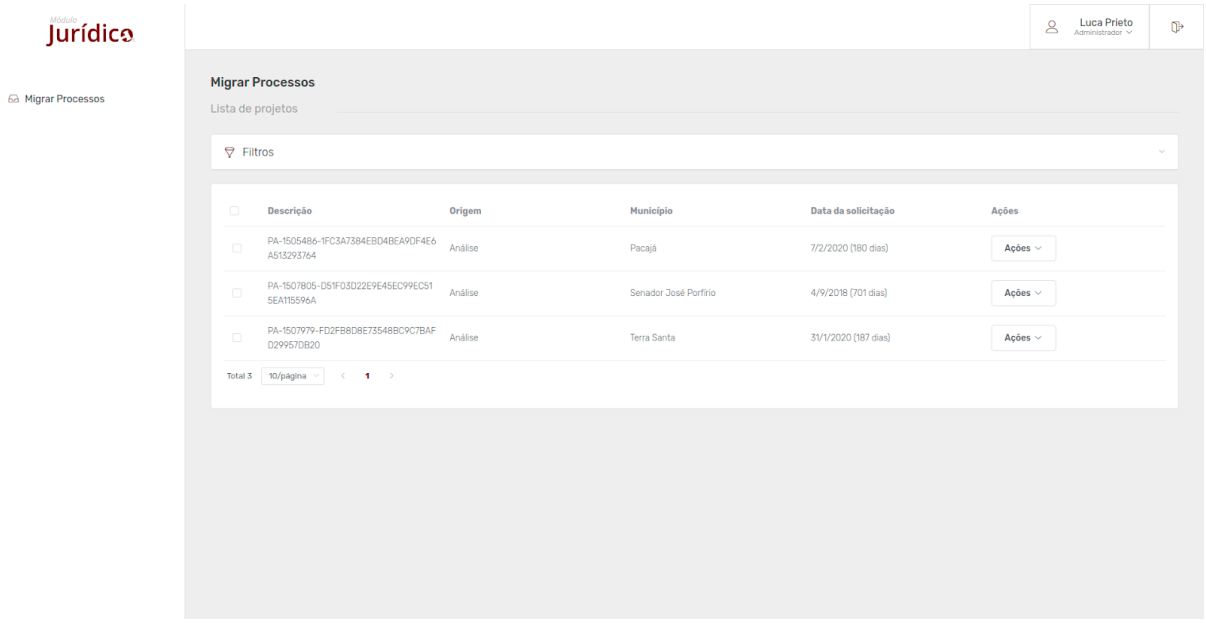
Na revisão foi apresentado ao Gerente de Projetos tudo que foi desenvolvido em comparação com o que foi demandado. Essa apresentação teve como um dos objetivos, alinhamento entre toda a equipe sobre o que foi ou não entregue em comparação ao que foi demandado. As demandas que não tinham sido feitas, passaram a ser executadas em uma nova Sprint, sendo que foram necessárias duas Sprints para sua execução.

Ao final da revisão na última Sprint, o Gerente de Projetos e o Product Owner fizeram sua apresentação ao cliente em forma de homologação sobre a demanda e retornaram com um feedback a equipe de desenvolvimento. O sistema não apresentou problemas e a demanda seguiu suas próximas etapas que foi o *deploy* no ambiente de produção juntamente com toda a documentação necessária para a entrega correta dos elementos demandados.

3.2.6 Retrospectiva

Na retrospectiva o time teve a oportunidade de levantar o que acharam de positivo, o que não foi bom e o que poderia melhorar. Nas ocasiões, vários tipos de problemas surgiram e para as próximas Sprints eles poderiam ser corrigidos assim como as melhorias que também foram adotadas para os processos da próxima Sprint. É uma reunião com duração de aproximadamente uma hora. Sua finalidade é destinada principalmente para preparar o time para novas Sprints, deixando todos cientes sobre o que ajuda, o que não ajuda, e o que pode ajudar na melhoria do processo de desenvolvimento. As Figuras 13 e 14 são exemplos das telas relacionadas a demanda em funcionamento ao final da Sprint.

Figura 13 - Exemplo da tela do administrador do sistema rodando em ambiente local



Fonte: LEMAF (2020)

Figura 14 - Exemplo da funcionalidade de migrar processos

Fonte: LEMAF (2020)

4. CONSIDERAÇÕES FINAIS

Com a conclusão do estágio, diversos aspectos chamaram a atenção do estagiário e ajudaram na formação de seus ideais, além de servirem como base para novas decisões relacionadas a sua carreira profissional e objetivos. Todo o período também serviu como um parâmetro de comparação para mensurar o quão eficientes foram todos os conhecimentos adquiridos na graduação em utilização nas atividades práticas do ambiente de profissional.

O primeiro aspecto foi a aprendizagem ganha no desenvolvimento de um sistema desde o início. Embora o Sistema de Gestão de Contratos sendo considerado um projeto de pequenas dimensões, serviu como base para consolidar os conceitos adquiridos nos treinamentos iniciais aos quais o estagiário foi submetido no início do período de estágio. Esse sistema não possuía nenhuma configuração ou base para armazenamento de dados, sendo todos criados desde o início pelo estagiário. Todos esses fatores serviram também para o entendimento do funcionamento das tecnologias utilizadas em conjunto, e quais suas respectivas funções.

As experiências em projetos práticos de algumas disciplinas serviram como fator de facilitação da adaptação do estagiário às rotinas do ambiente da organização. Essas experiências deram ao graduando uma noção do que é utilizado pelas empresas nos processos de desenvolvimento, e como isso é feito. Diversas atividades teóricas e conceitos obtidos em projetos práticos foram observados sendo utilizados na prática durante o estágio.

Os conceitos de orientação a objetos abordados nas primeiras disciplinas da graduação, ajudaram no desenvolvimento do *backend* dos sistemas, pois o manejo do Java, uma linguagem orientada a objetos, necessita de conhecimento teórico sobre esse paradigma para o entendimento do seu funcionamento. Os *frameworks* utilizados para o desenvolvimento do *backend* dos sistemas também utilizam os conceitos de orientação a objetos, e, para adaptá-los para os sistemas desenvolvidos, foi necessário utilizar também o conhecimento teórico obtido durante as disciplinas.

Os conhecimentos relacionados aos bancos de dados obtidos nas disciplinas referentes a essa matéria foram utilizados para atividades de modelagem de bancos e criação de *scripts* SQL nos projetos executados durante o estágio. A modelagem dos bancos é uma atividade que serve como base para todas as outras partes dos sistemas, e, para isso, é necessário que as estruturas desses bancos sejam criadas de forma que erros não ocorram quando o volume de

dados neles armazenados aumente. Para isso, todas as tabelas e relacionamentos dos bancos devem estar configurados conforme os conceitos de bancos de dados exigem, e o conhecimento obtido em projetos da graduação ajudou a entender essa necessidade e serviu para formular muitas estratégias para modelar a estrutura e criar os *scripts* para geração e administração dos bancos de dados.

O *framework* Scrum, estudado nas disciplinas relacionadas à engenharia de software, foi utilizado para os processos de desenvolvimento dos sistemas. As rotinas e ritos do Scrum possuem seus objetivos e quase sempre os membros do time de desenvolvimento influenciam diretamente na qualidade dos processos de desenvolvimento. Nas disciplinas da graduação relacionadas à engenharia de software, projetos práticos simularam o funcionamento Scrum nas rotinas das empresas, e, para o estagiário, esses projetos feitos nas disciplinas serviram como facilitadores para a adaptação ao Scrum durante os projetos desenvolvidos no estágio.

As tecnologias estudadas na disciplina Programação Web foram as mesmas utilizadas para os projetos executados durante o estágio, o que se fez de grande ajuda para a assimilação do conhecimento obtido durante o período de treinamento, bem como para o desempenho durante o desenvolvimento dos sistemas. O estagiário conseguiu executar suas atividades de desenvolvimento de forma mais eficiente e com mais qualidade, pois todas as tecnologias utilizadas já possuíam uma base de conhecimento formada durante a graduação.

Alguns dos conceitos estudados na disciplina Gerência de Projetos de Software foram observados, sendo aplicados pelos Gerentes de Projeto para administrarem a execução de demandas, principalmente durante as etapas de planejamento dos projetos. Diversas vezes foram observadas técnicas para contagem de pontos de função sendo utilizadas nos projetos do LEMAF, além de outras técnicas utilizadas, como o planejamento do tempo de execução das demandas, por exemplo.

Notou-se que as disciplinas que utilizaram, como metodologia de ensino, a aplicação de projetos práticos ajudaram diretamente o estagiário nas rotinas e atividades do seu estágio. Outras disciplinas da graduação poderiam, talvez, ter preparado melhor o estagiário para o mercado de trabalho, adotando, também, a mesma metodologia de ensino das disciplinas anteriormente citadas. Embora a vivência no meio profissional seja uma experiência única, passar por experiências semelhantes é importante para que o estudante tenha uma ideia do que existe no ambiente de trabalho, e, para isso, os projetos práticos poderiam ser adotados em todas as disciplinas da graduação.

A experiência de trabalho em equipe foi outro fator que serviu de grande aprendizado. A vivência com profissionais de diferentes níveis de experiência serviu como ganho de sabedoria, e foi observado como é vantajoso trabalhar em times de desenvolvimento que colaboram entre si para conclusão de um objetivo mútuo. No LEMAF, em particular, essa colaboração é incentivada por ser parte da cultura organizacional e também pelo fato de ajudar a existir agilidade para o desenvolvimento e soluções para contornar obstáculos que surgem durante os projetos.

O estágio também serviu como primeira experiência profissional ao estagiário, influenciando diretamente nas decisões tomadas para a sua carreira. Como consequência de todo o processo, uma oferta de trabalho foi proposta e aceita pelo estagiário que, com todas as experiências vividas, optou por seguir carreira na área, aceitando a oferta. Além disso, novos objetivos foram estabelecidos para o futuro, todos baseados nos pontos em que se notou ganho positivo e, também, em pontos nos quais foram notadas carências na base de conhecimento.

Por fim, pode-se concluir o quão importante é, para o graduando que se interessar por seguir carreira no mercado de trabalho, atuar como aprendiz primeiramente dentro de uma empresa. De nada vale uma extensa experiência teórica sem a prática àquele que deseja desenvolver produtos para o mercado, ainda que, para que os trabalhos práticos sejam executados de forma mais eficiente, uma base sólida de conhecimentos teóricos deva existir.

REFERÊNCIAS

BRASIL. Ministério da Justiça e Segurança Pública. **Tecnologia da informação**. Disponível em: <https://www.justica.gov.br/Acesso/institucional/tecnologia-da-informacao-1>. Acesso em: 03 ago. 2020.

BRASIL. Ministério do Meio Ambiente. **Manejo Florestal Sustentável**. Disponível em: <https://www.mma.gov.br/florestas/manejo-florestal-sustent%C3%A1vel>. Acesso em: 03 ago. 2020.

CALHAU PINTO, J. C. **YWeb - Plataforma de Criação de Templates para Marketing Online**. 2017. Dissertação (Mestrado Integrado em Engenharia Informática e Computação) – Faculdade de Engenharia da Universidade do Porto, Porto, 2017.

FATEC JACAREÍ. Geoprocessamento. **Revista Eletrônica**. Disponível em: <http://fatecjacarei.com.br/revistaeletronica/wordpress/index.php/2016/03/17/geoprocessamento/>. Acesso em: 03 ago. 2020.

GALANTE, V. R. **Metodologia Scrum para desenvolvimento de aplicativos**. Disponível em: <https://usemobile.com.br/metodologia-scrum-desenvolvimento/>. Acesso em: 03 ago. 2020.

JBOSS. **Hibernate Getting Started Guide.** 2020. Disponível em: <https://docs.jboss.org/hibernate/orm/5.4/quickstart/html_single/>. Acesso em: 03 ago. 2020.

JODAS, A. da S. **Uso do padrão MVC (Model-View-Controller) em métodos ágeis para desenvolvimento de sistemas Web por pequenas empresas**. 2012. Dissertação (Mestrado em Engenharia da Computação) – Instituto de Pesquisas Tecnológicas do Estado de São Paulo, São Paulo, 2012.

JOHNSON, R. E.; FOOTE, B. Designing reusable classes. **Journal of object-oriented programming**, v. 1, p. 22-35, 1988.

LABORATÓRIO DE ESTUDOS E PROJETOS EM MANEJO FLORESTAL (LEMAF). **Quem somos.** 2020. Disponível em: <http://www.lemaf.ufla.br/>. Acesso em: 03 ago. 2020.

MACHADO NETO, O. J. **Um framework para a construção de aplicativos de dispositivos móveis para usuários com deficiência motora decorrente de acidente vascular encefálico**. 2018. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, São Carlos, 2018.

MENDES, R. V. et al. **Desenvolvimento de uma ferramenta para organização e gerenciamento de atividades de docentes**. 2018. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Faculdade de Computação da Universidade Federal de Uberlândia, Uberlândia, 2018.

MOZILLA. **O que é JavaScript?**. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/About_JavaScript>. Acesso em: 08 ago. 2020.

MURTA, L.; WERNER, C.; BARROS, M. **Café da manhã com Java**. Disponível em: <http://www2.ic.uff.br/~leomurta/papers/murta1998.pdf>. Acesso em: 08 ago. 2020.

OGEDA, R. H. et al. **Estudo da utilização da linguagem Java no desenvolvimento de applets e aplicativos para ensino e pesquisa de engenharia química**. 1998. Dissertação (Mestrado em Engenharia Química) – Centro Tecnológico da Universidade Federal de Santa Catarina, Florianópolis, 1998.

PEREIRA, J. C. **Framework de persistência de dados com criptografia de chave assimétrica utilizando hibernate**. 2007. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau, Blumenau, 2007.

PICHETTI, J. G. B. **Sistema web para gerenciamento de bancas de trabalhos acadêmicos**. 2013. Trabalho de Conclusão de Curso (Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná, 2013.

ROCHA, L. F. da. **Aplicação da integração contínua no desenvolvimento de software**. 2014. Trabalho de Conclusão de Curso – Universidade do Sul de Santa Catarina, 2014.

RODRIGUES, J. **HTML básico - códigos HTML**. Disponível em: <https://www.devmedia.com.br/html-basico-codigos-html/16596>. Acesso em: 02 ago. 2020.

SCHMITZ, D. **Conheça o Vuetify - parte 1**. Disponível em: <https://vuejs-brasil.com.br/conheca-o-vuetify-tutorial-dicas-parte-1/>. Acesso em: 01 ago. 2020.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum-Um guia definitivo para o Scrum: As regras do jogo**. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em: 01 ago. 2020.

SPRING. **Web on Servlet Stack**. Disponível em: <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>. Acesso em: 03 ago. 2020.

TONINI, G.; SIQUEIRA, F. pgGrid: uma Implementação de Fragmentação de Dados para o PostgreSQL. **Anais do XI Workshop de Software Livre**. p. 48-53, 2010.

VUEJS. **A instância Vue.** Disponível em: <https://br.vuejs.org/v2/guide/instance.htm> Acesso em: 03 ago. 2020.

VUEJS. **Sintaxe de templates**. Disponível em: <https://br.vuejs.org/v2/guide/syntax.html> Acesso em: 03 ago. 2020.

WEISSMANN, H. L. **Vire o jogo com Spring Framework**. [S. l.]: Editora Casa do Código, 2014.