JOÃO DE SOUSA E SILVA

RELATÓRIO DE ESTÁGIO:

DESENVOLVIMENTO DE SOFTWARE NA EMPRESA GT4W - GEOTECHNOLOGY FOR WEB

LAVRAS – MG

MAIO, 2019

JOÃO DE SOUSA E SILVA

RELATÓRIO DE ESTÁGIO:
DESENVOLVIMENTO DE SOFTWARE NA EMPRESA GT4W - GEOTECHNOLOGY FOR WEB

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

Dilson Lucas Pereira
Orientador

LAVRAS – MG
MAIO, 2019

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Silva, João de Sousa e
Relatório de Estágio : Desenvolvimento de Software na empresa GT4W - Geotechnology For Web / João de Sousa e Silva. – Lavras : UFLA, Maio, 2019.
49 p. :

Relatório de Estágio (graduação)–Universidade Federal de Lavras, 2019.
Orientador: Dilson Lucas Pereira.
Bibliografia.

1. Desenvolvimento de software. 2. Java. 3. PostgreSQL. I. Pereira, Dilson Lucas.

JOÃO DE SOUSA E SILVA

RELATÓRIO DE ESTÁGIO: DESENVOLVIMENTO DE SOFTWARE NA EMPRESA GT4W - GEOTECHNOLOGY FOR WEB

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

APROVADA em 07 de Junho de 2019.

Prof. Dr. Dilson Lucas Pereira	UFLA
Prof. Dr. Neumar Costa Malheiros	UFLA
BSI Gabriela Enes Campos	GT4W

Dilson Lucas Pereira
Orientador

LAVRAS – MG
Maio, 2019

Dedico este trabalho à minha família, meu alicerce, que tanto se doou em prol desta conquista.

AGRADECIMENTOS

Agradeço ao apoio dos meus amigos e familiares, sobretudo de meu pai, minha tia e minha irmã. Sem eles, eu jamais chegaria onde cheguei.

RESUMO

O presente trabalho tem como objetivo relatar as atividades que foram desenvolvidas durante o período de estágio na empresa GT4W - Geotechnology For Web. A empresa em questão está atualmente sediada na cidade de Lavras - MG e atua em várias partes do Brasil desenvolvendo soluções em software de geoprocessamento para seus clientes. Durante o estágio foram desenvolvidos os módulos de um sistema para gestão e controle de concessões florestais utilizando as tecnologias Java, JavaScript, PostgreSQL, HTML e CSS e os frameworks Play Framework e AngularJS. Todo o processo foi realizado utilizando a metodologia de desenvolvimento ágil SCRUM e o padrão de código MVC (Model, View, Controller).

Palavras-chave: Estágio, Java, JavaScript, HTML, Geoprocessamento, Play Framework, MVC

ABSTRACT

The following work aims to report the activities that were developed during an internship period at the company GT4W - Geotechnology For Web. The company in question is currently based in Lavras - MG and operates over several parts of Brazil developing geoprocessing software solutions to its clients. During the internship the modules of a system for managing and controlling forest concessions were developed using the Java, JavaScript, PostgreSQL, HTML and CSS technologies and the frameworks Play Framework and AngularJS. The entire process was realized using the SCRUM methodology for fast development and the MVC code pattern (Model, View, Controller).

Keywords: Internship, Java, JavaScript, HTML, Geoprocessing, Play Framework, MVC

SUMÁRIO

1 INTRODUÇÃO

A GT4W Consultoria e Serviços Em Geoprocessamento LTDA é uma empresa fundada em 2011, que atualmente tem sede na cidade de Lavras, no estado de Minas Gerais. A empresa, presentemente, atua em projetos de cunho ambiental, tanto em conjunto com o LEMAF - Laboratório de Estudos e Projetos em Manejo Florestal da Universidade Federal de Lavras, quanto de forma individual, desenvolvendo softwares de geoprocessamento e oferecendo suporte aos mesmos. Com um time de aproximadamente 90 colaborares, a empresa tem um perfil jovem, comprometido e encontra-se em constante expansão através de novos projetos e contratações.

Durante o estágio na empresa, foram desenvolvidos os módulos de um sistema de gerenciamento de concessões de florestas públicas para exploração no estado do Pará. No início do projeto e antes do desenvolvimento das atividades, foi oferecido um treinamento de três meses, no qual foi possível aprender e se familiarizar com os frameworks e linguagens de programação utilizados.

O sistema foi desenvolvido utilizando as linguagens Java, JavaScript, HTML, CSS e os frameworks Play Framework e AngularJS, e o banco de dados PostgreSQL. Ele oferece suporte para as edições mais recentes dos navegadores Google Chrome e Mozilla Firefox, em suas versões para computadores com sistemas operacionais Windows e Linux. Para a organização do código foi utilizado o padrão MVC (Model, View, Controller).

1.1 Atividades desenvolvidas

Os módulos desenvolvidos para tornar possível o funcionamento básico do sistema de gestão de concessões florestais são:

- Gestão de Florestas Públicas,
- Gestão de Licitações,

- Gestão de Concessionários,
- Gestão de Contratos,
- Monitoramento de Contratos,
- Central de Comunicação,
- Central do Administrador,
- Gestão de Contratos (perfil Concessionário).

A fim de ofercer uma visão holística do sistema, as funcionalidades gerais de cada módulo serão brevemente explanadas a seguir. Posteriormente, o módulo de Gestão de Contratos será abordado de forma separada e mais minuciosa, pois nele se encontra o cerne do sistema, visto que este módulo serve como porta de entrada para a execução das principais atividades do fluxo de processo do cliente. A saber, o sistema conta com sete módulos: Gestão de Florestas Públicas, Gestão de Licitações, Gestão de Concessionários, Gestão de Contratos, Central de Comunicação e Central do Administrador. Dois perfis distintos possuem acesso ao sistema (cliente e concessionário). Dentre os módulos, dois são acessíveis para ambos os perfis: Gestão de Contratos e Central de Comunicação.

1.1.1 Gestão de Florestas Públicas

Este módulo permite a visualização, cadastro, edição e pesquisa das florestas públicas e suas Unidades de Manejo Florestal, solicitando a inserção de shapes (arquivos contendo a geometria da área) para georreferenciá-las. Também calcula e valida as áreas dos shapes inseridos e garante, através do processamento das geometrias informadas, que todas as unidades de manejo florestal de uma determinada floresta estejam obrigatoriamente contidas em seu perímetro. Além disso, permite que as espécies contidas nas florestas sejam designadas a diferentes categorias de exploração. As categorias são compostas por um grupo de espécies de árvores

com características semelhantes, enumeradas de 1 a 5. Esta informação é utilizada posteriormente no sistema para calcular os valores cobrados em contratos cujos preços são definidos com base nessas categorias.

1.1.2 Gestão de Licitações

Este módulo permite o cadastro e a visualização das licitações para exploração florestal. O cadastro da licitação foi dividido em vários passos de modo a facilitar o preenchimento correto dos dados e prover sua melhor organização. A licitação deve conter uma floresta pública cadastrada previamente no módulo Gestão de Florestas Públicas, e uma ou mais unidades de manejo florestal pertencentes àquela floresta. A licitação define ainda qual critério será utilizado para efetuar as cobranças do contrato, que podem ser feitas por categoria de exploração - que agrupam espécies de árvores semelhantes - ou por preço único. No primeiro caso, todas as espécies de uma categoria têm o mesmo preço, que normalmente varia para cada categoria diferente. No último caso, em que a cobrança tem por base um preço único, todas as categorias de exploração da floresta - e, por consequência, todas as espécies da mesma - têm um único preço. Neste módulo o sistema valida o cadastro das licitações de acordo com um conjunto de regras de negócio definidas pelo cliente e estima os preços do edital com base nos valores financeiros informados.

1.1.3 Gestão de Concessionários

Este módulo permite visualizar, cadastrar, editar e ativar/desativar os concessionários, que são pessoas jurídicas que participam dos processos licitatórios e eventualmente recebem concessões para exploração das unidades de manejo florestal. O módulo deve manter os dados dos diferentes responsáveis de cada concessionário e garantir que apenas determinados tipos de responsáveis tenham acesso ao sistema: os responsáveis legais e procuradores. Parte dos dados cadastrais dos

concessionários é enviada a um sistema de integração que serve como base de dados comum e portal de segurança para outros sistemas, inclusive para o Sistema de Gestão de Concessões (SGC). Os concessionários e responsáveis com acesso ao sistema o fazem através de um usuário com um perfil exclusivo, o Perfil Concessionário, que tem acesso limitado às funcionalidades do sistema. Este perfil tem acesso a apenas dois módulos, sendo eles a Central de Comunicação e a Gestão de Contratos, sendo que este último apresenta funcionalidades específicas ao perfil Concessionário, que serão explanadas posteriormente.

1.1.4 Gestão de Contratos

Módulo responsável pelo cadastro, edição, rescisão e gestão dos contratos. Nele, é possível gerenciar apostilamentos, que são ajustes periódicos nos preços dos contratos publicados no Diário Oficial do Estado. Este módulo também é responsável pela gestão das bonificações, que são descontos percentuais concedidos aos contratos cujos concessionários atendem a determinados critérios definidos em licitação.

1.1.5 Monitoramento de Contratos

Neste módulo, é possível monitorar o que é abatido e produzido nas unidades de manejo florestal, com base nos dados provenientes dos inventários de produção florestal, que são fornecidos pelos concessionários ou seus responsáveis. O sistema deve ser capaz de ler e processar os dados do inventário em uma planilha e, num momento posterior, identificar e reportar ao usuário possíveis irregularidades nos mesmos.

1.1.6 Central de Comunicação

Este módulo é responsável por intermediar a comunicação entre o cliente e seus concessionários através de uma central de mensagens semelhante à uma

caixa de entrada de correio eletrônico. À cada mensagem enviada para os concessionários, é disparado um alerta via e-mail para os responsáveis cadastrados para que eles possam acessar o sistema e respondê-la. As mensagens podem ter um prazo de resposta definido no momento do envio. Nesses casos, o sistema calcula os prazos e mantém o controle de quais respostas não foram enviadas a tempo.

1.1.7 Central do Administrador

Neste módulo, o cliente, que terá direitos de administrador do sistema, poderá alterar algumas funcionalidades padrão e configurar alertas, que são mensagens automáticas a serem enviadas aos concessionários periodicamente, via Central de Comunicação.

1.1.8 Gestão de Contratos (Concessionário)

Este módulo, de acesso exclusivo aos concessionários cadastrados no sistema com contratos ativos, permite o envio de planilhas contendo o inventário de produção florestal. O inventário é uma relação que detalha tudo o que foi produzido na unidade de manejo florestal em um determinado período. Estes dados serão então processados e pré-validados pelo sistema e revisados pelo cliente.

1.2 Módulo de Gestão de Contratos

Uma vez que tenham sido cadastradas as florestas públicas, suas respectivas unidades de manejo florestal e as licitações, o usuário consegue então cadastrar os contratos, que constituem a base sobre a qual se constroem as demais funcionalidades do sistema. A seguir estão listadas as funcionalidades oferecidas pelo módulo de gestão de contratos na ordem em que aparecem no sistema - ordem esta que obedece ao fluxo de processo de negócio do cliente.

1.2.1 Cadastro e Edição de Contrato

Sendo o ponto de partida para este módulo do sistema, a tela de Cadastro de Contrato permite a inserção de informações referentes ao contrato: Número da licitação, UMF, CNPJ do concessionário. Estas informações são derivadas de dados já previamente cadastrados em outros módulos do sistema e são campos de preenchimento obrigatório. Os demais campos não são diretamente dependentes de dados recuperados de outros módulos, mas também são obrigatórios para o cadastro do contrato. São eles: número do processo, data de assinatura do contrato, data de publicação do Diário Oficial do Estado (DOE), data de início da vigência e data de fim da vigência. Destes campos, apenas as datas podem ser editadas após a conclusão do cadastro do contrato.

1.2.2 Rescisão de Contrato

Neste ponto é possível rescindir um contrato, que esteja em situação vigente, antes do término do mesmo, previsto pela data fim de vigência. Para tanto, é necessário informar o número do processo administrativo, a data de publicação do Diário Oficial do Estado (DOE) e inserir o anexo referente à publicação do DOE. Após o preenchimento e confirmação destas informações, o contrato será rescindido e deixará de ser listado nos demais módulos do sistema.

1.2.3 Gerenciar Apostilamentos

Este submódulo do sistema permite o cadastro e a visualização de apostilamentos, que são ajustes períodicos feitos no preço do contrato. Para cadastrar um apostilamento é obrigatório informar a data de início de vigência, o índice de reajuste e o início e fim do período do índice acumulado (mês e ano). Com base nestas informações, o sistema estima qual será o valor corrigido do contrato, respeitando o tipo de preço do mesmo (que pode ser por categoria de exploração ou por preço único) para efetuar os cálculos. Neste submódulo também é possível

monitorar a variação percentual no valor do contrato e a diferença, em reais, com relação ao valor incial e anterior aos apostilamentos.

1.2.4 Gerenciar bonificações

Submódulo no qual é possível cadastrar e visualizar as bonificações, bem como acompanhar as atualizações no preço do contrato a cada dia. Uma bonificação é um desconto temporário no valor do contrato oferecido a um concessionário que atinja determinados pré-requisitos previstos na licitação, chamados de indicadores. Tais indicadores são métricas que auxiliam no cumprimento da exploração sustentável, como aproveitamento de biomassa, geração de empregos locais pela concessão florestal, recuperação de áreas degradadas, dentre outros. Para o cadastro de uma bonificação é necessário informar o percentual de bonificação, o indicador de referência e as datas inicial e final de aplicação. Ao contrário dos apostilamentos, nos quais apenas um deles pode estar vigente por período, é possível que haja várias bonificações ativas e sobrepostas em um determinado momento, desde que possuam indicadores diferentes. Isto permite que, em datas distintas, o mesmo contrato apresente variação no valor independente da existência de um ou mais apostilamentos.

Ao preencher todos os dados necessários, o sistema realiza uma simulação e exibe a relação dos preços mínimos do edital, valores ofertados pelo concessionário no contrato, valores corrigidos por apostilamento e o preço bonificado. Em alguns casos, quando o preço bonificado for inferior ao mínimo previsto em edital, o valor do edital é então considerado em detrimento do primeiro.

1.3 Processo

Para a execução deste projeto, bem como as de todos os demais projetos da empresa, o processo utilizado é o de desenvolvimento ágil SCRUM. O desenvolvimento ágil tem por objetivo diminuir a burocracia e as documentações excessivas

para focar em entregas pequenas, rápidas e incrementais que gerem valor para o cliente. O SCRUM, que é uma das metodologias de desenvolvimento ágil, tem equipes autogerenciáveis que possuem grande autonomia para decidir a melhor forma de desenvolver o produto (neste caso, o software) dentro do escopo do projeto.

Seguindo o SCRUM, para entregar determinada função ou módulo do sistema, as tarefas necessárias são transformadas em histórias, que normalmente englobam uma funcionalidade (ex.: cadastro de floresta pública, edição dos dados da floresta). A história então é dividida em atividades menores que podem ser feitas de forma rápida por cada membro (criar tela de cadastro, criar classe Java, implementar método de salvar). Tendo conhecimento de todas as histórias e suas atividades, a equipe se reúne para planejar quais delas serão realizadas durante a próxima Sprint. A Sprint dura de 10 a 15 dias úteis e engloba um conjunto de histórias que a equipe deve entregar ao seu final. Histórias cujas atividades não foram finalizadas não são consideradas entregues e podem aparecer novamente em Sprints posteriores.

A cada dia de desenvolvimento, a equipe se reúne durante, no máximo, 15 minutos para discutir o andamento da Sprint e os possíveis impedimentos. Cada membro deve reportar as atividades que realizou desde a última reunião diária e quais pretende realizar até a próxima reunião, e informar caso algo o esteja impedindo de realizar suas atividades. As atividades, bem como seu estado atual, são mantidas em um quadro (no formato físico ou digital) para que toda a equipe possa visualizá-las rapidamente. Ao final da Sprint, a equipe se reúne novamente para discutir seus pontos positivos e negativos e apresentar o que foi desenvolvido ao cliente (ou a um representante). O cliente pode aceitar o que foi feito ou solicitar alterações, e então é feita uma nova reunião de planejamento para a próxima Sprint.

Além do SCRUM, alguns processos são seguidos para o desenvolvimento do software durante o estágio. Os códigos (nomes de classes, métodos e variáveis) devem seguir um padrão pré-estabelecido pela equipe e obedecer ao modelo MVC (Model-View-Controller). As histórias da Sprint só serão disponibilizadas para serem testadas quando todas as atividades estiverem finalizadas e após a realização de uma revisão de código por parte de outro desenvolvedor, preferencialmente algum que não tenha participado diretamente do desenvolvimento daquela história.

2 REFERENCIAL TEÓRICO

Neste capítulo, serão abordados os conceitos e as tecnologias mais importantes na empresa e no projeto. O primeiro conceito a ser discutido, a Arquitetura MVC, tem grande importância pois sua aplicação pode ser percebida em diferentes áreas do sistema. Adiante, a metodologia de desenvolvimento ágil SCRUM, que foi utilizada no projeto, será abordada. Em seguida, serão discutidas as tecnologias empregadas, a começar pelo AngularJS, seguido pelo Java e, por fim, o PostgreSQL.

2.1 Arquitetura MVC

A utilidade de qualquer software é inerente à sua habilidade de interagir com algo ou alguém. Na maioria das vezes, softwares interagem com pessoas e, portanto, necessitam de interfaces que permitam tais interações. Por sua vez, o desenvolvimento de software moderno parte da premissa de que interfaces mudam com o tempo. As interfaces gráficas são particularmente sujeitas a alterações. Ao passo em que surgem novos dispositivos, com formatos e métodos de entrada distintos, e novos padrões de design, este tipo de interface sofre mudanças para acompanhar as novas tendências.

Por outro lado, a aplicação presente por trás da interface pode ser relativamente constante, conforme afirma (DEACON, 2005). Uma aplicação bancária, por exemplo, que costumava utilizar interfaces em linha de comando ou menus baseados em caracteres, é provavelmente a mesma aplicação que hoje é executada por trás de uma interface gráfica ou um aplicativo móvel. Neste sentido, (DEACON, 2005) constata que construir a interface dentro da própria aplicação seria algo prejudicial a ambas, tornando a aplicação menos flexível e mais difícil de migrar e dificultando o uso da interface por outras aplicações.

Para lidar com essa questão, uma arquitetura foi definida ainda na década de 70 pelo SmallTalk, denominada Model-View-Controller (MVC), segundo a

qual a aplicação é dividida em pelo menos três partes, de modo a separar a parte lógica do problema de sua interface. São elas:

2.1.1 Model

(DEACON, 2005) define Model como a essência imutável da aplicação/domínio. Segundo ele, no paradigma das linguagens de programação orientadas a objetos, a model compreende o conjunto de classes que modelam e dão suporte ao problema. Estas classes tendem a ser estáveis, sofrendo poucas alterações, e existem por tanto tempo quanto o próprio problema. Alguns exemplos de model seriam: Cliente, Contrato, Licitacao, Usuario.

Existe ainda uma subdivisão do conceito de model dentro da arquitetura MVC, compreendendo a domain model e a application model. (DEACON, 2005) explana que a primeira delas seria, de fato, composta pelos objetos representativos da essência do problema e nada mais. A segunda, por sua vez, abrange classes que sabem da existência das Views e provêm alguma forma de atualizá-las com as informações da domain model. No entanto, este nível maior de abstração não é considerado no processo de desenvolvimento de software da GT4W, portanto estas diferenciações não serão discutidas em maiores detalhes.

2.1.2 View

As Views são classes que permitem visualizar suas models. (DEACON, 2005) exemplifica que as Views nos fornecem janelas, muitas vezes ”literais”(referindo-se, aqui, às janelas dos sistemas operacionais), para a model. As views têm um forte cunho gráfico devido à sua natureza de permitir a visualização de certos dados da model, mas não necessariamente o são. Elas somente sabem que a model existe e que possui alguns dados a serem exibidos.

2.1.3 Controller

Segundo o paradigma original, proposto pelo SmallTalk, o papel da Controller é permitir a manipulação da View. Essas classes lidam com a entrada de dados, ao passo em que as Views lidam com a saída dos mesmos. No âmbito das aplicações desenvolvidas na GT4W, a Controller assume um segundo papel: validar as entradas de dados para garantir que a model não receba variáveis com dados vazios ou tipos inválidos.

2.1.4 Comunicação entre Model, View e Controller

A comunicação da View para a Model se dá através de eventos. Um clique em um botão dispara uma mensagem para que o objeto da model realize uma ação; um valor digitado em um campo de entrada pode também enviar uma mensagem para que o objeto correspondente da model tenha seu valor atualizado. Tais mensagens são disparadas quando estes eventos ocorrem.

Neste ponto, observa-se outra diferença entre o padrão proposto pelo SmallTalk e o que é de fato praticado no processo de desenvolvimento na empresa Geotechnology For Web. No modelo mais atualizado do padrão MVC, chamado de MdMaVC, é papel da application model enviar as mensagens apropriadas para os objetos da domain model, que então respondem à requisição (DEACON, 2005). No processo aplicado na GT4W, a Controller fica responsável por receber a mensagem do evento. Tendo recebido-a, a Controller valida os dados e então envia as mensagens apropriadas aos objetos das models, que executam as operações necessárias e respondem para as Controllers com os valores resultantes das operações executadas. Fica então a cargo da Controller atualizar a View com os novos valores.

2.2 Metodologia SCRUM

O desenvolvimento de software enfrenta desafios como seu custo alto, que por vezes extrapola os valores inicialmente acordados com o cliente, entregas incompletas ou fora dos prazos definidos e a baixa qualidade do software final. Tendo em vista estes problemas, (SILVA, 2009) afirma que foram estabelecidas normas e modelos de qualidade para inserir qualidade no processo de produção de software, destacando-se as normas da série ISO 9000. Neste contexto, as metodologias oferecem descrições de como atingir os padrões propostos pelas normas e modelos de qualidade. Segundo (SILVA, 2009), as metodologias fornecem guias para a definição das atividades que serão feitas, além de critérios para seu monitoramento. Também definem quando os artefatos devem ser desenvolvidos e quais serão eles.

O SCRUM é uma das metodologias ágeis de desenvolvimento de software que prevê maior foco em pessoas, entregas que agreguem valor ao cliente e respostas a mudanças. Isto não significa que no SCRUM não há documentações ou planejamentos, mas que eles apenas têm menor prioridade para esta metodologia. De acordo com (SILVA, 2009), alguns dos artefatos previstos pelo SCRUM são o Product Backlog e o Sprint Backlog. O primeiro trata-se de um documento sujeito a mudanças que é definido ao início do projeto pelo cliente, contendo as características esperadas do software em ordem de prioridade. Por sua vez, o Sprint Backlog é um documento que reúne as partes do Product Backlog que serão implementadas pelo time SCRUM.

(SILVA, 2009) afirma que os processos do SCRUM são divididos em várias iterações chamadas de Sprint, que têm um tempo fixo, normalmente de trinta dias, no qual o time trabalha para atingir o que foi especificado para a iteração. Ao término da Sprint, o time apresenta o resultado do desenvolvimento naquele período. Ainda de acordo com (SILVA, 2009), é realizada uma reunião de planejamento de Sprint, chamada de Sprint Planning, no primeiro dia da iteração. Nela,

o Scrum Master, que é o líder do processo, define em conjunto com o cliente quais serão as funcionalidades contempladas pelo Product Backlog. Diariamente são feitas as reuniões diárias, que devem durar até quinze minutos, para que os membros do time avaliem o que foi feito por cada um no dia anterior e no dia atual, e se foram encontrados impedimentos.

Ao término da Sprint é feita a Sprint Review, que, segundo (SILVA, 2009), trata-se de uma reunião com o cliente que poderá dar novo direcionamento ao projeto. Após esta reunião é feita uma última reunião, a Sprint Retrospective. (SILVA, 2009) afirma que nesta reunião avalia-se os aprendizados e ajustes necessários visando sempre melhorar o processo. Em seguida, inicia-se uma nova iteração, ou Sprint, com uma nova reunião de planejamento, e assim sucessivamente. (SILVA, 2009) destaca ainda que o SCRUM só considera uma tarefa como pronta se a mesma estiver apta a entrar em produção quando o cliente decidir, e que a equipe SCRUM deve entregar ao cliente algo funcional ao final de cada Sprint, e não apenas documentações.

2.3 AngularJS

AngularJS é um framework JavaScript cujo foco é a simplicidade. Ao contrário de muitos de seus concorrentes, que são resultado de esforços da comunidade de código aberto, o AngularJS foi desenvolvido e é mantido por engenheiros altamente capacitados da Google, provendo, assim, ao desenvolvedor que faz uso desta ferramenta uma equipe de suporte especializado e dedicado para sanar eventuais dúvidas com relação ao Angular. Este framework permite que o desenvolvedor deixe de lado aspectos mais maçantes da programação, como ter de atualizar manualmente as views, e foque na lógica da aplicação.

Algumas das vantagens fortes do AngularJS destacados por (BHANSALI, 2014) são:

- Integração com aplicações existentes,

- Simplicidade,
- Extensibilidade.

2.3.1 Integração com aplicações existentes

É fácil adicionar o AngularJS a aplicações já existentes, pois o framework só começa a avaliar a página após o término do processo de carregamento da mesma. Isto pode indicar que aplicações já existentes não precisam sofrer alterações para que tenham adicionadas a elas funcionalidades do Angular, visto que este só entra na etapa final do carregamento.

2.3.2 Simplicidade

O AngularJS utiliza objetos POJO (Plain Old JavaScript Objects), que os Desenvolvedores de JavaScript já estão habituados, ao contrário de outros frameworks, que requerem a criação de objetos próprios com métodos para recuperar e definir seus valores (Getters e Setters). Além disso, (BHANSALI, 2014) destaca que é possível trabalhar com o Angular em um documento HTML básico proveniente do sistema de arquivos local, não sendo necessário utilizar nenhum servidor Web. Sendo assim, é fácil criar novas aplicações rapidamente.

2.3.3 Extensibilidade

Segundo (BHANSALI, 2014), AngularJS utiliza Diretivas, o que permite criar elementos customizados e atributos que estendem o vocabulário HTML padrão. Ele exemplifica demonstrando que é possível definir o modo como uma tag HTML customizada é renderizada pelo navegador e então atribuir comportamentos específicos a ela, possibilitando a criação de uma biblioteca própria de elementos reutilizáveis.

Ademais, (BHANSALI, 2014) ressalta algumas funcionalidades do framework criado pela Google, dentre elas a ligação de dados bidirecional (atualizações

nos dados da model são automaticamente refletidas no documento HTML e vice-versa); a injeção de dependências, que facilita o desenvolvimento e entendimento da aplicação; e a utilização do MVC, apesar do fato de que o modelo tradicional não é seguido à risca. O AngularJS apresenta-se como uma ferramenta altamente poderosa que agiliza o processo de desenvolvimento em projetos de larga escala.

2.4 Java

Para desenvolvimento da estrutura back-end do projeto, a linguagem utilizada é o Java. Esta linguagem orientada a objetos é comumente utilizada para desenvolvimento de aplicações Web por possuir características que permitam a criação de aplicações distribuídas, escaláveis, seguras e de baixo custo de manutenção (MARQUES, 2011). Por sua alta popularidade, Java conta com uma enorme gama de bibliotecas, muitas das quais são gratuitas, que facilitam a manipulação de arquivos, comunicação de rede, envio de e-mails, integração com bancos de dados, criptografia de dados e muitas outras funcionalidades.

2.4.1 Java Persistence API (JPA)

A JPA, ou Java Persistence API, é a especificação da linguagem para que seja feito o mapeamento objeto-relacional (transformação de objetos Java em relações do banco de dados) e gerenciar sua persistência. O JPA permite que os dados dos objetos sejam persistidos no banco de dados de forma implícita, o que simplifica muito o processo de desenvolvimento.

2.5 PostgreSQL

O PostgreSQL é um Sistema Gerenciador de Banco de Dados Relacional criado, inicialmente, na Universidade de Berkeley, Califórnia com o nome de POSTGRES. Trata-se de uma ferramenta gratuita, que opera sob a licença BSD,

o que permite que ele seja utilizado para fins comerciais. Suas características são: SGBD relacional com suporte a ACID (transações), Replicação, Cluster (alta disponibilidade), Multithreads, Segurança SSL e criptografia, SQL, Incorporável em aplicações gratuitamente e Capacidade de armazenamento (MILANI, 2008).

Com relação ao fato de o PostgreSQL ser um SGBD relacional com suporte a ACID (transações), (MILANI, 2008) afirma que o mesmo é um SGBD relacional completo que oferece suporte a operações que atendem aos requisitos de Atomicidade, Consistência, Isolamento e Durabilidade (ACID), além de garantir a integridade referencial.

Acerca da propriedade de replicação, (MILANI, 2008) aponta que o PostgreSQL disponibiliza os recursos que são necessários para que seja feita a replicação de dados entre diferentes servidores. Uma vantagem apontada por ele faz referência à sua licença, a BSD, que permite a utilização em aplicações comerciais.

A característica de cluster (alta disponibilidade) é explanada pela constatação de (MILANI, 2008) de que é possível configurar o PostgreSQL como um cluster de informações, para aumentar sua capacidade em múltiplos servidores, conectados e sincronizados entre si de modo a atender múltiplas requisições.

A propriedade de multithreads se dá pelo fato de que o PostgreSQL alça mão do recurso de multithread dos sistemas operacionais para gerenciar várias conexões com o banco de dados de uma única vez. Dessa forma, ele possibilita que mais de uma pessoa acesse a mesma informação no banco de dados sem que haja filas ou atrasos. No caso de múltiplas requisições contendo alguma requisição que seja de gravação, o PostgreSQL força filas de acesso para garantir a integridade dos dados (MILANI, 2008).

Com relação à segurança SSL e criptografia, o PostgreSQL conta com suporte nativo ao SSL, que é um protocolo de segurança que permite o estabelecimento de conexões seguras para o tráfego dados sensíveis, além disso, a ferramenta oferece extensibilidade para uso de algoritmos de criptografia.

Acerca do SQL, o PostgreSQL está em conformidade com os padrões estabelecidos pelo ANSI SQL e é continuamente atualizado para atender às novas versões do padrão.

A propriedade que permite que o PostgreSQL seja incorporável em aplicações gratuitamente se dá graças à utilização da licença de uso BSD, permitindo a utilização do SGBD gratuitamente para aplicações com fins comerciais, ao contrário de alguns de seus concorrentes. Isto o torna uma excelente alternativa para micro e pequenas empresas.

A capacidade de armazenamento do PostgreSQL é citada por (MILANI, 2008) como uma de suas características de que lhe conferem vantagem, afirmando que o SGBD consegue armazenar grandes volumes de informação em suas tabelas de modo eficiente e confiável, superando até mesmo o tamanho máximo de arquivos suportado por alguns sistemas operacionais. (MILANI, 2008) ressalta que o tamanho máximo de um banco de dados do PostgreSQL é ilimitado, e o máximo de uma tabela é de 32 Terabytes.

3 VISÃO GERAL DAS ATIVIDADES

Esta seção detalha os procedimentos utilizados no decorrer do desenvolvimento do projeto.

3.1 Procedimentos Metodológicos

Este trabalho foi realizado no período de Agosto de 2018 a Maio de 2019, objetivando a documentação da experiência do discente no desenvolvimento de software do Sistema de Gestão de Concessões, na empresa GT4W Consultoria e Serviços em Geoprocessamento. Durante o processo de desenvolvimento do sistema, no qual o discente atuou como desenvolvedor, foi possível colocar em prática conhecimentos obtidos em sala de aula durante toda a a graduação, bem como adquirir novos saberes sobre ferramentas, frameworks que auxiliam e agilizam o desenvolvimento de software, e vivências no ambiente empresarial.

Com base nas informações e experiências adquiridas foi possível analisar o estágio como um todo através da relação entre as disciplinas que mais auxiliaram no processo, os conhecimentos obtidos durante a graduação, os saberes assimilados no contexto do estágio e as atividades desenvolvidas.

3.2 Desenvolvimento de Software

O desenvolvimento de software é um processo que engloba tanto a programação quanto os testes do sistema, que são realizados após o término da etapa de programação. Quaisquer erros encontrados durante os testes devem ser corrigidos, somente assim considera-se finalizado o desenvolvimento de algum módulo ou funcionalidade do sistema.

A priorização das funcionalidades e módulos a serem desenvolvidos e o comportamento do sistema são pré-definidos, majoritariamente, pelo Product Owner e, porventura, pelo próprio cliente.

Tendo recebido a nova funcionalidade a ser elaborada, os Desenvolvedores da equipe debatem conjuntamente, e, por vezes, com o auxílio do Analista de Banco de Dados, qual a melhor forma de implementar o software para contemplá-la. Neste processo, discutem-se quais classes, tabelas do banco de dados e métodos serão necessários para garantir que o sistema execute as novas operações sem comprometer o funcionamento de outras pré-existentes. Só então a etapa programação se inicia efetivamente. Ao término da programação os Desenvolvedores executam testes simples para garantir o funcionamento básico do sistema antes de disponibilizá-lo para o Analista de Qualidade de Software.

3.3 Análise de Desempenho

Com base nas experiências adquiridas ao longo do período de estágio na empresa, é possível realizar uma análise geral do desempenho atingido a partir dos pontos relacionados a seguir.

3.3.1 Conhecimentos Adquiridos

Um dos grandes conhecimentos adquiridos foi a integração de diferentes tecnologias e linguagens de programação em um único sistema, visto que, durante a graduação, por vezes as tecnologias foram estudadas de forma separada e a forma como elas se ligam nem sempre ficou clara. Por exemplo, o aprendizado da integração entre as classes Java com as tabelas do banco de dados através do Java Persistence API (JPA).

O conhecimento e utilização de frameworks para facilitar e agilizar o desenvolvimento de software também foi um saber assimilado de grande importância, visto que tais ferramentas são amplamente utilizadas no mercado de trabalho, mas pouco abordadas durante a graduação.

Por fim, o contato direto com desenvolvedores de software mais experientes configurou-se como um constante meio de aprendizado e aperfeiçoamento

das melhores práticas de programação. Por meio das revisões de código que eram constantemente feitas ao término do desenvolvimento de cada atividade, foi possível aprender mais sobre reutilização de código, mapeamento de entidades e uso de padrões.

3.3.2 Pontos Positivos e Negativos

Dentre os pontos positivos, foi possível observar, por parte do discente, uma maior facilidade em disciplinas com base de programação e/ou relacionadas ao estágio, bem como o ganho de experiência da dinâmica do mercado de trabalho, que não seria possível somente com a graduação. Além disso, o período de estágio proporcionou incomparáveis crescimentos no âmbito profissional e pessoal.

Em contrapartida, o tempo do discente dedicado aos estudos foi, a princípio, impactado negativamente, visto que este passou a ser dividido entre a graduação e o estágio. No entanto, tal experiência serviu de aprendizado para uma melhor organização do tempo e das atividades.

3.3.3 Disciplinas que Auxiliaram no Estágio

Devido ao escopo do estágio, que abrangeu o desenvolvimento de software, algumas disciplinas vistas durante o curso foram imprescindíveis para a execução das atividades, sobretudo aquelas diretamente ligadas à programação e às tecnologias utilizadas. São elas: Algoritmos e Estruturas de Dados (I, II e III), Banco de Dados (I e II), Programação Orientada a Objetos, Arquitetura de Software, Redes de Computadores e Sistemas Distribuídos. Ainda é possível destacar disciplinas que auxiliaram no estágio por meio do estudo de conceitos importantes para o desenvolvimento de software, como Gestão da Qualidade de Software, Projeto e Análise de Algoritmos, Interação Humano-Computador e Segurança, Auditoria e Avaliação de Sistemas de Informação.

3.4 Tarefas Executadas

No processo de desenvolvimento de software na empresa, todos os desenvolvedores atuam em duas frentes de desenvolvimento, conhecidas como front-end (a parte visível da aplicação) e back-end (a parte da aplicação responsável pelo processamento dos dados, com a qual o usuário não tem contato direto e que geralmente mantém as regras de negócio). No projeto do Sistema de Gestão de Concessões, as tecnologias utilizadas no front-end foram HTML, CSS e JavaScript (com o framework AngularJS), ao passo em que o back-end foi construído baseado em Java (utilizando o Play Framework) e PostgreSQL. Sendo assim, as tarefas executadas pelo discente foram desde a construção de tabelas no banco de dados até a criação de telas com base em protótipos, conforme o detalhamento a seguir.

3.4.1 Criação de Tabelas e Evolução do Banco de Dados

Utilizando o PostgreSQL, o discente criou e também contribuiu para o mapeamento de relações no banco de dados, responsáveis pelo armazenamento dos dados do sistema. Além disso, relações pré-existentes foram modificadas, quando necessário, para comportar as evoluções do projeto. Tanto a criação quanto a modificação das tabelas eram realizadas através de scripts SQL, que passavam por uma validação pelo Administrador de Banco de Dados da empresa antes de serem executados no sistema. Neste ponto, foram encontradas dificuldades na criação dos Scripts quando estes deveriam fazer com que o banco de dados executasse cálculos e funções de certa complexidade. Nesses casos, o auxílio da equipe de desenvolvimento foi fundamental.

3.4.2 Mapemanto de Entidades

De acordo com as regras de negócio fornecidas e com o mapeamento do banco de dados, o discente atuou na modelagem do sistema através da criação

de classes Java integradas às tabelas do banco, obedecendo à arquitetura MVC, bem como no mapeamento das relações entre essas classes. Ao término do mapeamento, e seguindo o processo da empresa, o código era revisado por outro desenvolvedor da equipe a fim de garantir sua corretude.

3.4.3 Criação de Serviços

Utilizando as classes e tabelas mapeadas, o discente atuou diretamente na criação de serviços para executar operações do sistema (por exemplo, salvar floresta, editar concessionário, enviar mensagem etc.) por meio de métodos construídos nas classes Java. Tais serviços obedecem a um conjunto de regras e devem validar os dados recebidos a fim de garantir a integridade do sistema. Assim como nos mapeamentos de entidades, o código dos serviços passava por uma revisão por parte de outro desenvolvedor antes de ser aceito, de acordo com o processo de desenvolvimento de software da organização.

3.4.4 Criação de Telas

Utilizando as tecnologias HTML, JavaScript (com o framework AngularJS) e CSS, e obedecendo aos protótipos e regras fornecidos, o discente criou telas do sistema, que consomem os serviços do back-end e permitem que o usuário interaja com a aplicação através de um navegador compatível. As telas criadas normalmente validam os dados inseridos e retornam mensagens de confirmação ou erro, de acordo com a situação do sistema. Nestas atividades, uma dificuldade recorrente foi a familiarização com as linguagens utilizadas, dado o pouco contato prévio com as mesmas durante o curso de graduação. Tais dificuldades, a princípio, acarretaram em um ritmo mais lento de desenvolvimento, mas foram superadas com os estudos oferecidos pela empresa, a prática cotidiana e o apoio da equipe.

4 DESCRIÇÃO DAS ATIVIDADES

Neste capítulo estão relacionadas as atividades realizadas pelo discente durante o desenvolvimento do projeto. É necessário salientar que a execução das atividades do projeto referido teve início antes do desenvolvimento deste trabalho. No entanto, foi possível atuar em todos os módulos do sistema durante o período de estágio. Nos módulos cujo desenvolvimento já se encontrava adiantado, a atuação do discente se deu através de atividades nais quais o mesmo efetuou correções de erros ou aplicou melhorias. Em módulos mais recentes, como o Monitoramento de Contratos, o discente teve a oportunidade de atuar diretamente na evolução deles através da modelagem de classes ou criação de métodos, por exemplo.

O Sistema de Gestão de Concessões teve sua construção feita a partir de seus módulos. Novamente, eles são:

- Gestão de Florestas Públicas,
- Gestão de Licitações,
- Gestão de Concessionários,
- Gestão de Contratos,
- Monitoramento de Contratos,
- Central de Comunicação,
- Central do Administrador,
- Gestão de Contratos (perfil Concessionário).

A fim de pormenorizar o desenvolvimento das atividades, o submódulo Gerenciar Bonificações, contido na Gestão de Contratos, foi escolhido.

A estruturação deste submódulo teve início com a criação da tabela responsável por armazenar as bonificações no banco de dados. Tal tabela armazena

dados inerentes à bonificação, como as datas de início e fim de vigência, o indicador de referência e o percentual de bonificação. Também foram criadas mais duas tabelas para armazenar os períodos bonificadores e as sobreposições de períodos bonificadores, visto que um contrato pode ter mais de uma bonificação ativa em determinado período.

A partir daí, foram feitos o mapeamento da classe de bonificação no Java e as alterações no mapeamento do contrato para comportar as bonificações. Então teve início o processo de criação do serviço responsável por calcular as alterações nos preços dos contratos com base nas bonificações e seus percentuais, respeitando as sobreposições com outras bonificações ou apostilamentos, caso existam, e do serviço encarregado de salvar as bonificações.

Por último, foram criadas as telas Gerenciar Bonificações, Cadastro de Bonificação e Visualizar Bonificação. A primeira lista todas as bonificações cadastradas para determinado contrato. A segunda tela torna possível a inserção de informações para o cadastro de uma nova bonificação, mediante validação dos dados, e a simulação dos valores que o contrato assumirá caso o cadastro seja confirmado. A última tela permite que os dados cadastrados para uma bonificação específica sejam visualizados e que o preço contratual seja verificado em uma data específica, exibindo as sobreposições de bonificações e apostilamentos, caso existam.

Uma grande dificuldade encontrada na criação deste submódulo se deu pelo fato de que, apesar de ser um acontecimento raro, é possível que mais de uma bonificação esteja ativa ao mesmo tempo em um contrato, desde que seus indicadores de referência sejam diferentes entre si. Para atender a este requisito, foram criados os conceitos de período bonificador (com data de início e fim e uma bonificação) e período bonificador sobreposto (também possui data de início e fim e pode compreender um apostilamento e duas ou mais bonificações). Esta abordagem acarretou em um grande problema, pois a cada novo apostilamento

cadastrado, o cálculo dos períodos bonificadores, períodos sobrepostos e de seus respectivos valores deve ser refeito caso sejam identificadas bonificações afetadas pelo apostilamento, uma vez que os valores de referência do contrato são alterados por ele.

Além disso, o sistema deve comportar cenários em que várias bonificações estejam sobrepostas, ainda que tais situações sejam pouco prováveis, o que torna os cálculos e os testes muito complexos. Neste contexto, foram identificados erros nos casos em que muitas bonificações se sobrepõem e/ou quando se cadastra um apostilamento que exige que os cálculos de sobreposições sejam refeitos, pois o sistema calcula corretamente os valores, mas não as datas limites dos períodos de sobreposição. Esta funcionalidade continuou em aberto por alguns meses, sendo um dos grandes desafios do projeto.

Outro grande desafio enfrentado pela equipe de desenvolvimento se deu pelo fato de que o cliente constantemente solicitava alterações de funcionalidades e regras do sistema já validadas por ele, o que com frequência exigia refatorações no código que desencadeavam erros.

5 CRONOGRAMA DE ATIVIDADES

A seguir, encontra-se a listagem e o cronograma das atividades desenvolvidas durante o período de estágio do discente.

1 – Escolha do Tema e definição do orientador: a escolha do tema do projeto foi realizada tendo como base a área de atuação no estágio e a afinidade do discente com o tema.

2 – Estudos teóricos relacionados à área de Desenvolvimento de Software: a fim de construir uma base teórica para respaldar o desenvolvimento de software, o estagiário estudou sobre padrões de código, boas práticas de programação e conceitos de Orientação a Objetos. Alguns dos estudos tiveram como base a linguagem de programação Java afim de prover maior familiarização com a mesma para uso no projeto. Tais estudos foram realizados como parte do programa de treinamento oferecido pela empresa, que disponibilizou um de seus colaboradores para explicar os conteúdos e sanar eventuais dúvidas.

3 – Estudos sobre as ferramentas/tecnologias utilizadas na empresa: ainda no escopo do programa de treinamento, o discente realizou estudos sobre os processos da organização, os ritos SCRUM e as ferramentas e tecnologias utilizadas. O foco foi no aprendizado das tecnologias e na integração das mesmas para a composição do sistema. Dentre as tecnologias que foram estudadas, destacam-se Java, JavaScript, HTML e PostgreSQL.

4 – Desenvolvimento de Software: os módulos do sistema foram desenvolvidos ao longo do andamento do projeto levando em conta as prioridades definidas pelo cliente e pelo Product Owner e o escopo de cada Sprint. Eventualmente foram feitas entregas ao cliente para que o mesmo pudesse avaliar se o software em desenvolvimento atendia às suas necessidades e solicitar alterações quando necessário. A etapa de desenvolvimento abrangeu ainda as correções de erros encontrados no

sistema pelo Analista de Qualidade de Software ou pelo cliente e as considerações do Product Owner.

5 – Análise do processo de desenvolvimento: através dos ativos gerados durante o desenvolvimento de software, foi feita a análise do processo e das dificuldades encontradas. Tal análise foi realizada continuamente por meio das reuniões previstas pelo SCRUM, chamadas de Retrospective e Review, as quais aconteciam ao término de cada Sprint. Nessas reuniões, a equipe revisava o que foi de fato entregue para o cliente, contrastando com o que havia sido planejado, e discutia os pontos positivos e negativos da Sprint e quais ações poderiam ser tomadas a fim de melhorar o processo e o desempenho do time.

6 – Comparação dos resultados das análises: os dados levantados durante o desenvolvimento do sistema eram comparados a fim de mensurar o desempenho da equipe no decorrer das Sprints e ajustar o escopo de Sprints posteriores. Esta comparação era feita pela equipe durante as reuniões de planejamento de Sprint conhecidas como Planning. Nelas, o time SCRUM revisava quanto conseguiu entregar nas Sprints anteriores e definia as atividades que se comprometeriam a entregar na próxima Sprint.

7 – Confecção do relatório: concomitante ao desenvolvimento do projeto, as atividades executadas durante o estágio foram analisadas e inseridas no relatório pelo discente.

A tabela a seguir permite a visualização dos períodos de execução das atividades:

Tabela 5.1 – Cronograma das Atividades 1-7.

Atividades/Meses	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										

REFERÊNCIAS BIBLIOGRÁFICAS

BHANSALI, D. A. Angularjs: A modern mvc framework in javascript. *Journal of Global Research in Computer Science*, v. 5, n. 12, p. 17–23, 2014.

DEACON, J. Model-view-controller (mvc) architecture. *JOHN DEACON Computer Systems Development, Consulting Training*, p. 1 – 6, 2005.

MARQUES, J. R. S. Desenvolvimento de software para web usando java para uma aplicação de comércio eletrônico. p. 1–54, 2011.

MILANI, A. *POSTGRESQL Guia do Programador*. 1ª. ed. [S.l.]: Novatec Editora LTDA., 2008.

SILVA, F. G. Uma análise das metodologias Ágeis fdd e scrum sob a perspectiva do modelo de qualidade mps.br. *SCIENTIA PLENA*, v. 5, n. 12, 2009.