UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BRUNO GUILHERME ANDRETTA DE MIRANDA

ESTUDO E DESENVOLVIMENTO DE UM APLICATIVO WEB USANDO A METODOLOGIA DE DESENVOLVIMENTO DIRIGIDO A TESES (TDD) COM MÉTODOS ÁGEIS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2014

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BRUNO GUILHERME ANDRETTA DE MIRANDA

ESTUDO E DESENVOLVIMENTO DE UM APLICATIVO WEB USANDO A METODOLOGIA DE DESENVOLVIMENTO DIRIGIDO A TESTES (TDD) COM MÉTODOS ÁGEIS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2014

BRUNO GUILHERME ANDRETTA DE MIRANDA

ESTUDO E DESENVOLVIMENTO DE UM APLICATIVO WEB USANDO A METODOLOGIA DE DESENVOLVIMENTO DIRIGIDO A TESTES (TDD) COM MÉTODOS ÁGEIS

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Informática como requisito parcial para obtenção do grau de Bacharel no Curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof Wilson Horstmeyer Bogado

CURITIBA

2014

AGRADECIMENTOS

Agradeço ao meu orientador Wilson Horstmeyer Bogado por ter me guiado e motivado no desenvolvimento deste Trabalho de Conclusão de Curso.

Aos professores do curso Bacharelado em Sistemas de Informação que me proporcionaram ter os conhecimentos necessários para construir um futuro promissor no campo de sistemas de informação.

E aos meus pais, que sempre me deram bons conselhos e oportunidades durante minha vida.

RESUMO

Miranda, Bruno. Estudo e desenvolvimento de um aplicativo web usando a metodologia de desenvolvimento dirigido a testes (TDD) com métodos ágeis. 84 f. Trabalho de Conclusão de Curso – Curso de Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

Este trabalho de conclusão de curso apresenta os resultados do estudo sobre o uso de métodos ágeis em desenvolvimento de softwares. Nesse caso específico, o software é uma aplicativo web para controle e automatização de estágios para o Departamento de Informática na Universidade Tecnológica Federal do Paraná. As metodologias ágeis surgiram contrapondo as metodologias clássicas, com foco na interação constante com o cliente, pouca documentação e trabalha com requisitos mutáveis. Para o desenvolvimento da aplicação Web é usado algumas boas práticas da programação extrema. As utilizadas no projeto são o uso da técnica de testar o código antes de codificar (TDD), o uso de estórias de usuário, a criação de pequenas versões, reuniões com os cliente num curto período de tempo, padrão no código e codificação simples.

Palavras-chave: Metodologia Ágil, Desenvolvimento Dirigido a Testes, aplicativo Web

ABSTRACT

Miranda, Bruno. Study and development of a web application using the test driven development (TDD) with agile methods. 84 f. Trabalho de Conclusão de Curso – Curso de Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

This TCC presents the results of the study on the use of agile methods in software development. In this particular case, the software is a web application for control and automation of internship for the Department of Informatics at Federal Technological University of Paraná. Agile methodologies were opposing the classical methodologies, focusing on constant interaction with the client, working with little documentation and changing requirements. For the development of Web application is used some good practices of Extreme Programming. Those used in the design are the use of the technique to test the code before coding (TDD), the use of user stories, creating small versions, meetings with the client in a short time, the standard code and simple coding.

Keywords: Agile Methodology, Test Driven Development, Web Application

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

DAINF	Departamento de Informática
UTFPR	Universidade Tecnológica do Paraná
TDD	*Test Driven Development*
XP	*Extreme Programming*
IDE	*Integrated Development Environment*
JSF	*Java Server Faces*
JPA	*Java Persistence API*
JVM	*Java Virtual Machine*
MVC	*Model View Controller*
XML	*Extensible Markup Language*
POM	*Project Object Model*
PRAE	Professor Responsável da Atividade do Estágio

SUMÁRIO

1 INTRODUÇÃO

Na década de 70, o desenvolvimento de software era feito de forma desorganizada, desestruturada e sem planejamento (UTIDA, 2012). Os clientes, normalmente, ficavam insatisfeitos, pois a comunicação entre cliente e o desenvolvedor era escassa (PRESSMAN, 1995). Os prazos não eram cumpridos e os custos não correspondiam às expectativas. Esse período ficou conhecido como a "crise de software".

Para combater essa crise surgiu a necessidade de criar processos, métodos e técnicas que tornassem o desenvolvimento de software mais organizado, bem documentado e que atendesse as necessidades dos clientes. Com isso, surgiram os métodos de desenvolvimento, que posteriormente ficaram conhecidos como métodos tradicionais.

Os métodos tradicionais foram elaborados para planejar e documentar o software antes de ser implementado, pois na época em que foram criados os custos para correção de erros eram muito elevados (PRESSMAN, 1995). Porém, atualmente, o mercado está cada vez mais dinâmico, os prazos estão mais curtos e há constantes mudanças de requisitos, portanto, foi indispensável a criação de uma nova metodologia que agregasse a versatilidade do mercado. Assim foram propostos os métodos ágeis. Os métodos ágeis, em contraponto aos métodos clássicos que focam excessivamente na documentação, propõem uma maior proximidade com os clientes, pequenas versões e ciclos rápidos de desenvolvimento, simplicidade na aprendizagem e documentação e ser aberto a mudanças ao longo do projeto (DIAS, 2005). Neste trabalho, o sistema será desenvolvido utilizando uma metodologia ágil.

O intuito desse projeto é fazer um levantamento bibliográfico sobre as metodologias usadas no desenvolvimento de software (clássica e ágil) e compará-las. Após o estudo inicial será construído um aplicativo web para controle e automatização de estágio no DAINF (Departamento de Informática) localizado na Universidade Tecnológica Federal do Paraná (UTFPR), utilizando algumas práticas da Programação Extrema (XP), em especifico, será utilizado a prática TDD (*Test Driven Development*, em português, desenvolvimento dirigido a testes).

1.1 Justificativa da Escolha do Tema

Existem três motivos para a escolha desse tema. Primeiro, os métodos ágeis são, ainda, pouco utilizados no desenvolvimento de software no Brasil. Portanto, criar um aplicativo utilizando totalmente uma metodologia ágil mostrará que é possível o seu uso na construção de projetos de software. Segundo, o aplicativo desenvolvido abordará grande parte das disciplinas ministradas no curso, entre elas Fundamentos de Programação, Banco de Dados, Análise e Desenvolvimento de Software e Engenharia de Software. E terceiro e último, além de aplicar o conhecimento estudado no curso, será aprendido um novo conhecimento que é a tecnologia web.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo desse projeto é desenvolver um aplicativo web para controle e automatização de requisições de estágio para o Departamento de Informática na UTFPR empregando em seu desenvolvimento os métodos ágeis.

1.2.2 Objetivos Específicos

- Estudar na literatura as metodologias aplicadas no desenvolvimento de software
- Desenvolver um aplicativo web para controle e automatização de estágios na Universidade Tecnológica Federal do Paraná para o Departamento de Informática.
- Aplicar os princípios das metodologias ágeis na construção do sistema
- Analisar os resultados do desenvolvimento do aplicativo utilizando os métodos ágeis

1.3 Organização do documento

Este documento está dividido em cinco capítulos. No presente Capítulo temos a introdução do tema abordado no trabalho, bem como a sua justificativa e seus objetivos gerais e específicos. No capítulo 2, abordamos o levantamento bibliográfico, ou seja, uma pesquisa na literatura sobre os assuntos tratados no texto. O levantamento contém uma visa geral sobre desenvolvimento

de software, uma descrição dos métodos clássicos e dos seus modelos mais utilizados, um detalhamento sobre os métodos ágeis e seus modelos mais atuais, entre eles o XP (Pogramação Extrema, em português), e uma comparação entre os métodos clássicos e ágeis. Após o referencial teórico, será exposta a metodologia empregada no trabalho. Na metodologia explica-se como será aplicada uma metodologia ágil no desenvolvimento de nosso sistema de controle e automatização de estágios, que diferente da tradicional, opta por maior interação com cliente. Ainda nesse terceiro capítulo, temos a demonstração das tecnologias utilizadas no processo de desenvolvimento do aplicativo Web. No quarto capítulo, serão analisados os resultados advindos do desenvolvimento do aplicativo Web empregando uma metodologia ágil. E, por fim, no último capítulo, a conclusão do trabalho juntamente com os possíveis trabalhos futuros.

2 LEVANTAMENTO BIBLIOGRÁFICO

Nesse capítulo abordaremos uma revisão da literatura sobre o processo de desenvolvimento de software. Será exposto, de forma cronológica, o avanço no uso de boas práticas e métodos eficazes para criação de sistemas de computadores. Os primeiros métodos surgiram para suplantar a crise de software ocorrida na década de 70. Para isso foram criados processos e etapas bem definidas, onde cada etapa deveria ser finalizada e documentada antes de prosseguir para as seguintes. Porém, essas metodologias possuíam algumas desvantagens, como a documentação excessiva e pouca interação com o cliente, trazendo a necessidade da busca de novas metodologias. Com isso, surgiram métodos ágeis que valorizam a comunicação contínua com os clientes e que usa pouca documentação focando na geração de código simples e rápido.Nas próximas seções desse capítulo, serão demonstradas essas duas metodologias e os seus métodos mais significantes e no fim uma comparação entre elas.

2.1 Desenvolvimento de Software

Antes da década de 70, a criação de sistemas de informação era feita sem seguir nenhum método, processo ou prática e não possuía uma documentação de projeto. Com isso, os prazos não eram atingidos e os custos eram muitos elevados, sendo assim, o desenvolvimento de software não era viável para a necessidade crescente da época.

Nesse contexto surgiu a engenharia de software para suprimir a crise de software, com a criação de processos padronizados, planejados e estruturados (UTIDA, 2012). Essa nova abordagem possibilitou que as empresas conseguissem desenvolver softwares que atingissem os prazos e não criassem custos extras, o que fez compensar o gasto nesse tipo de produto tecnológico.

Esses processos de desenvolvimento ou metodologias de desenvolvimento são definidos com um conjunto de boas práticas para o desenvolvimento de software, sendo que essas práticas são definidas em fases ou etapas, que são subprocessos ordenados e gerenciáveis (SOMMERVILLE, 2003). Na próxima seção, serão expostas as primeiras metodologias criadas, que são

conhecidas como metodologias clássicas ou tradicionais.

2.2 Métodos Clássicos

As metodologias clássicas foram criadas para tornar o desenvolvimento de sistemas mais organizado através de métodos e técnicas (SOMMERVILLE, 2003). Os métodos tradicionais de desenvolvimento de software são caracterizados por uma documentação bem detalhada, planejamento extenso e etapas bem definidas (PRESSMAN, 1995).

Todos os métodos considerados tradicionais possuem quatro fases fundamentais: Especificação do software, Projeto e Implementação de software, validação de software e evolução do software (SOMMERVILLE, 2003). Cada ciclo gera um marco no projeto, que geralmente é algum documento, protótipo ou, até mesmo, uma versão do sistema.

Na fase de especificação do software é feita a análise do problema ao qual o software será desenvolvido, para isso, define-se, juntamente com o cliente, o escopo que abrange o projeto, as funcionalidades (requisitos) que o sistema precisa ter e as limitações do software. Essa etapa é de grande importância, pois determina todo o desenvolvimento do projeto.

A próxima etapa é a de projeto e implementação de software. Nessa fase são criados os modelos através de diagramas a fim de representar visualmente as funcionalidades do sistema, e a partir desses diagramas o software é codificado numa linguagem de programação.

Na terceira fase, ocorre a verificação e a validação do sistema. A verificação é feita para garantir que os requisitos declarados na etapa de especificação foram implementados corretamente. A validação garante que o software atenda todas as expectativas e necessidades do cliente antes da entrega do produto final (PRESSMAN, 1995).

Na última fase do ciclo, temos a evolução do software que consiste na entrega do software e a alteração do mesmo, se o for necessário para o cliente.

Com base nessas quatro etapas do modelo de processo de software foram criados vários métodos. A seguir abordaremos os principais: o modelo em cascata (PRESSMAN, 1995; DIAS, 2005), o modelo incremental (PRESSMAN, 1995; SOMMERVILLE, 2003), o modelo de prototipagem (PRESSMAN, 1995; SOMMERVILLE, 2003) e o modelo em espiral (BOEHM; GROUP, 1988).

2.2.1 Modelo em Cascata

O modelo em cascata foi o primeiro processo publicado de desenvolvimento de software. Conhecido também como linear ou sequencial ele foi proposto por Royce em 1970, em busca de criar uma metodologia que foca na construção de um software voltado a atender as necessidades dos usuários (BECK, 1999). Tem como base a definição de etapas bem delimitadas, onde cada fase possui uma documentação bem detalhada e as quais são realizadas sequencialmente.

Como qualquer modelo tradicional, o modelo de ciclo de vida possui as fases características de um desenvolvimento de software clássico. Portanto, é feito um levantamento de requisitos na etapa inicial, uma modelagem desses dados coletados, a codificação dos modelos, testes para verificar e validar a codificação e depois da entrega do produto, temos a manutenção do mesmo. O modelo em cascata dividiu os processos de desenvolvimento nas seguintes etapas (Figura 1): Modelagem, Projeto, Codificação, Testes, Manutenção.

- Modelagem: Etapa onde se estabelece o levantamento de requisitos para todos os elementos do sistema a ser desenvolvido e prossegue com a atribuição de alguns pequenos subconjuntos desses requisitos ao software. Os requisitos são documentados e revistos com os clientes antes de prosseguir à próxima fase.

- Projeto: Um processo múltiplo de passos que é subdividido em quatro atributos distintos: estrutura de dados (analisa de que forma os dados serão tratados), arquitetura de software (define a estrutura de como o projeto será desenvolvido), caracterização da interface (representa a interação do usuário com o sistema e detalhes procedimentais) e organização (o passo a passo do projeto) (PRESSMAN, 1995).

- Codificação: Tradução dos requisitos para linguagem de máquina, se os requisitos foram estabelecidos detalhadamente, esse processo será executado mecanicamente pelos programadores.

- Testes: Após a geração do código do sistema, inicia a fase de testes para garantir que cada função implementada está funcionando corretamente e se a integração entre elas também está.

- Manutenção: o software sofrerá mudanças durante o processo de implantação do sistema, pois poderão ocorrer erros não encontrados nos testes, mudanças externas não previstas e o cliente pode exigir acréscimo de alguma funcionalidade, portanto nessa etapa é necessário replicar todas as etapas anteriores do ciclo de vida para corrigir o sistema de acordo com as novas mudanças.

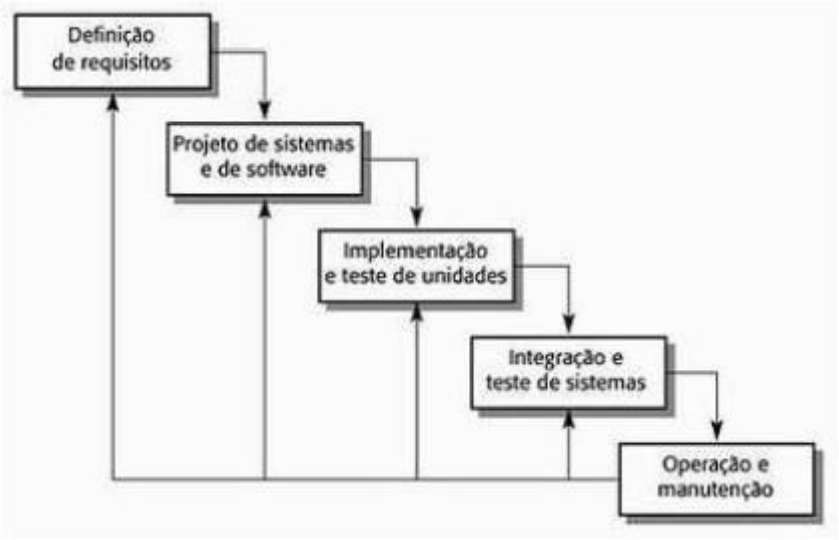


Figura 1: Modelo Cascata

Fonte: (SOMMERVILLE, 2003)

O modelo linear pode ser comparado com os princípios de produção em massa implantados pelo Taylorismo. A evolução sequencial do projeto é semelhante à de uma fábrica onde as funcionalidades (requisitos) são tratadas como matéria prima que são moldados à medida que os processos avançam na linha de produção. Cada transformação gera um conjunto de artefatos a serem analisados nas próximas etapas da produção. Procura-se assegurar que cada artefato seja produzido de forma correta para que o resultado final seja previsível e determinado. Dentro do projeto, os artefatos são representados como documentos criados para direcionar o trabalho a ser executado nas etapas posteriores.

O modelo em cascata é o mais utilizado no mercado, porém não é o mais eficaz e possui alguns problemas. O principal problema é a dificuldade de alterações durante o desenvolvimento, pois para modificar ou criar um requisito no projeto seria necessário refazer o ciclo de vida desde a etapa inicial (DIAS, 2005).

Numa tentativa de aprimorar este modelo, técnicas iterativas e incrementais surgiram, propondo uma metodologia menos sequencial e adaptativa as necessidades dos clientes. Um desses modelos é o modelo incremental abordado na próxima seção.

2.2.2 Modelo Incremental

O modelo incremental foi desenvolvido para tentar solucionar alguns problemas ocorridos no modelo em cascata. O desenvolvimento é divido em etapas, denominados de incrementos,

que é um conjunto de funcionalidades propostas pelo cliente.

Em cada incremento é realizado todo o ciclo do desenvolvimento de software utilizado no modelo em cascata. Cada etapa produz um sistema totalmente funcional, mesmo que não possua todos os requisitos do sistema. Esse ciclo de incrementos é realizado até todos os requisitos serem satisfeitos. O primeiro incremento é chamado de núcleo do produto que conterá as principais funcionalidades, as demais funções serão agregadas ao núcleo em cada incremento posterior (Figura 2).

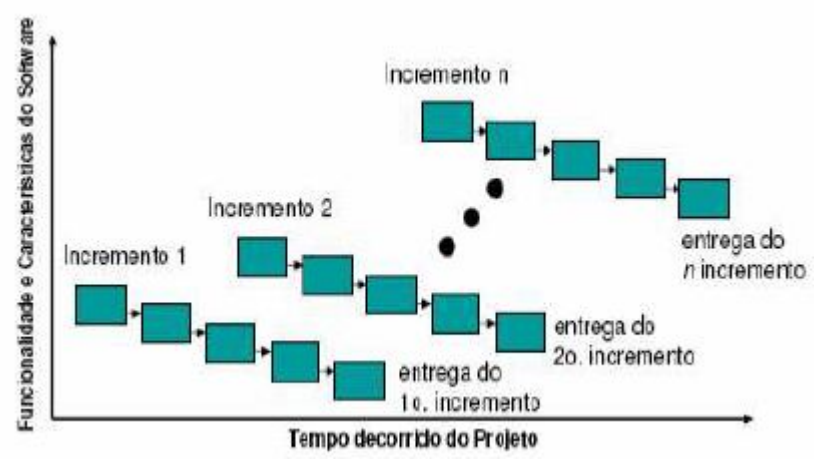


Figura 2: Modelo Incremental

Fonte: (PRESSMAN, 1995)

Essa metodologia trouxe algumas vantagens para o processo de desenvolvimento de software. A construção de um sistema funcional que contenha poucos requisitos facilita na identificação e correção de erros antes de adicionar novas funções. As necessidades não especificadas inicialmente podem ser implementadas nos próximos incrementos. Os *feedbacks* de interação anteriores podem ser usados nos próximos incrementos. Porém, o uso de incrementos também pode trazer desvantagens como o número de interações necessárias para a conclusão do projeto não poder ser definida no inicio do projeto e o gerenciamento e manutenção do sistema completo podem se tornar complexos.

Outro modelo que tem como principal característica a interatividade é modelo de prototipagem, que será discutido em sequencia.

2.2.3 Modelo de Prototipagem

Esse modelo é utilizado para sistemas desenvolvidos para clientes que não definiu os requisitos de forma clara. Ou para os desenvolvedores que não têm certeza do algoritmo a ser utili-

zado, na adaptabilidade do sistema operacional ou na interação homem-máquina a ser usada no projeto(PRESSMAN, 1995).

Essa metodologia se caracteriza pela criação de protótipos do sistema com as definições dadas pelo usuário, a fim de testar e validar as funcionalidades antes do desenvolvimento do sistema real (PRESSMAN, 1995)(ciclo de prototipagem pode ser visto na Figura 3).

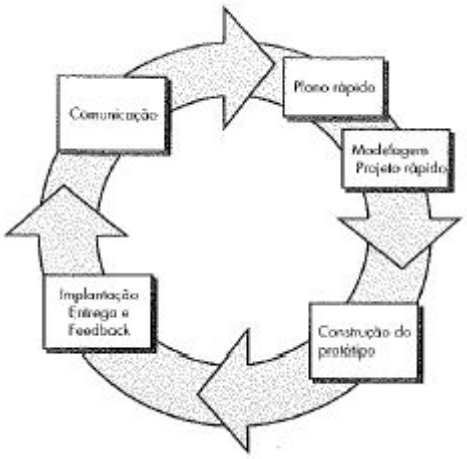


Figura 3: Modelo de Prototipagem

Fonte: (PRESSMAN, 1995)

O desenvolvimento de software através da utilização de protótipos pode diminuir a incerteza do projeto, pois o desenvolvedor e o cliente visualizam o sistema sem colocá-lo em prática. O protótipo ajuda a encontrar requisitos que o cliente poderia não ter visto inicialmente e, se bem feito, reduz o esforço na hora de desenvolver o sistema real. Porém, devido ao um gasto de tempo no protótipo, o cliente poderá querer que o produto seja desenvolvido mais rapidamente, tornando o protótipo no produto final.

Na procura de novas abordagens em modelos de desenvolvimento de software surgiu um modelo que combina a natureza da prototipagem e os aspectos sistemáticos do modelo em cascata, esse modelo foi denominado como modelo espiral e será apresentado a seguir.

2.2.4 Modelo Espiral

O modelo espiral foi proposto por Boehm (BOEHM; GROUP, 1988) e foi desenvolvido para abranger as características do modelo sequencial e o da prototipação, com um elemento novo, a análise de riscos (PRESSMAN, 1995). Em vez de representar o processo de software como uma ordem de atividades que retorna algo entre uma atividade e outra, o processo é representado como uma espiral (SOMMERVILLE, 2003).

O modelo define quatro regiões importantes no desenvolvimento do software (Figura 4):

- Planejamento: coleta de informação dos clientes, definição dos objetivos do sistema, analisar alternativas e restrições para o desenvolvimento e definição de custos e tempo de projeto.
- Análise de riscos: Gerenciamento e recuperação de riscos, a fim de solucioná-los.
- Engenharia: construção e desenvolvimento do sistema em si.
- Avaliação pelo cliente: análise pelo cliente sobre o sistema sobre as funções criadas até um ponto determinado.

Figura 4: Modelo em Espiral

Fonte: (PRESSMAN, 1995)

O paradigma de modelo espiral é a abordagem mais realística para o desenvolvimento de software em grande escala. Ela usa os passos sistemáticos do modelo em cascata, mas de forma interativa, a prototipação para reduzir os riscos e a análise de riscos para o cliente e o desenvolvedor entenderem e reagirem aos riscos encontrados em cada etapa do projeto (PRESSMAN, 1995).

Os modelos clássicos ou tradicionais são muitos utilizados no desenvolvimento de software, entretanto seu excesso de documentação e dificuldade de gerenciar mudanças impede que

empresas pequenas possam utilizar desse recurso para criar um produto de qualidade. Para resolver esse problema, foram criados os métodos ágeis. Os métodos ágeis e seus modelos serão expostos na próxima seção.

2.3 Métodos Ágeis

Os métodos ágeis surgiram como uma contraproposta às metodologias clássicas, como uma alternativa para se adaptar ao contexto do mercado atual, sob condições de constantes mudanças e incertezas (NETO, 2009). Desenvolvimento ágil de software é definido como uma abordagem de desenvolvimento que trata os problemas de mudanças rápidas: mudanças no mercado, nos requisitos, na tecnologia de implementação e na equipe de projeto (COCKBURN; HIGHSMITH, 2001).

Os métodos ágeis foram projetados para seguir quatro princípios: primeiro, produzir sistemas funcionais em semanas e alcançar *feedbacks* rápidos do cliente, segundo, criar soluções simples para não dificultar quando ocorrerem mudanças no projeto, terceiro, melhorar continuamente a qualidade do software para que cada iteração tenha menor custo de implementação e, quarto, testar constantemente a fim de detectar defeitos mais cedo (ABRAHAMSSON et al., 2002).

Os quatro princípios partilham de uma única premissa: o cliente aprende sobre as suas necessidades. Na medida em que o sistema fica disponível para o usuário manipular, novas necessidades surgem e as prioridades são redirecionadas, gerando mudanças no sistema. A aprendizagem é um fator importante agregado nos métodos ágeis, pois permite que o cliente foque apenas nas funcionalidades que tem maior valor para o seu negócio (ROCHA; OLIVEIRA; VASCONCELOS, 2004).

A popularização dos métodos ágeis ocorreu com o "Manifesto Ágil", que indica alguns princípios básicos (BECK et al., 2001):

- Indivíduos e interações são mais importantes que processos e ferramentas.
- Software em funcionamento é mais importante do que documentação excessiva.
- Feedback do cliente é mais importante do que negociação de contratos.
- Adaptação às mudanças é mais importante do que seguir um planejamento sequencial.

Na próxima seção, abordaremos os principais métodos ágeis que são: SCRUM (MAINART; SANTOS, 2010; FILHO et al., 2005; NETO, 2009; DIAS, 2005) e *Extreme Programing* (BECK et

al., 2001; MAINART; SANTOS, 2010; FILHO et al., 2005; NETO, 2009; DIAS, 2005; UTIDA, 2012). Também será exposto o *Test Driven Development* (SANTOS, 2010; SILVA, 2012; PRANGE, 2007) que consiste numa boa prática do XP e que será implantado na construção do aplicativo Web de controle e automatização de estágios.

2.3.1 *Scrum*

O *Scrum* é uma metodologia criada para gerenciar o processo de desenvolvimento de software onde os requisitos são voláteis. Ela é indicada para equipes pequenas de até dez integrantes, sendo ideal uma equipe que contenha cinco a nove integrantes. Essa abordagem se baseia na flexibilidade, adaptabilidade e produtividade, em que as técnicas usadas no construção do sistema ficam a cargo dos desenvolvedores (SCHWABER, 2004).

Os papéis dos membros do projeto são bem definidos nesse método e serão descritos a seguir:

- *Product Owner* (Proprietário do Produto): É o dono do negócio, responsável pelo financiamento, define os requisitos a serem implementados e analisa os resultados de cada iteração.
- *Scrum team* (Equipe Scrum): É a equipe formada por até dez pessoas. Ela é responsável por analisar os requisitos elaborados pelo proprietário e executá-los no projeto a fim de gerar uma versão do produto a cada iteração.
- *Scrum Master* (Mestre): Também pode ser chamado de gerente do projeto. Verifica se a equipe está produzindo de acordo com o planejado, gerencia os recursos do projeto e participa de todas as reuniões

O ciclo *Scrum* se inicia com a coleta de requisitos e definições de restrições feitas pelo cliente e documentadas nas *Product Backlog* (Tarefas do Produto). Os requisitos (tarefas) serão analisados e seus recursos estimados e divididos em *releases* (versões). Cada *release* é formado por um conjunto de tarefas prioritárias denominado *Sprint Backlog* (Tarefas do Sprint), e serão implementadas em iterações chamadas de *Sprint*. No inicio de cada *Sprint* temos uma reunião de planejamento (*Sprint Plannng Meeting*) para definir os membros da equipe e os recursos a serem alocados. Durante a execução dos *Sprints*, o time se reúne diariamente(*Daily Scrum Meeting*) e discute o projeto num tempo de aproximadamente quinze minutos. Ao final de cada ciclo, é realizada uma reunião de revisão(*Sprint Review Meeting*), onde a equipe mostra os resultados aos clientes. Por fim, o *Scrum Master* realiza uma reunião de equipe (*Sprint*

Restrospective Meeting) para analisar o progresso do software com objetivo de melhorar os processos, a equipe e o produto para os seguintes *Sprints* (NETO, 2009), na Figura 5 temos o ciclo de Desenvolvimento *Scrum*.

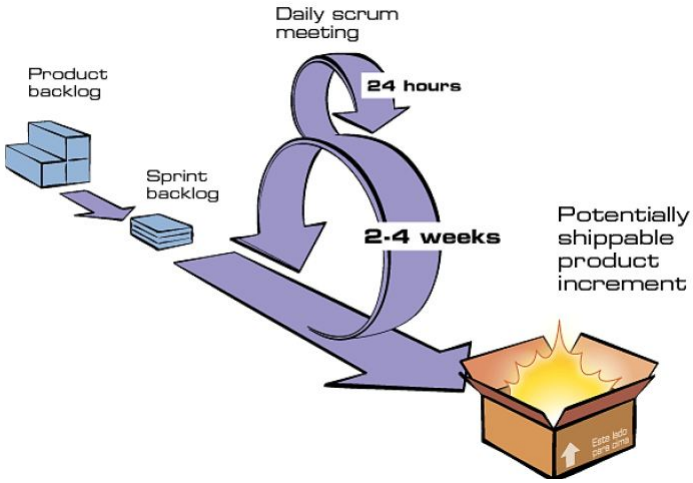


Figura 5: Ciclo de Desenvolvimento Scrum

Fonte: (IT, 2009)

2.3.2 *Extreme Programming*(XP)

A *Extreme Programming*(XP) é um método ágil para pequenas e médias equipes de desenvolvimento de software que possuem requisitos vagos e suscetíveis a mudanças (BECK, 1999). O XP visa garantir a satisfação do cliente, enfatizando o desenvolvimento ágil do projeto. Para isso, segue quatro valores primordiais: comunicação, simplicidade, *feedback* e coragem.

A comunicação é um aspecto importante para garantir um bom relacionamento entre o cliente e o desenvolvedor. Esta deve ser feita de maneira mais pessoal possível, pois agrega confiança no cliente e na equipe que irá desenvolver o software.

A simplicidade visa garantir uma implementação de códigos simples, limpo e sem funções desnecessárias. Para o XP é mais importante perder tempo em adicionar novas modificações do que implantar um código mais complicado que talvez possa nem ser usado (UTIDA, 2012).

Os *feedbacks* ajudam na compreensão da evolução do sistema tanto para o cliente quanto para os programadores. Dessa forma, eventuais erros podem ser corrigidos facilmente e novas melhorias podem ser feitas, direcionando o sistema de acordo com as reais necessidades do cliente.

E o último princípio, ter coragem para por em prática os valores anteriores. Tentar facili-

tar a comunicação entre cliente/desenvolvedor e gerente/equipe. Simplificar o código sempre que possível e estar disposto a receber os *feedbacks* do cliente e implantá-los no processo de desenvolvimento.

A XP, além de utilizar os quatros valores, baseia-se em doze práticas (BECK, 1999):

- Jogo do Planejamento: Em cada interação do projeto é feita uma reunião com os clientes, gerentes e desenvolvedores para determinar o escopo, as propriedades do negócio e as estimativas técnicas dessa versão. Na coleta de requisitos é usada uma técnica chamada de "estória de usuários", o cliente deve escrever seus requisitos em pequenos textos e com pouca informação, para ser possível criar versões rápidas de software.
- Pequenas Versões: Cada versão do sistema deve ser tão pequena quanto possível e que possa ser colocada em produção, é indispensável que cada entrega de versão ocorra em poucas semanas após a entrega da última.
- Metáfora: Representação de como o sistema funciona através da modelagem de objetos de negócio.
- Projeto Simples: A codificação do software deve ser o mais simples possível. Recurso utilizado para manter um custo de manutenção baixo.
- Testes: Os Testes são criados antes do código para antecipar algum erro de codificação. Todos os testes são automatizados e executados regularmente.
- Refatoração: Os desenvolvedores reestruturam o software durante todo o processo, sem modificar seu comportamento externo.
- Programação Pareada: A programação do sistema é feito por dois programadores que compartilham uma mesma máquina. Os pares são trocados frequentemente para que toda a equipe conheça o código e troquem experiências.
- Propriedade Coletiva: Os programadores de equipe tem acesso liberado para modificar qualquer parte do código do sistema.
- Integração Contínua: A integração do código é feita periodicamente. Essa prática viabiliza a descoberta de problemas precocemente e que todos da equipe trabalhem na mesma versão do sistema.
- Semana de 40 horas: O desenvolvimento de software deve ser realizado num determinado número de horas que não desgaste a equipe, pois longos períodos reduzem o desempenho e afetam a qualidade do produto final.

- Cliente junto aos desenvolvedores: O cliente precisa estar com a equipe de projeto sempre que disponível, para responder perguntas sobre os requisitos e direcionar o software para benefício de seu negócio.

- Padronização de Código: No inicio do projeto deve ser criado um padrão de codificação a fim de não poder identificar quem da equipe escreveu o código e auxiliar na condução do trabalho.

2.3.2.1 *Test Driven Development (TDD)*

O processo de testes de um sistema de informação é uma maneira de assegurar que a codificação está correta, completa e para nivelar sua qualidade. Portanto, o uso desse procedimento é de extrema importância para qualquer aplicação de software (PRANGE, 2007). O TDD é uma técnica de desenvolvimento de software ágil derivado da prática de testes aplicadas no método ágil XP. O conceito básico do TDD resume-se em fazer os testes antes de começar a escrever o código, de forma incremental. O uso da técnica de programar o sistema após os casos de testes viabiliza o entendimento do comportamento do software, pois para cada teste é definido como uma parte do sistema deve funcionar (SANTOS, 2010).

O desenvolvimento guiado por teste é constituído por um ciclo de iterações. Cada funcionalidade incorporada no sistema deve ser uma iteração e seguir cinco passos propostos pelo TDD (BECK, 2002) (Figura 6):

- Escrever o teste: O primeiro passo é escrever o caso de teste através da análise das estórias de usuário (Nos métodos ágeis cada requisito é escrito na forma de pequenos textos com o mínimo de funcionalidades).

- Verificar se o teste falha: O caso de teste deve falhar (simbolicamente representado por um sinal vermelho) na primeira instância, pois nenhum código foi implementado. Se o teste passar, sem qualquer alteração no código, então o teste deve estar errado e deverá ser refeito.

- Implementar o código: Após o teste falhar, o programador deve escrever o mínimo de código possível somente para passar no caso de teste. Assim o código implementado estará mais limpo e simples, facilitando uma manutenção posterior.

- Executar o teste novamente: agora com o código criado, será necessário executar o teste novamente. Se o teste passar (simbolicamente representado por um sinal verde) é execu-

tado o próximo passo senão o desenvolvedor deve analisar a falha, que pode ser algum erro no código ou fazendo outro teste falhar.

- Refatorar o código: O último passo é a da refatoração e a limpeza de código duplicado. Nessa etapa são permitidas apenas modificações estruturais do sistema. Após a refatoração é necessário executar o teste novamente para ver se a modificação alterou alguma parte do sistema.

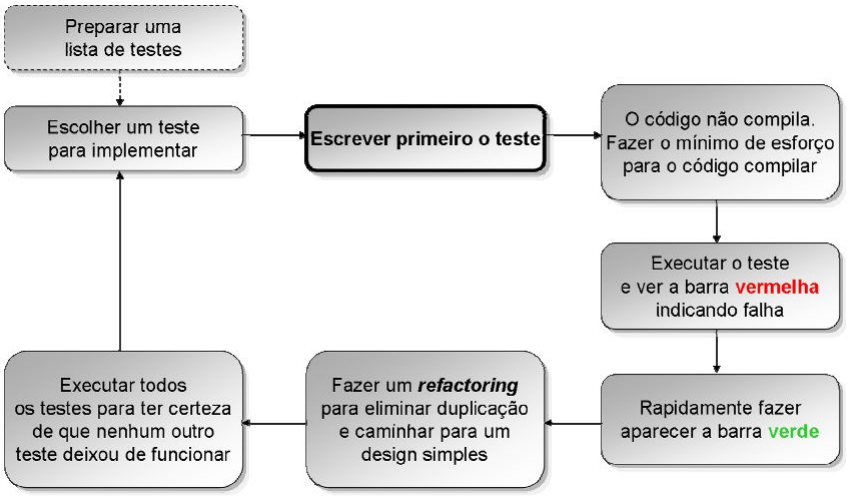


Figura 6: Ciclo de Desenvolvimento dirigido a testes

Fonte: (NETO, 2009)

Além de melhorias no desenvolvimento do projeto, o TDD possui uma grande quantidade de vantagens. O processo de desenvolvimento é simplificado, pois somente é implementado o código para os testes especificados naquele momento, não sendo necessário analisar todos os requisitos do sistema. Os códigos produzidos são automaticamente testados. O desenvolvedor tem um conhecimento aprofundado dos requisitos, pois ao escrever os casos de testes é preciso uma compreensão de todo o sistema. A utilização de depuração é bastante reduzida. O código fica mais limpo e simples, aumentando a qualidade do software.

Pela grande geração de testes comparada ao da metodologia tradicional, o TDD tem suas desvantagens a partir desse fato. Quando os requisitos são alterados, alguns testes podem falhar e deverão ser refeitos, aumentando o custo da manutenção. Dependendo do tamanho do sistema, ao executar todos os testes o processamento pode ficar lento e exigir muito do computador.

2.3.3 Comparação entre Métodos Clássicos e Métodos Ágeis

Depois da análise das características das metodologias (clássicas e ágeis) e de seus modelos mais significantes, apresentado uma comparação entre elas. Ambas possuem alguns processos em comum que justifica a comparação. Esses processos são uma sequência de atividades de software realizado por indivíduos ou equipe, a fim de gerar produtos para serem entregues aos clientes.

A metodologia tradicional surgiu para formalizar e padronizar o desenvolvimento de software, portanto foca-se em processos e algoritmos a fim de organizar os sistemas criados. Contrapondo essa ideia, os métodos ágeis valorizam mais a capacidade dos indivíduos e a boa comunicação entre eles para construir um produto de qualidade do que os processos e as ferramentas.

A documentação de um projeto é de extrema importância para o sucesso de um software, pois facilita a compreensão do sistema como um todo e é necessária para tomar decisões durante o desenvolvimento do sistema. Porém, o uso excessivo, comumente utilizado nos métodos tradicionais, pode trazer malefícios, uma vez que se a documentação não estiver sincronizada com o andamento do projeto, ela distorce a realidade e as decisões tomadas podem ser erradas. De acordo com manifesto ágil, a documentação deve ser a mínima possível e concisa.

Segundo os métodos ágeis, para gerar um produto com boa qualidade e que atenda as reais necessidades do cliente é preciso um *feedback* contínuo com o mesmo, pois ele orienta e direciona o desenvolvimento do sistema a cada iteração. Portanto, deve-se determinar um contrato que estabeleça a forma de comunicação do cliente com a equipe do projeto. Na metodologia clássica, o contrato é feito para determinar custos, prazos e requisitos de todo o sistema e o *feedback* só acontece no final do projeto, o que pode trazer alguns transtornos visto que há uma possibilidade de mudança de requisitos após o fechamento do contrato.

Uma das diferenças mais perceptíveis entre as metodologias tradicionais e os métodos ágeis é que as metodologias tradicionais são preditivas e as ágeis são adaptativas. Ser preditivo significa prever as mudanças antes de elas acontecerem de fato. Portanto, o desenvolvimento de software preditivo propõe um planejamento rigoroso, no início do projeto, de todas as mudanças que possam ocorrer durante o processo de construção do software. Porém, esta prática pode causar problemas ao desenvolvimento, devido ao fato dos requisitos, normalmente, mudarem ao longo do projeto. Pressupondo que os requisitos sempre são mutáveis, os métodos ágeis utilizam de técnicas adaptativas para o desenvolvimento de software, ou seja, eles se adaptam aos novos fatores que surgem durante o desenvolvimento do sistema.

Ambas as metodologias de desenvolvimento de software possuem pontos fortes e fracos. Enquanto os métodos clássicos focam em projetos grandes que podem dar ao luxo de construir uma documentação detalhada, os métodos ágeis, pensando na acessibilidade de pequenas empresas, construíram uma metodologia capaz de facilitar o desenvolvimento de software para grupos menores. A metodologia tradicional deve ser aplicada em situações em que os requisitos são estáveis e mudanças futuras são previsíveis. Já os métodos ágeis são indispensáveis para sistemas com requisitos mutáveis, equipes pequenas e o seu desenvolvimento precisa ser rápido.

3 METODOLOGIA

Nesse capítulo, abordaremos a metodologia empregada no desenvolvimento do aplicativo web para controle e automatização de estágios. Devido ao fato do sistema ser Web, ele é, essencialmente, dinâmico, por isso, o uso de uma metodologia ágil é necessário para a criação do produto. O projeto será desenvolvido aplicando algumas práticas do XP, sendo estas, adaptadas aos recursos disponíveis. Em específico, o uso da prática de desenvolvimento dirigido a testes será usado em todo o processo de criação do aplicativo. Como o aplicativo Web é projetado para seguir as regras do TDD será necessário o uso de testes automatizados. Para testes unitários será utilizado o *framework JUnit* e para testes de integração a ferramenta *Cucumber*.

O sistema será desenvolvido utilizando a plataforma de ambiente integrado *Netbeans IDE*, onde será usada a linguagem Java para codificação da lógica do aplicativo, o *framework* JSF(*Java Server Faces*) integrado com *ICEFaces* para codificação da interface do sistema e o *JasperIreport* para geração de relatórios em PDF. Para integrar o banco de dados com a aplicação, utilizaremos a API padrão de persistência do Java (JPA-*Java Persistence API*) que define um meio de mapeamento objeto-relacional que será implementado pelo *framework Hibernate*. Um princípio do desenvolvimento Web é a necessidade de um servidor para rodar a aplicação, portanto para essa função será utilizado o *Apache Tomcat*. Outro aspecto de uma aplicação Web é o uso de varias bibliotecas adicionais, porém ficar adicionando bibliotecas sempre que for instalar o projeto em outro computador é muito desgastante, portanto para facilitar esse processo será usada a ferramenta *Maven* para automação de compilação.

O sistema, inicialmente, terá apenas um desenvolvedor, porém como o software será *open-source* e implantado numa universidade federal, futuramente, o projeto poderá ser modificado, portanto um sistema de controle de versão será implantado e esse será o Git.

3.1 Práticas da XP

As boas práticas da XP são necessárias para um melhor aproveitamento do desenvolvimento ágil de software. No aplicativo Web que será desenvolvido nesse trabalho, não será possível utilizar todas as práticas propostas, pois o sistema terá apenas um desenvolvedor, portanto práticas que são planejadas para serem implantadas em equipes serão descartadas.

Para fazer a coleta de requisitos será empregada a técnica de estórias de usuários. A estória de usuário consiste em o usuário descrever as necessidades que ele gostaria de ver no sistema em pequenos textos, para esse levantamento será utilizados blocos de notas de papel, onde cada folha apresentará dois itens: a função do usuário no sistema e as suas necessidades. O uso de pequenos textos é necessário para evitar uma descrição extensa e pouco concisa, esse recurso facilita na hora de gerar versões rápidas do sistema.

Os métodos ágeis focam na interação contínua com o cliente para facilitar as modificações no projeto e garantir um produto que atenda às reais necessidades dele. Portanto, nesse projeto, após a coleta inicial dos requisitos, será feito a cada curto período de tempo, uma reunião com os clientes com uma pequena versão funcional do sistema onde teremos um retorno dos usuários a fim de direcionar o rumo do projeto.

A codificação do sistema seguirá um padrão e será implementado de maneira que seja limpo e simples. O uso dessas práticas será necessário posteriormente, quando o sistema estiver implantando na universidade e outro desenvolvedor quiser fazer alterações no sistema. O tempo dedicado para o desenvolvimento do sistema será de quatro horas diárias em cinco dias das semanas. Esse número foi determinado para não atrapalhar o desempenho no projeto e nas outras atividades externas do desenvolvedor.

Outra prática que será utilizada é o uso de testes e refatoração antes da codificação. Uma derivação dessas práticas e o TDD, que será incorporado nesse projeto. No desenvolvimento do sistema serão utilizados testes unitários e testes de integração. Os testes unitários são necessários para validar os dados válidos e inválidos dos métodos no software. Feito após os testes unitários, o teste de integração visa validar grupos de métodos integrados, validados pelos testes de unidade separadamente.

3.2 Tecnologias utilizadas

3.2.1 *JUnit*

O *JUnit* é uma implementação do *framework xUnit* feito para a linguagem Java que permite a criação de testes de unidades. Com ele, é possível verificar se cada método de uma classe está funcionando como esperado, exibindo os prováveis erros ou falhas.

No *JUnit* podemos identificar uma classe de teste quando seu nome termina com "*Test*". Também é possível usar anotações para detectar métodos de testes na classe, utilizando *@Test*. Para verificar se os métodos estão retornando os resultados esperados, o *JUnit* utiliza de asserções que analisa a consistência dos parâmetros.

Para facilitar o entendimento, na Figura 7 temos um exemplo simples. A classe *CalculaSomaTest* é uma classe de teste que possui um método de teste *testarSoma()* identificado com a anotação *@Test*. Para verificar se o resultado do cálculo está correto é utilizada a asserção *Assert.assertEquals*, que analisa se os objetos são semanticamente iguais.

```
public class CalculaSomaTest {

    @Test
    public void testarSoma(){
        Calculadora calc = new Calculadora();
        int resultado = calc.soma(1, 2);
        Assert.assertEquals(3, resultado);
    }

}
```

Figura 7: Exemplo Simples de *JUnit*

Fonte: Autoria própria

3.2.2 *Cucumber*

Após serem implementados os testes de unidade devemos integrar os métodos para verificar se os métodos estão trabalhando corretamente em conjunto, para realizar os testes de integração utilizamos a ferramenta *Cucumber*. O *Cucumber* é um *framework* que permite fazer testes de integração a partir da escrita de estórias de usuários, ou seja, as funcionalidades (*features*) do sistema dividindo-as em cenários (*Scenario*).

Na Figura 8, temos um exemplo de uma estória de usuário na forma de cenário que verifica

se quando o usuário entrar na página inicial da aplicação Web aparece a frase “Olá Mundo” na tela.

```
Feature: Abrir a página inicial

  Scenario: Abrir página incial
    Given Eu estou na página inicial
    Then Eu poderia ver "Olá, mundo!"
```

Figura 8: Exemplo Simples de *Feature* no Cucumber

Fonte: Autoria própria

Depois de escrever a funcionalidade, temos que executá-la numa maneira que os métodos em Java identifiquem os cenários, escritos numa linguagem de alto nível. Para isso, temos uma classe de passos (*Steps*) que através de anotações identifica os cenários e separando-os em testes de unidade, pode ser visto na Figura 9.

```
@Given("^Estou na página Inicial$")
public void openHomePage() throws Throwable {
    driver.get("http://localhost:8087/HelloWorld");
}

@Then("^Eu poderia ver \"([^\"]*)\"$")
public void shouldSee(String text) throws Throwable {
    String source = driver.getPageSource();
    Assert.assertTrue("Source: " + source, source.contains(text));
}
```

Figura 9: Exemplo Simples de *Step* no Cucumber

Fonte: Autoria própria

3.2.3 *NetBeans IDE*

NetBeans IDE é um ambiente de Desenvolvimento Integrado de código-fonte aberto gratuito para desenvolvimento de software. O IDE é executado em muitas plataformas, como *Windows* e Linux. Ele possui todas as ferramentas necessárias para criar aplicações *desktop*, corporativas, Web e móveis com a plataforma Java. Essa plataforma suporta vários *frameworks* essenciais numa aplicação Web, como *JSF, Hibernate e Apache Tomcat*. Possui grande quantidade de bibliotecas e *APIs* para garantir que o desenvolvedor tenha foco na produtividade e na rapidez no seu projeto. O *NetBeans IDE* é escrito totalmente em Java e funciona em qualquer sistema operacional que suporte a máquina virtual Java (JVM).

3.2.4 *Java Server Faces*

Java Server Faces é um *framework* para desenvolvimento Web criado para simplificar a modelagem das interfaces do usuário. O JSF é baseado na arquitetura MVC, o que torna o projeto mais organizado, pois separa a visualização dos modelos de negócio.

O padrão MVC possui três camadas (modelo, visualização e controle), onde o modelo é responsável por representar os objetos de negócio, a visualização é representada pela interface do usuário e o controle faz a ligação entre o modelo e a visualização.

Na camada de controle, o *JSF* utiliza-se de *servlets* chamados de *Faces Servlets*, por arquivos XML de configuração e por vários manipuladores de ações. O *Face Servlets* recebe as requisições dos usuários na Web, redireciona para o modelo e envia uma resposta. Os arquivos de configuração possuem informações sobre os mapeamentos das regras de navegação. Na camada do modelo, é executada uma lógica de negócio ao receber dados da camada de visualização. A visualização é composta por uma hierarquia de componentes, sendo possível criar interfaces mais ricas e complexas (Figura 10).

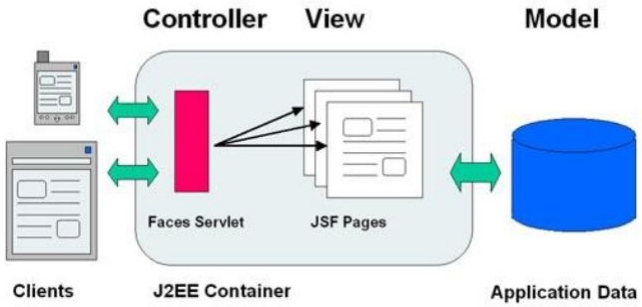


Figura 10: Estrutura do JSF

Fonte: (ALGAWORKS, 2010)

3.2.5 *ICEFaces*

ICEfaces é um *kit* de desenvolvimento de software *open-source* que estende o *JavaServer Faces*, empregando Ajax. Ele é usado para construir aplicações ricas para internet utilizando a linguagem de programação Java. Com *ICEfaces*, a codificação de interação e Ajax no lado do cliente é programado em Java, ao invés de *JavaScript*, ou com *plug-ins*.

3.2.6 *JPA e Hibernate*

O *JPA* é uma API padrão do Java para fazer a persistência ao mapear objetos para um banco de dados. Para implementar esse padrão o principal *framework* é o *Hibernate*.

O *Hibernate* é um *framework* para mapeamento objeto-relacional escrito na linguagem Java. Esse *framework* simplifica o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo de objeto de uma aplicação, mediante o uso de anotações.

O *framework* gera as chamadas SQL e libera o programador do trabalho manual da conversão dos dados resultantes, mantendo o programa portável para qualquer banco de dados SQL.

3.2.7 *Apache Tomcat*

O *Apache Tomcat* é um servidor Web Java, desenvolvido pela *Apache Software Foundation*, é distribuído como software livre.

Como servidor Web, o *Apache Tomcat* provê um servidor Web HTTP puramente em Java. O Servidor inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando-se manualmente arquivos de configuração formatados em XML.

3.2.8 *Maven*

O *Maven* é uma ferramenta de automação de compilação utilizada primariamente em projetos Java. O *Maven* utiliza um arquivo XML (POM) para descrever o projeto de software sendo construído, suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e *plug-ins* necessários.

Segundo a estrutura proposta pelo *Maven*, com um arquivo simples (POM) é possível compilar o código fonte, executar todos os testes, gerar o artefato final (em Java, o arquivo .jar) e gerar uma documentação básica sobre o projeto.

3.2.9 *Git*

O *Git* é um sistema de controle de versionamento gratuito e de código livre. Cada clone do *Git* é um repositório com todos os recursos com histórico completo e recursos completos de rastreamento das últimas versões, não depende de acesso à rede ou de um servidor central.

3.2.10 *JasperIreport*

O *JasperReports* é um *framework* para a geração de relatórios. É uma ferramenta totalmente *open source* e gratuita.

Os passos para gerar um relatório são bem simples. O primeiro passo é compilar o relatório em XML. Depois da compilação, o resultado é um objeto do tipo *JasperReport*. O próximo passo é preencher o relatório com os dados, e o resultado dessa etapa fica armazenado em um objeto do tipo *JasperPrint*. Esse objeto já representa o relatório finalizado, a partir dele podemos enviar para impressão diretamente, ou podemos exportar para um outro formato, tal como PDF por exemplo (Figura 11).

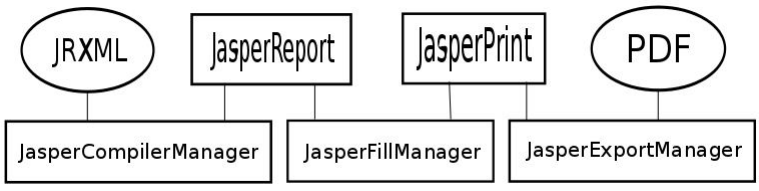


Figura 11: Processo do *JasperIreport*

Fonte: (K19, 2010)

4 ESTUDO DE CASO

Nesse capítulo, será explanado como foi realizado o desenvolvimento do aplicativo Web para controle e automatização de estágios para o Departamento de Informática na Universidade Federal do Paraná.

Inicialmente, é necessário explicar o funcionamento do processo de estágio aplicado atualmente, bem como toda a sua estrutura, sendo estas: as pessoas envolvidas e as suas respectivas funções no tratamento de estágio e os relatórios utilizados na formalização dos processos de contração e controle do período de estágio.

No quesito de desenvolvimento em si é detalhados dois aspectos: o uso dos métodos ágeis e a codificação do aplicativo. No gerenciamento do projeto, foram utilizadas as estórias de usuários para transcrever as necessidades de cada usuário que terá acesso ao sistema e também, foi criado um cronograma de reuniões para, em um curto período de tempo, o usuário ter um retorno do que foi realizado no projeto. Na codificação, o uso da prática de TDD (gerar os testes antes da codificação) permitiu a padronização do código e a diminuição do uso de depuração do código.

Sendo o sistema um aplicativo web, o projeto foi desenvolvido utilizando um servidor *Tomcat* para rodar as páginas feitas em JSF. No controle das páginas é usada a linguagem Java e para o gerenciamento dos dados é usado o banco de dados *Posgresql*.

4.1 Processo de Estágio

O estágio, para o caso estudado nessa monografia, é o ato educativo escolar supervisionado, desenvolvido no ambiente de trabalho, que visa à preparação para o trabalho produtivo de educandos que estejam frequentando o ensino regular em instituições de educação superior.

O estágio visa ao aprendizado de competências próprias da atividade profissional, objetivando o desenvolvimento do educando para a vida cidadã e para o trabalho.

O estágio poderá ser obrigatório ou não obrigatório. Estágio obrigatório é aquele definido como tal no projeto pedagógico do curso, cuja carga horária é requisito para aprovação e obtenção de diploma. Estágio não obrigatório é aquele desenvolvido como atividade opcional, acrescida à carga horária regular e obrigatória (DIEEM, 2014c).

Na UTFPR, o processo de estágio usado atualmente é totalmente feito a partir de relatórios em papel que comprovam se o estágio está sendo realizado de acordo com as exigências propostas pela universidade. Quem gerencia esses relatórios é o professor responsável por estágio que deve guardar esses documentos e avisar as pessoas envolvidas sobre cada etapa do processo.

Nas seções a seguir será detalhado como o processo de estágio funciona a partir das pessoas que estão envolvidas diretamente nesse fluxo e de acordo com os relatórios gerados que formalizam o processo em toda a sua dimensão.

4.1.1 Pessoas envolvidas

As pessoas envolvidas, a partir da visão do departamento de informática, são: o aluno, o professor orientador, o supervisor da empresa e o professor responsável.

O aluno inicia o processo de estágio ao criar o plano de estágio com as informações do estágio ao qual ele quer ingressar. Se aprovado, ele deve apresentar um relatório parcial sobre o estágio a cada semestre. Se o estágio for obrigatório, deve ser feito um relatório descritivo ao término do estágio, além de uma apresentação a uma banca descrevendo suas atividades no estágio.

O Professor Orientador deve acompanhar os aluno$s$ durante todo o período de estágio. Para isso, ele deve realizar uma visita à empresa para avaliar se está de acordo com as normas da instituição de ensino. A cada seis meses, o professor orientador analisa o relatório parcial do aluno e o aprova ou não. Em caso de estágio obrigatório, ele deve avaliar o relatório descritivo e a apresentação do aluno através de um relatório chamado ficha de avaliação.

O Supervisor da Empresa orienta o estagiário durante o período de estágio e gera a cada seis meses um relatório parcial sobre o desempenho do estagiário em relação à realização das atividades propostas para ele.

O Professor Responsável (PRAE)aprova o plano de estágio dos alunos, fica responsável por adicionar os relatórios do processo de estágio no sistema integrado de estágio da universidade, gerencia as datas de conclusão de estágios obrigatórios e fica no encargo de avisar os alunos sobre os relatórios pendentes. Portanto, o professor responsável é o coordenador do processo

de estágio.

4.1.2 Gerenciamento de Relatórios

Todo o sistema de estágio é baseado em relatórios em papel que identificam informações sobre o estagiário, sobre a empresa e sobre as atividades desenvolvidas no período de estágio. Portanto, nessa seção serão descritas de forma sucinta todas as etapas do estágio a partir dos relatórios gerados.

A primeira etapa consiste no aluno conseguir ser selecionado numa vaga de estágio, sendo possível estagiar numa empresa externa ou dentro da universidade. Conseguido a vaga, o aluno deve gerar um plano de estágio com os seus dados pessoais, os dados da empresa, as atividades desenvolvidas, as datas de início e fim do estágio e o dados do professor que irá orientá-lo (este pode ser escolhido pelo aluno ou indicado pelo professor responsável) (ver anexo A). Após a conclusão do preenchimento, o aluno deve entregar o plano de estágio juntamente com o termo de compromisso (empresa externa) ou somente o plano (a empresa é a universidade) para avaliação. O PRAE avalia se o aluno pode fazer o estágio de acordo com o seu período na faculdade e pelo horário de estágio que não podem coincidir com os horários das aulas. Se aprovado, o aluno poderá começar o programa de estágio, se não o aluno deverá refazer o plano de estágio e o termo de compromisso (se houver) ou aguardar até o momento que ele possa realizar o estágio (Figura 12).

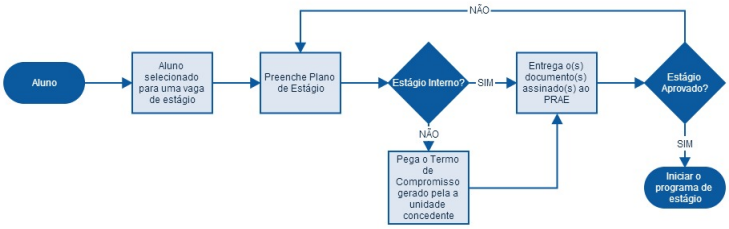


Figura 12: Fluxograma do primeira etapa do processo de estágio
Fonte: Autoria Própria

Ao chegar à data de término do termo de compromisso, o estagiário pode prorrogar o seu estágio para mais um período, sendo que o estágio só pode ser realizado num tempo máximo de dois anos. Para formalizar esse novo contrato, a empresa deve gerar um termo aditivo contendo informações adicionais ao do termo de compromisso, tendo que pelo menos alterar a data de inicio e término do novo período de estágio. Concluído o termo aditivo, o aluno deve levar

ao PRAE para que ele declare o termo aditivo válido ou não. No caso de inválido, o aluno deve comunicar à empresa para fazer as devidas alterações e gerar um novo termo aditivo, caso contrário, o estagiário continua o estágio por mais esse tempo determinado no termo (13).

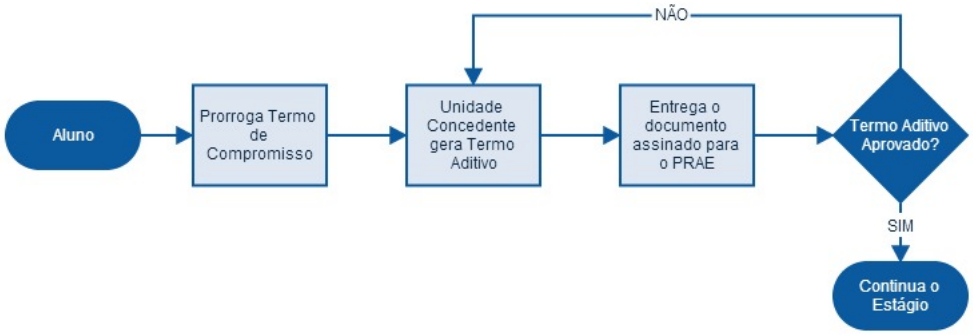


Figura 13: Fluxograma da geração de um termo aditivo

Fonte: Autoria Própria

Terminado o processo de contratação com o plano de estágio e o termo de compromisso, o professor orientador tem no máximo seis meses para fazer uma visita à empresa no qual o aluno foi prestar estágio. No caso de vários alunos estagiarem na mesma empresa, o professor orientador pode visitar a empresa apenas uma vez, porém deve gerar um relatório de visita (ver anexo B) para cada aluno. Esse relatório de visita visa garantir que o aluno está num ambiente propicio para aprender e desenvolver seus conhecimentos na área de formação. Após o preenchimento do relatório, o PRAE deve avaliar o relatório e, se aprovado, lançá-lo no sistema integrado de estágio da universidade (Figura 14).

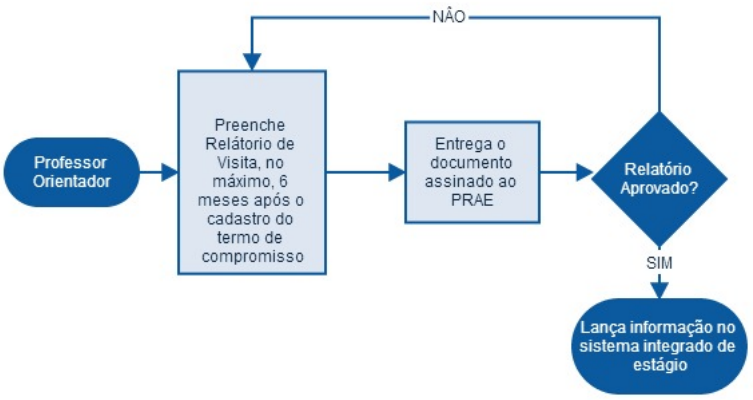


Figura 14: Fluxograma da geração de um relatório de visita

Fonte: Autoria Própria

A cada seis meses ou na conclusão do estágio, o estagiário deve escrever um relatório parcial sobre o ambiente da empresa onde trabalha, se ele está de acordo com as exigências impostas pela universidade, quais conhecimentos adquiridos no curso de formação são aplicados na prática e, no final, avaliar o estágio com o ponto de vista do aluno (ver anexo C). Esse relatório visa acompanhar o andamento do estágio, por isso, o professor orientador deve lê-lo e aprová-lo (Figura 15).

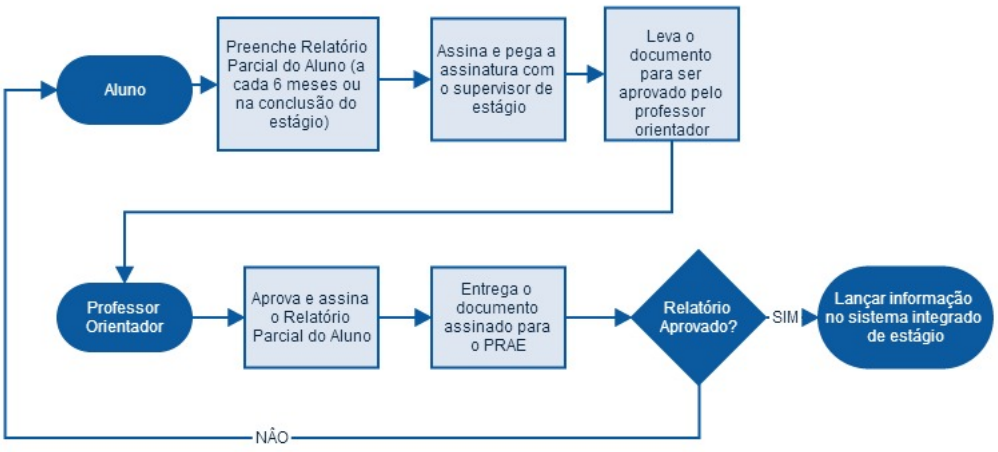


Figura 15: Fluxograma da geração de um relatório parcial do aluno

Fonte: Autoria Própria

Semelhante ao relatório parcial do aluno, o supervisor da empresa deve a cada seis meses,

na conclusão do estágio ou na conclusão de uma etapa do estágio obrigatório (somente se algum dos dois casos anteriores não ocorrer) gerar um relatório parcial. O documento avalia o estagiário a partir da visão da empresa, respondendo questionários sobre o desempenho do aluno no período determinado (ver anexo D). Conclui-se essa etapa, com a análise do PRAE sobre o relatório e aprovação do mesmo (Figura 16).

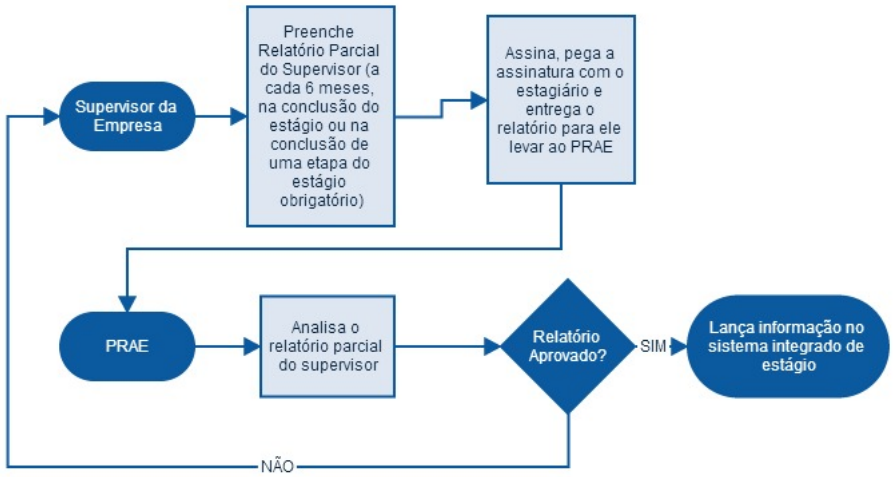


Figura 16: Fluxograma da geração de um relatório parcial do supervisor da empresa

Fonte: Autoria Própria

Os relatórios a seguir expõem apenas os documentos necessários na fase de estágio obrigatório, portanto é importante ressaltar que cada curso pode dividir a carga horária em mais de um período, porém os relatórios são necessários independentemente do curso.

Na conclusão do primeiro período do estágio obrigatório ou o único (depende de cada curso), o estagiário cria um relatório sobre a empresa, detalhando o que é a empresa, como ela está estruturada, sobre as atividades desenvolvidas que foram praticadas etc. (ver anexo B). Terminado o texto, ele deve ser passado ao professor orientador no qual fará uma avaliação do documento, a fim de verificar a validade do estágio (Figura 17).

Figura 17: Fluxograma da geração de um relatório descritivo
Fonte: Autoria Própria utilizado o site gliffy

A última etapa do estágio obrigatório ou a única (dependendo do curso), o estagiário faz uma apresentação para uma banca, onde ele deve explanar sobre o que foi aprendido com o estágio, o seu relacionamento com a empresa e sobre os conhecimentos que interligam os conteúdos de formação do curso com a prática empresarial. Cada integrante da banca dará uma nota à apresentação que será repassada ao PRAE que concluirá a aprovação do aluno na disciplina de estágio obrigatório (Figura 18).

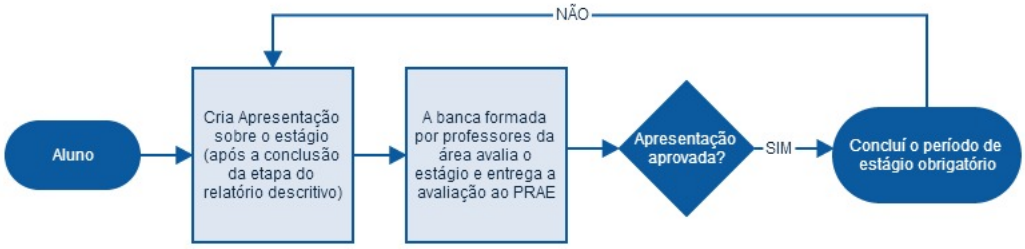


Figura 18: Fluxograma da geração da apresentação do relatório descritivo
Fonte: Autoria Própria utilizado o site gliffy

A cada etapa finalizada do estágio obrigatório, o professor orientador faz uma avaliação dessa fase a partir dos documentos entregues pelo aluno: relatório descritivo e a apresentação final para banca (ver anexo E). Finalizado a avaliação, o professor entrega o documento ao PRAE que lança no sistema integrado de estágio (Figura 19).

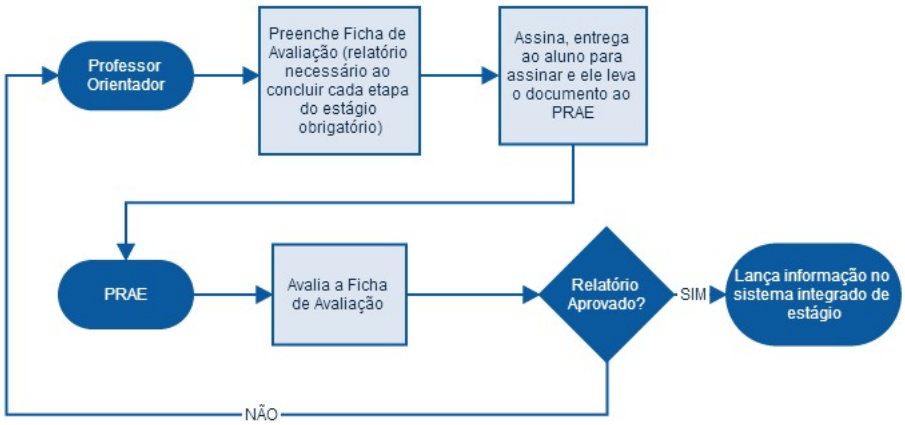


Figura 19: Fluxograma da geração da ficha de avaliação do professor orientador
Fonte: Autoria Própria utilizado o site gliffy

4.2 O sistema de controle e automatização de estágio

Após a análise do processo de estágio na UTFPR, pode-se notar que o controle dos relatórios (em papel) fica com o PRAE, portanto o sistema atual sobrecarrega o coordenador de estágio com a responsabilidade de avisar as pessoas envolvidas de todas as etapas do estágio, o que gera o desconhecimento das outras pessoas envolvidas, devido a esta centralização.

O sistema de controle e automatização de estágio visa contornar essa centralização no professor responsável ao permitir que os relatórios possam ser visualizados por todas as pessoas que fazem parte do processo de estágio a qualquer momento. Por isso, o sistema criado é totalmente online, permitindo o acesso dos relatórios através de um usuário, portanto há um controle do que cada usuário poderá visualizar.

Os relatórios em papel são transcritos para um formato digital através de uma ferramenta de criação de relatórios em PDF. Como o sistema tem integração com o banco de dados da universidade, é possível preencher os dados do relatório automaticamente, somente puxando as informações já existentes. Desse modo, descarta o uso de papel e é possível consultar os relatórios diretamente no site.

O aplicativo, também, tem como objetivo armazenar todas as informações sobre o processo de estágio, isso é feito, para garantir que todos os indivíduos que participam do sistema de estágio possam ter um maior conhecimento sobre ele, facilitando o gerenciamento e o controle

do mesmo.

4.2.1 Gerenciamento do Projeto utilizando métodos ágeis

Como o aplicativo foi pensado como um sistema web dinâmico, o melhor meio para o desenvolvimento é a utilização dos métodos ágeis, pois garante uma maior interação com os usuários, gera pequenas versões regularmente e diminuiu na criação de uma excessiva documentação.

O processo de coleta de requisitos, diferente dos métodos tradicionais, foi feito durante todo o percurso de desenvolvimento do aplicativo. Nessa etapa, foram feitas reuniões em ciclos curtos de tempo onde era possível mostrar ao cliente uma pequena versão do aplicativo com algumas funções já em funcionamento. Na primeira reunião, foi feito um levantamento inicial de requisitos através do uso de estórias de usuários, nas reuniões posteriores, porém, somente foram feitas anotações de alterações pelo desenvolvedor do sistema.

4.2.1.1 Estória de Usuário

Como dito no capítulo de metodologia, o projeto foi desenvolvido aplicando os métodos ágeis, portanto o levantamento de requisitos se baseia em estórias de usuários. Nos métodos ágeis os requisitos não precisam ser escritos todos no começo do projeto, em cada encontro com o cliente pode-se alterar alguns requisitos ou criar novos requisitos.

A estória de usuário consiste em o usuário descrever as necessidades que ele gostaria de ver no sistema em pequenos textos, para esse levantamento será utilizados blocos de notas de papel, onde cada folha apresentava três itens: “Como, Quero e Para que possa”. O item “Como” refere-se à função do cliente no sistema, “Quero” refere-se ao requisito em si e “Para que possa” refere-se à necessidade que o requisito precisa atender. O uso de pequenos textos é necessário para evitar uma descrição extensa e pouco concisa, esse recurso facilita na hora de gerar versões rápidas do sistema.

Na tabela 1, temos os requisitos iniciais coletados no primeiro encontro com os clientes. Os usuários que participaram e escreveram as primeiras estórias foram: PRAE, Orientador do Estágio, o Supervisor da Empresa e o aluno.

Tabela 1: Tabela de Estória de Usuários Iniciais

REQUISITO	COMO	QUERO	PARA QUE POSSA
1	PRAE	Gerenciar Relatórios sem assinaturas	Alertar Empresa e Aluno
2	PRAE	Gerenciar Contratos na virada de semestre	O aluno, se necessário, mudar o horário no contrato de estágio
3	PRAE	Gerenciar Relatórios pendentes	Cobrar a assinatura dos envolvidos
4	PRAE	Consultar o resultado das avaliações dos orientadores e supervisores	Lançar no sistema acadêmico
5	PRAE	Cadastrar termo de estágio	Iniciar acompanhamento do estágio
6	PRAE	Cadastrar termo aditivo	Continuar com o acompanhamento do estágio
7	PRAE	Consultar a relação de todos os estagiários	Saber a situação atual do aluno e identificar eventuais problemas
8	Orientador	Consultar a relação dos meus orientados	Saber a situacao do estagiário e/ou obter informação do contrato
9	Orientador	Preencher a avaliação do relatório de estágio de meus orientaos	Compor os requisitos da avaliação final de estágio
10	Supervisor	Preencher a avaliação dos estagiários	Compor os requisitos exigidos para a avaliação final
11	Supervisor	Consultar a relação dos estagiários	Verificar a situação e/ou ver dados cadastrais
12	Aluno	Cadastrar um plano de estágio	Necessário para iniciar o estágio
13	Aluno	Preencher avaliação parcial do estágio	Relatório pertinente para o processo de estágio
14	Aluno	Consultar situação do estágio	Verificar se os meus relatórios estão corretos.

O recurso de estórias de usuários foi utilizado somente na primeira reunião com os usuários, a partir da segunda, foram feitas anotações pelo desenvolvedor derivadas das conversas feitas naquele encontro.

Na próxima seção, será detalhado como foi gerenciado o cronograma de reuniões.

4.2.1.2 Cronograma de Reuniões

Uma das ideias dos métodos ágeis é ter um contato regular com o cliente através de reuniões curtas e em pequenos períodos de intervalo (normalmente duas semanas), isso faz com o usuário possa visualizar como o projeto está se desenvolvendo, para que o desenvolvedor possa analisar se o que está sendo criado está de acordo com as necessidades propostas pelo cliente e que a interação entre os envolvidos possa gerar ou modificar os requisitos.

Nesse projeto, não foi possível criar um cronograma fixo. Houve dois aspectos que não permitiram tal ação: os clientes que testariam o sistema não estavam todo o tempo disponível para as reuniões e o desenvolvedor teve dificuldades na implementação de algumas funções não podendo gerar uma versão executável para o usuário. Porém, mesmo com os contratempos, houve 14 reuniões que definiram o andamento do projeto (Apêndice A).

4.2.2 Desenvolvimento do aplicativo web

Depois de concluído o gerenciamento do projeto, foi iniciada a implementação do sistema de controle e automatização de estágios para o Departamento de Informática da Universidade Federal do Paraná.

No quesito de controle de dados, a universidade possui um banco de dados que fornece algumas informações úteis para o processo de estágio, porém não contém todos os dados necessários. Devido a essa carência, foi construído um novo banco de dados que armazena essas informações e adiciona novas informações especificas do estágio.

Para integrar as práticas da programação extrema com a codificação do *software* utilizou-se o *framework* JSF, que é planejado para usar o modelo MVC, que auxilia na simplicidade e reusabilidade de código.

Na parte de testes, o sistema seguiu o padrão TDD, que diferente dos métodos tradicionais, consiste em criar primeiro o teste para depois gerar o código para ser aprovado por ele. Essa prática pode ser difícil de ser aplicada inicialmente, porém com o sistema seguindo os padrões XP pode ser útil para diminuir os custos em depuração do aplicativo.

Para demostrar o resultado do desenvolvimento do projeto, no final dessa subseção serão mostradas as páginas geradas para o sistema de controle e automatização de estágio que foram construídas seguindo os ideais dos métodos ágeis.

4.2.2.1 Banco de Dados

Para a construção do aplicativo foi necessário criar um novo banco de dados para gerenciar os dados que são fornecidos pela universidade com os dados criados unicamente pelo projeto. Devido ao fato do banco criado e do já existente serem distintos, todas as tabelas do aplicativo foram geradas do zero e apenas os dados foram exportados do sistema de banco de dados da UTFPR.

No total criaram-se 30 tabelas, sendo que 9 delas são apenas como suporte, ou seja, armazenam informações para ser adicionadas em outras tabelas (essas informações são recolhidas do sistema acadêmico). As outras 21 são tabelas editáveis e foram geradas especificamente para o aplicativo, sendo que 15 são a representação direta dos relatórios do processo de estágio e as 6 restantes são tabelas auxiliares para as tabelas dos relatórios.

O diagrama de entidade-relacionamento ilustrado na Figura 20 demonstra a estrutura das tabelas editáveis e os relacionamentos entre elas.

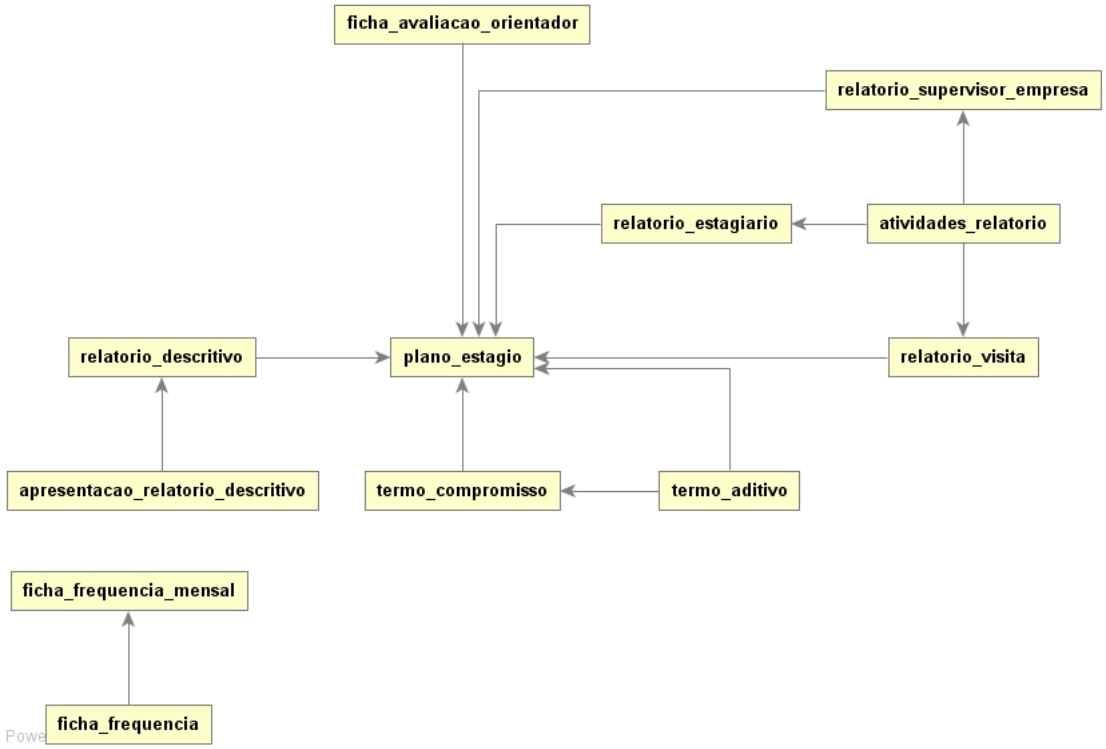


Figura 20: DER das tabelas dos relatórios do processo de estágio
Fonte: Autoria Própria

4.2.2.2 Gerenciamento de código implementado

Os métodos ágeis no desenvolvimento de aplicativos são práticas cujo objetivo é ajudar criar sistemas com codificação simples e limpa para que todos da equipe de programação possam alterar o projeto de maneira fácil. Portanto, o uso de padrões de projeto é de suma importância. No sistema implementado nesse projeto, foi utilizado o padrão MVC, recorrente em aplicativos web.

O padrão MVC é um modelo de arquitetura de *software* que separa a representação da informação da interação do usuário com ele. Divide-se em 3 componentes: o modelo, a visão e o controlador. O modelo consiste nos dados de aplicação, regras de negócio, lógica e funções. Uma visão pode ser qualquer saída de representação de dados, como uma interface ou uma página Web. O controlador faz a ligação entre o modelo e a visão.

Para o componente modelo, utilizou-se o *framework Hibernate* (subseção 3.2.6) que faz o mapeamento objeto-relacional, ou seja, mapeia as classes em *Java* para tabelas de dados (no caso desse projeto, tabelas em *SQL*). Isso possibilita o armazenamento, manipulação e geração de dados.

O MVC originalmente foi desenvolvido para computação pessoal, porém foi amplamente adaptado para aplicativos Web. Portanto, muitos *frameworks* foram criados tendo como base esse modelo. Nesse projeto, o *framework* JSF (subseção 3.2.4) foi escolhido para a implementação da interface com usuário integrado com *ICEFaces* (subseção 3.2.5).

Sendo o *ICEFaces* e o *Hibernate* desenvolvidos para trabalharem com *Java*, a parte de controle ficou responsável pelo uso dessa linguagem. Como o *Java* foi projetado para ser modelado para ser orientado a objeto, os *frameworks* de interface de usuário e de manipulação de dados funcionam através do uso de objetos que interligam essas duas camadas.

Esse projeto foi desenvolvido para a UTFPR que poderá manter o sistema futuramente, portanto para manter o código de fácil acesso, foi utilizado um sistema de controle de versionamento, o *Git* (subseção 3.2.9). Para que o código possa ser conduzido por outro(s) desenvolvedor (ers) em diferentes computadores, foi usada a ferramenta *Maven* (subseção 3.2.8) para automação de compilação que visa agrupar em um arquivo *POM*, as dependências e componentes externos do projeto, a ordem de compilação, diretórios e plugins necessários.

4.3 Resultados

Nessa seção serão abordados os resultados obtidos com a aplicação dos métodos ágeis num desenvolvimento de um aplicativo Web para controle e automatização de estágios.

Ao contrario dos métodos tradicionais de desenvolvimento de sistemas, a primeira tarefa ao programar um aplicativo é construir, a partir dos requisitos, os testes que serão posteriormente transformados em códigos em uma linguagem de programação, para isso nesse projeto foi utilizada a prática TDD.

A partir dos testes gerados, podemos, enfim, desenvolver o sistema em si que executa os requisitos propostos pelo cliente o mais fielmente possível.

4.3.1 Desenvolvimento Dirigido a Testes

Nessa subseção serão apresentados os resultados dos testes gerados com o uso da prática TDD.

A partir dos requisitos criados pelos usuários, foram desenvolvidas, com a ajuda da ferramenta *Cucumber*, as funcionalidades (*features*) que serviram como base no processo de codificação. Cada *feature*, normalmente, representa uma página do aplicativo Web e foi divida em

cenários (*Scenarios*) que identificam todos os módulos da página.

Muitos dos cenários gerados foram reutilizados em outras *features* apenas alterando os campos de entrada, portanto a seguir, serão descritos somente os principais cenários utilizados no desenvolvimento do aplicativo Web para controle e automatização de estágios.

4.3.1.1 Validar código errado

Uma das ideias do aplicativo Web é de facilitar o preenchimento dos relatórios de estágio ao digitalizar os documentos, para automatizar o processo, foi usado um banco de dados que contém informações previamente cadastradas que inserem os campos automaticamente, deixando apenas os campos que são exclusivamente do documento para o preenchimento do usuário.

Portanto, em alguns casos, é necessário fazer uma busca no sistema para que o aplicativo encontre as informações para o preenchimento automático, sendo assim, há uma necessidade da validação quando a busca não encontra os dados procurados.

Na figura 21, é ilustrado um cenário que valida a busca de um professor por um aluno ao cadastrar uma ficha de avaliação. O teste simula um professor orientador com o código de matrícula 37627, querendo cadastrar uma ficha de avaliação, mas para isso, é necessário preencher o nome do aluno a ser avaliado. Nesse cenário, o professor insere um nome que não está cadastrado no sistema, o que gera uma mensagem de validação ”código não encontrado”.

```
Scenario: Validar nome incorreto
  When Eu navego para a página "/restrito/cadastrar_ficha_avaliacao_orientador.xhtml?idProfessorOrientador=37627"
  And Eu preencho "buscar_nome_aluno" com "João da Silva Sauro"
  And Eu clico no botao "procurar_nome_aluno"
  And Eu poderia ver "Código não encontrado!"
```

Figura 21: Cenário de Teste: Validar código errado
Fonte: Autoria Própria

4.3.1.2 Validar os campos obrigatórios

Em cada relatório criado, há dados obrigatórios que dão credibilidade ao documento que comprovam a responsabilidade de cada envolvido no processo de estágio. Dessa forma, um dos requisitos é a uso da validação de campos obrigatórios que não permitem concluir o cadastro dos relatórios sem primeiro preencher todos os campos obrigatórios deste documento especifico.

A figura 22, demonstra o teste que simula o preenchimento do plano de estágio por um aluno que após selecionar o tipo de plano de estágio (0 Não Obrigatório; 1 Obrigatório) e o

curso no qual será feito o estágio, ele tenta salvar o plano sem preencher os dados obrigatórios, sendo validado com alertas na página.

```
Scenario: Validar os campos obrigatórios
    When Eu navego para a página "/restrito/cadastrar_plano_estagio.xhtml?idAluno=105113"
    And Eu clico no botao "tipo_plano_estagio:0"
    And Eu seleciono "SISTEMAS DE INFORMAÇÃO" na lista "lista_cursos"
    And Eu clico no botao "salvar_plano_estagio"
    Then Eu poderia ver "Campo Selecione o professor orientador! é obrigatório!"
    And Eu poderia ver "Campo Estágio é interno ou externo! é obrigatório!"
    And Eu poderia ver "Campo Nome do Supervisor é obrigatório!"
    And Eu poderia ver "Campo E-mail Supervisor é obrigatório!"
    And Eu poderia ver "Campo Data de início é obrigatório!"
    And Eu poderia ver "Campo Data de término é obrigatório!"
    And Eu poderia ver "Campo Atividades Estágio é obrigatório!"
```

Figura 22: Cenário de Teste: Validar campos obrigatórios
Fonte: Autoria Própria

4.3.1.3 Buscar por código de matrícula

Esse cenário é um caso especifico dos relatórios temo de compromisso e termo aditivo. Diferente dos outros relatórios, esses documentos são gerados pela empresa que contrata o aluno para estagiar, portanto o sistema não cria o relatório apenas o anexa no banco de dados.

O termo de compromisso é diretamente ligado ao plano de estágio, pois os dados gerados no plano de estágio devem ser exatamente iguais ao do termo. Já o termo aditivo é uma forma de extensão do termo de compromisso que permite alterar os dados gerados no termo de compromisso. Dessa forma, em ambos os casos, é necessário indexar o plano de estágio com os termos, para facilitar o cadastro, foi criada uma função que permite a busca do código do plano de estágio através do código de matrícula do estagiário.

Na figura 23, mostra como é feita a busca do plano de estágio através do código de matrícula. O professor responsável por estágio ao anexar o termo de compromisso, precisa adicionar o código do plano de estágio gerado pelo sistema, se o PRAE não possuir o código, é possível buscar o código clicando num *link*, que gera um *popup* que pede a inserção do código de matrícula do estagiário, ao procurar pela matricula, o sistema retorna o plano de estágio vigente do aluno, permitindo continuar o processo de cadastramento.

```
Scenario: Validar busca por código de matrícula
    When Eu navego para a página "/restrito/cadastrar_termo_compromisso.xhtml"
    And Eu clico no botao "form1:consultar_plano_estagio:link_busca_matricula"
    And Eu preencho "codigo_matricula" com "1051130"
    And Eu clico no botao "form1:consultar_plano_estagio:procurar_matricula"
    And Eu clico no botao "form1:consultar_plano_estagio:tabela_plano_estagio:0:link_codigo_plano_estagio"
    Then Eu poderia ver "75101873000866"
```

Figura 23: Cenário de Teste: Buscar por código de matrícula

Fonte: Autoria Própria

4.3.1.4 Cadastrar e alterar relatório

Todos os relatórios cadastráveis possuem um padrão, que facilita para o usuário o cadastramento de documentos distintos, sem ele se perder na hora do preenchimento dos campos.

O cadastro funciona da seguinte maneira: o sistema preenche os dados automaticamente quando possível, através de algum código dado pelo usuário ou pela identificação da pessoa que acessou a página, os dados obrigatórios devem ser preenchidos e ao final do preenchimento o relatório é salvo no banco de dados, ao clicar no botão de salvar o aplicativo cria o relatório em pdf e o mostra em um popup, se o usuário perceber que os dados que ele preencheu estão errados, ele tem a opção de altera-los. Se estiverem corretos, a página será redirecionada para a tela inicial do usuário.

Esse processo pode ser visto no cadastro de um relatório de visita (Figura 24).

```
Scenario: Cadastrar e Alterar Relatório Visita
  When Eu navego para a página "/restrito/cadastrar_relatorio_visita.xhtml"
  And Eu preencho "nome_aluno" com "Bruno Guilherme Andretta de Miranda"
  And Eu clico no botao "procurar_nome_aluno"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:0:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:1:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:2:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:3:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:4:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:5:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:6:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:7:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:8:resposta:0"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:9:resposta:0"
  And Eu preencho "descricao_empresa" com "Empresa de acordo com as normas da UTFPR."
  And Eu preencho "sugestoes_curso" com "Não há sugestões."
  And Eu clico no botao "salvar_relatorio_visita"
  And Eu clico no botao "alterar_relatorio_visita"
  And Eu clico no botao "atividades:tabela_atividades_relatorio:0:resposta:1"
  And Eu clico no botao "salvar_relatorio_visita"
  And Eu clico no botao "voltar_pagina_inicial"
  Then Eu poderia ver "sucesso!"
```

Figura 24: Cenário de Teste: Cadastrar e alterar relatório

Fonte: Autoria Própria

4.3.1.5 Validação após cadastro

Cada relatório no processo de estágio determina um período de início e termino, sendo que não é possível gerar outro documento do mesmo tipo enquanto este esteja em vigência.

Para validar essa etapa, o aplicativo Web bloqueia o acesso ao cadastro de um relatório enquanto este estiver ativo, e mostra uma mensagem para situar o usuário dessa exigência no período de estágio.

Um exemplo pode ser visto na figura 25, que não permite o cadastro de um novo plano de estágio enquanto este está em vigência. Então, quando o aluno tenta entrar na página de cadastro, é mostrado uma mensagem de validação.

```
Scenario: Validar se o estágio está em vigêcia
    When Eu navego para a página "/restrito/cadastrar_plano_estagio.xhtml?idAluno=105113"
    Then Eu poderia ver "Seu código já possui um estágio em vigência, favor encerrá-lo primeiro!"
```

Figura 25: Cenário de Teste: Validação após cadastro
Fonte: Autoria Própria

4.3.2 Páginas Desenvolvidas

Nessa subseção, apresentamos, após a análise dos requisitos dos usuários e a construção dos testes, os resultados em forma de páginas Web.

Como o sistema foi projetado para o uso de reutilização e padronização de código, as páginas possuem semelhanças entre si, portanto somente serão detalhadas as páginas que exemplificam as funções recorrentes usadas no aplicativo.

Para melhor entendimento, primeiramente será explicado como funciona a navegação das páginas, bem como um resumo de como cada página deve ser usada.

4.3.2.1 Navegação

O aplicativo Web foi planejado para ser usado por quatro usuários distintos: o aluno, o professor responsável pelo estagio, o professor orientador do aluno e o supervisor da empresa. Portanto, cada usuário terá acesso a uma tela inicial que contem as páginas que ele pode usar e orientações sobre qual é o seu papel no processo de estágio (A figura 26 mostra um exemplo de tela inicial de um aluno). Todas as páginas criadas tem um botão (”voltar”) que redireciona

para a tela inicial do seu respectivo usuário.

Figura 26: Tela inicial de um aluno
Fonte: Autoria Própria

A seguir, serão explicadas todas as páginas separadas por cada usuário.

O aluno é o primeiro envolvido no processo de estágio, pois após a conquista de um estágio, ele deve trazer os documentos iniciais para a universidade e será avaliado e irá avaliar as etapas provenientes das atividades realizadas na empresa. As páginas referentes a ele são:

- Cadastrar plano de estágio - Página de geração do plano de estágio para iniciar o processo de estágio
- Preencher relatório parcial - Cadastro do relatório necessário ao terminar um semestre de atividades ou na conclusão do estágio
- Anexar relatório descritivo - Anexar o relatório referente à conclusão da primeira etapa do estágio obrigatório (quando este possuir mais de uma etapa) ou na finalização do mesmo (quando este possuir apenas uma etapa). Também é possível visualizar um modelo do relatório (anexo B).

- Anexar apresentação do relatório descritivo Anexar à apresentação final do estágio obrigatório, anexá-lo após de anexar o relatório descritivo. Possibilidade de visualizar um modelo de apresentação, também.

- Ficha de frequência - adicionar diariamente o que foi realizado no estágio e a carga horária, no final do mês é possível gerar um relatório mensal da ficha (anexo F). Há a possibilidade de visualizar outros meses já concluídos.

- Visualizar situação - Consulta o andamento do processo de estágio, através dos relatórios gerados ou anexados ao sistema e também na consulta de estágio finalizados.

O professor responsável é o coordenador de toda a atividade de estágio que ocorre em um determinado curso de graduação. Sendo ele quem aprova e fiscaliza os relatórios durante esse processo. Em sequencia, as funções que ele pode exercer dentro do sistema:

- Cadastrar termo de compromisso - Cadastra um termo de compromisso de um determinado plano de estágio. Necessário cadastrar o plano de estágio primeiro.

- Cadastrar termo aditivo - Cadastramento de um termo aditivo de um determinado termo de compromisso previamente cadastrado.

- Avaliar Relatório do Supervisor da Empresa - Aprova um ou mais relatórios parciais do supervisor da empresa ou anexa um ou mais relatórios parciais aprovados.

- Atualizar Anexo - Anexar os relatórios que foram modificados após eles serem inseridos no sistema.

- Visualizar situação - Consulta todos os relatórios referentes ao processo de estágio com qual possui envolvimento, bem como é possível aprovar o relatório parcial do supervisor da empresa e finalizar um estágio vigente de um determinado aluno.

O professor orientador serve como mediador entre a universidade, o aluno e a empresa, sendo assim, ele fica no encargo de verificar se a empresa está de acordo com as exigências da UTFPR através de uma visita, bem como avaliar os relatórios parciais dos seus orientados e fazer uma avaliação do estágio quando este for um estágio obrigatório. A seguir, serão mostradas, resumidamente, as páginas de acesso do orientador.

- Cadastrar relatório de visita - Cadastrar um relatório de visita, que visa avaliar as condições da empresa ao prover um estágio adequado ao aluno que o professor orienta.

- Cadastrar ficha de avaliação - Preencher ficha necessária para aprovação do aluno no estágio obrigatório deve ser preenchida uma ficha para cada etapa do estágio obrigatório.
- Avaliar relatório parcial do aluno - Faz a aprovação de um ou mais relatórios parciais do aluno ou anexar um ou mais relatórios parciais aprovados.
- Visualizar situação - Consulta todos os relatórios com os quais possui ou possuía envolvimento, bem como a possibilidade de aprovar um ou mais relatórios parciais do aluno.

E, por fim, o supervisor da empresa também possui acesso ao sistema, porém por ser uma pessoa externa à universidade, ele tem poucas atribuições no processo de estágio, tendo apenas essas duas possibilidades no aplicativo:

- Cadastrar Relatório do Supervisor da Empresa - Cadastra o relatório no término de cada semestre de estágio ou na conclusão do mesmo.
- Visualizar situação do aluno - Consulta todos os relatórios dos estagiários que trabalhem para ele, bem como suas respectivas fichas de frequência.

As próximas seções, serão focadas apenas nas funções essenciais no aplicativo, para evitar o excesso de repetições de módulos.

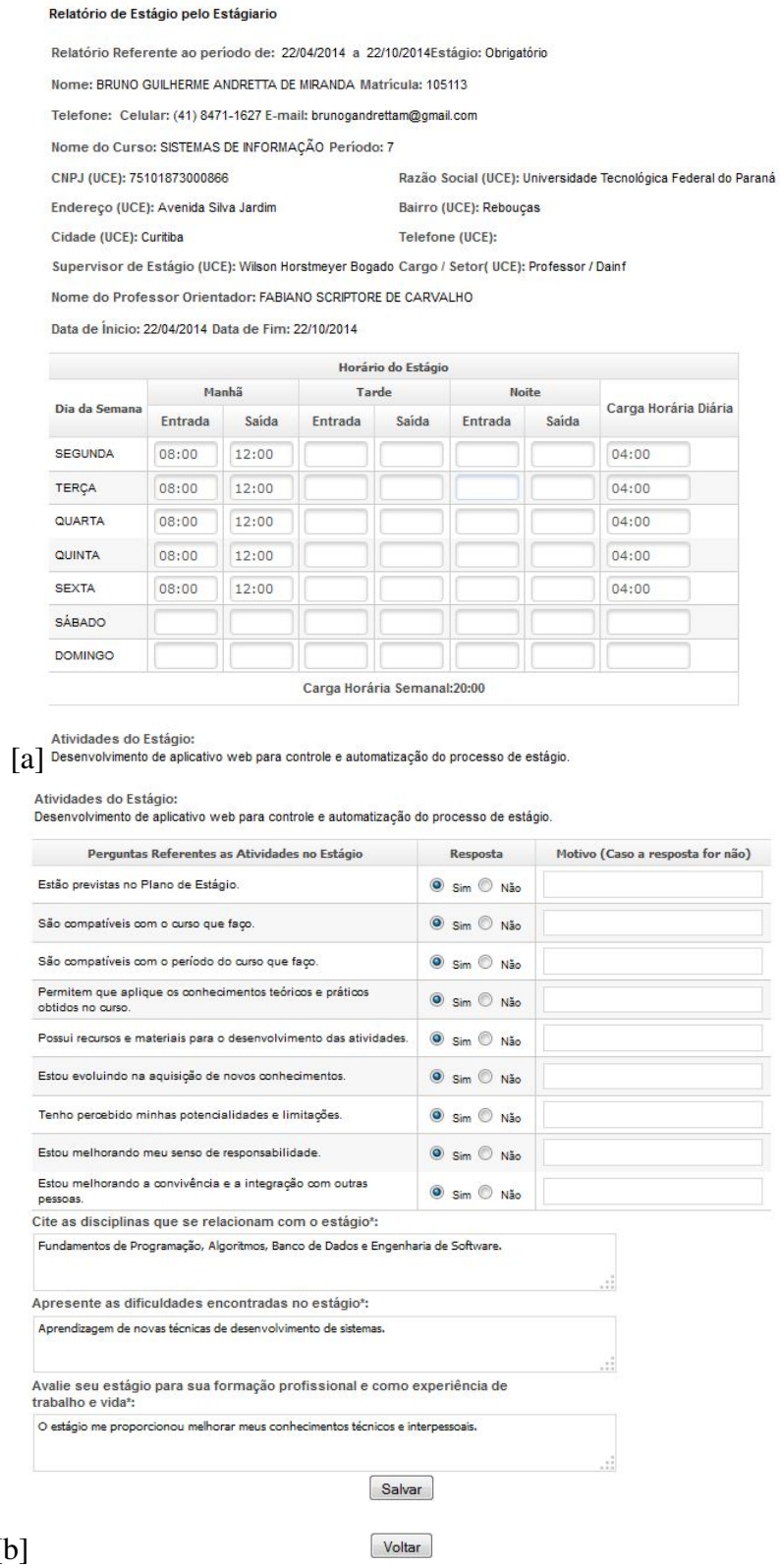
4.3.2.2 Página de cadastro

As páginas de cadastro simulam o preenchimento de relatórios que são feitos em papel, bem como gravam as informações no banco de dados, para futuras consultas e impressões. O primeiro documento a ser feito no processo de estágio é o plano de estágio que contém as dados sobre as pessoas envolvidas, sobre as atividades no estágio e sobre a duração do mesmo. A partir do plano, os outros relatórios são feitos. Para automatizar o processo de cadastramento, o sistema pega as informações previamente armazenadas no banco de dados e replica nos documentos, no caso dos relatórios sequentes ao plano de estagio, apenas são replicados os dados do próprio plano.

Como visto na seção 4.3.1.4, os cadastros de relatórios seguem o mesmo padrão, apenas o preenchimento do termo de compromisso e termo aditivo difere desse esquema, pois somente é anexado o relatório e não gerado como os demais.

Na figura 27 é ilustrado um exemplo de cadastro no sistema. O cadastro do relatório parcial do aluno é feito a cada seis meses ou na conclusão do estágio, para agilizar o processo, o

próprio aplicativo já fornece os dados que podem ser retirados do plano de estágio ao qual o relatório está vinculado. Os dados obrigatórios restantes devem ser preenchidos manualmente. Ao finalizar o preenchimento, deve clicar em salvar, ao fazer isso, será gerado um relatório em pdf que é mostrado em um *popup*, nele é possível voltar à página de cadastro e alterar alguma informação ou concluir o processo e voltar à página inicial do usuário.



Relatório de Estágio pelo Estágiario

Relatório Referente ao período de: 22/04/2014 a 22/10/2014Estágio: Obrigatório

Nome: BRUNO GUILHERME ANDRETTA DE MIRANDA Matrícula: 105113

Telefone: Celular: (41) 8471-1627 E-mail: brunogandrettam@gmail.com

Nome do Curso: SISTEMAS DE INFORMAÇÃO Período: 7

CNPJ (UCE): 75101873000866 Razão Social (UCE): Universidade Tecnológica Federal do Paraná

Endereço (UCE): Avenida Silva Jardim Bairro (UCE): Rebouças

Cidade (UCE): Curitiba Telefone (UCE):

Supervisor de Estágio (UCE): Wilson Horstmeyer Bogado Cargo / Setor(UCE): Professor / Dainf

Nome do Professor Orientador: FABIANO SCRIPTORE DE CARVALHO

Data de Início: 22/04/2014 Data de Fim: 22/10/2014

Horário do Estágio

Dia da Semana	Manhã		Tarde		Noite		Carga Horária Diária
	Entrada	Saída	Entrada	Saída	Entrada	Saída	
SEGUNDA	08:00	12:00					04:00
TERÇA	08:00	12:00					04:00
QUARTA	08:00	12:00					04:00
QUINTA	08:00	12:00					04:00
SEXTA	08:00	12:00					04:00
SÁBADO							
DOMINGO							

Carga Horária Semanal:20:00

Atividades do Estágio:
Desenvolvimento de aplicativo web para controle e automatização do processo de estágio.

[a]

Atividades do Estágio:
Desenvolvimento de aplicativo web para controle e automatização do processo de estágio.

Perguntas Referentes as Atividades no Estágio	Resposta	Motivo (Caso a resposta for não)
Estão previstas no Plano de Estágio.	Sim Não	
São compatíveis com o curso que faço.	Sim Não	
São compatíveis com o período do curso que faço.	Sim Não	
Permitem que aplique os conhecimentos teóricos e práticos obtidos no curso.	Sim Não	
Possui recursos e materiais para o desenvolvimento das atividades.	Sim Não	
Estou evoluindo na aquisição de novos conhecimentos.	Sim Não	
Tenho percebido minhas potencialidades e limitações.	Sim Não	
Estou melhorando meu senso de responsabilidade.	Sim Não	
Estou melhorando a convivência e a integração com outras pessoas.	Sim Não	

Cite as disciplinas que se relacionam com o estágio*:

Fundamentos de Programação, Algoritmos, Banco de Dados e Engenharia de Software.

Apresente as dificuldades encontradas no estágio*:

Aprendizagem de novas técnicas de desenvolvimento de sistemas.

Avalie seu estágio para sua formação profissional e como experiência de trabalho e vida*:

O estágio me proporcionou melhorar meus conhecimentos técnicos e interpessoais.

Salvar

Voltar

[b]

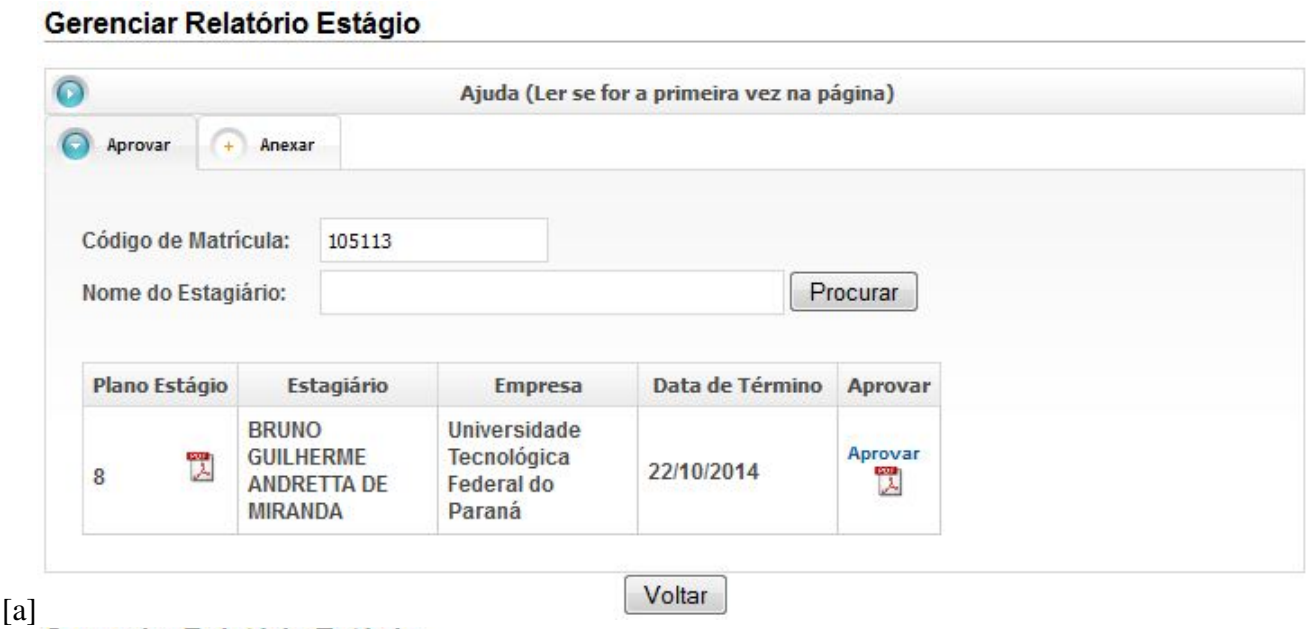
Figura 27: Na figura [a] temos a primeira parte do cadastro onde são replicadas as informações do plano de estágio vinculado. Já a figura [b] mostra os dados a serem preenchidos, bem como os botões de salvar (concluir ou alterar o cadastramento) e voltar(voltar a página inicial do usuário).

4.3.2.3 Página de aprovação de relatório parcial

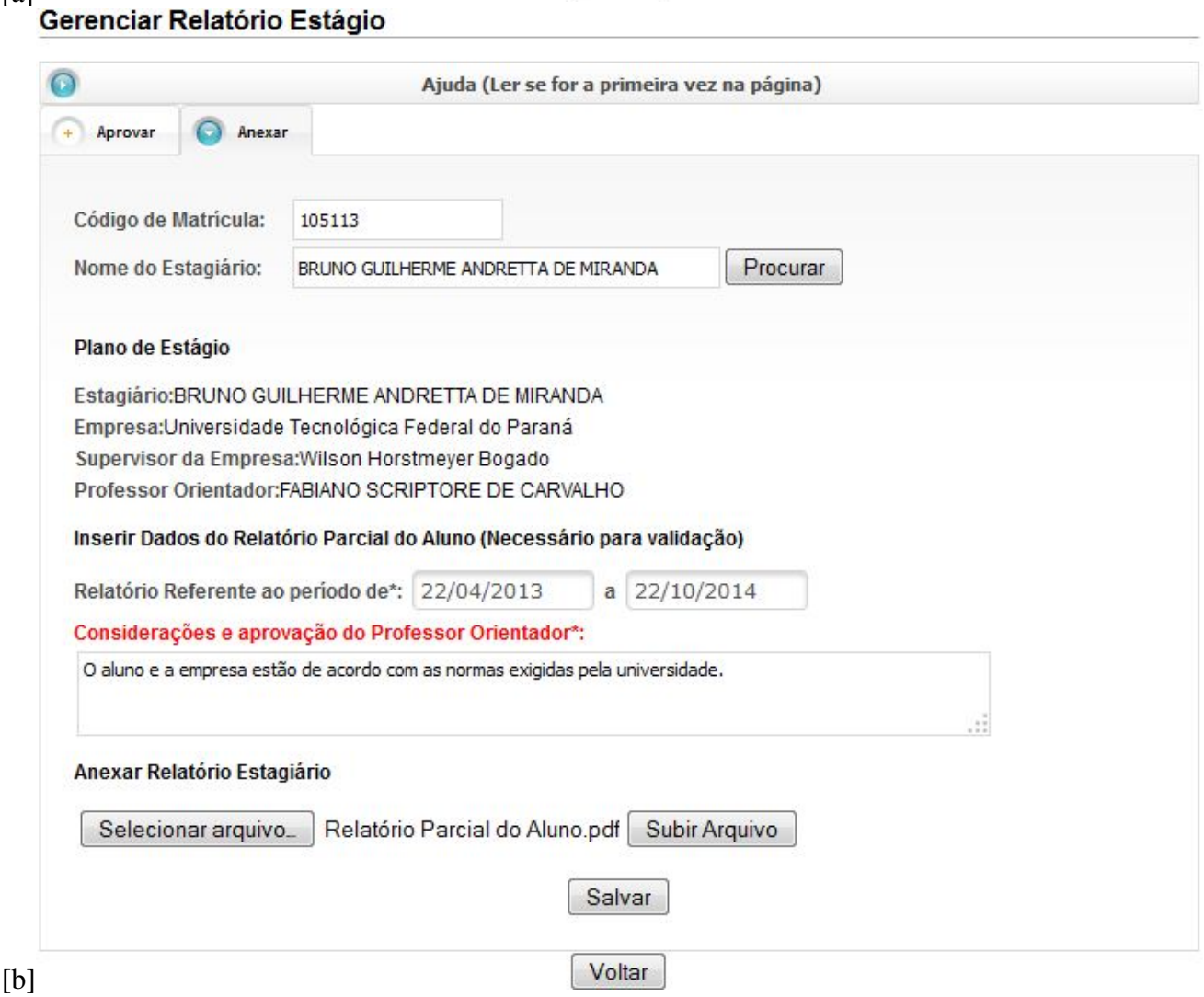
No processo de estágio, há dois relatórios parciais que servem para avaliar o desempenho do estágio através da visão do aluno (relatório parcial do aluno) e outro através da visão do supervisor da empresa (relatório parcial do supervisor da empresa).

Nesses dois casos os relatórios precisam passar por uma aprovação após serem escritos. Portanto, esses documentos possuem duas etapas no aplicativo. A primeira é fazer o preenchimento do relatório pelo aluno ou supervisor, ou seja, uma página de cadastro (mencionado na seção anterior). E a segunda parte, é a aprovação que pode ser feita de duas maneiras: aprovar o relatório pelo aplicativo ou anexar o relatório gerado externamente.

A figura 28 ilustra a página com o processo de aprovação que é feito pelo professor orientador no caso do relatório parcial do aluno e pelo professor responsável no caso do relatório parcial do supervisor da empresa. Na aba de aprovação, o usuário pode selecionar um relatório parcial a ser aprovado, após a seleção, ele é direcionado à página de cadastro do relatório, porém com um campo adicional onde ele deve colocar as suas considerações e aprovações (esse campo só é mostrado ao usuário que deve fazer a aprovação). Na aba anexar, o usuário deve anexar o relatório parcial gerado externamente.



[a]



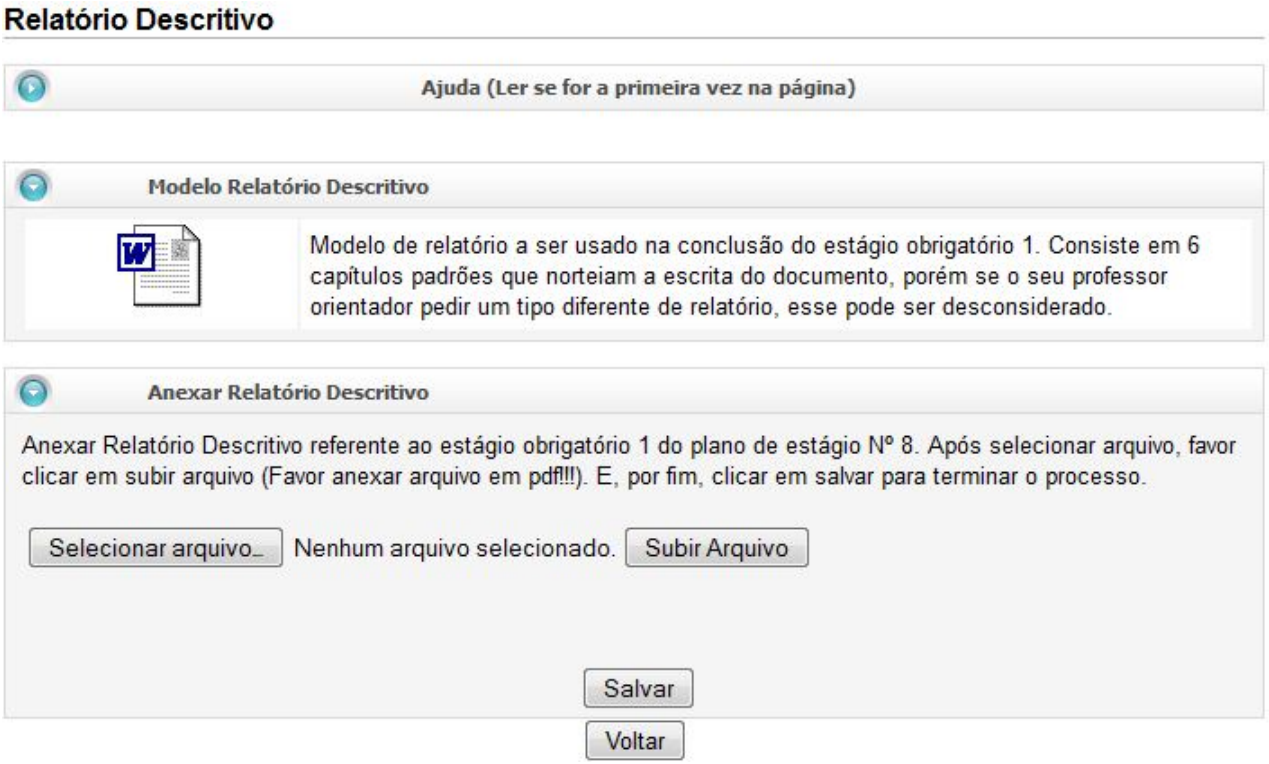
[b]

Figura 28: Na figura [a] temos a aba aprovar, onde o usuário pode selecionar um relatório para aprovação. Já a figura [b] mostra a aba anexar, necessária quando o relatório parcial é feito externamente ao sistema.

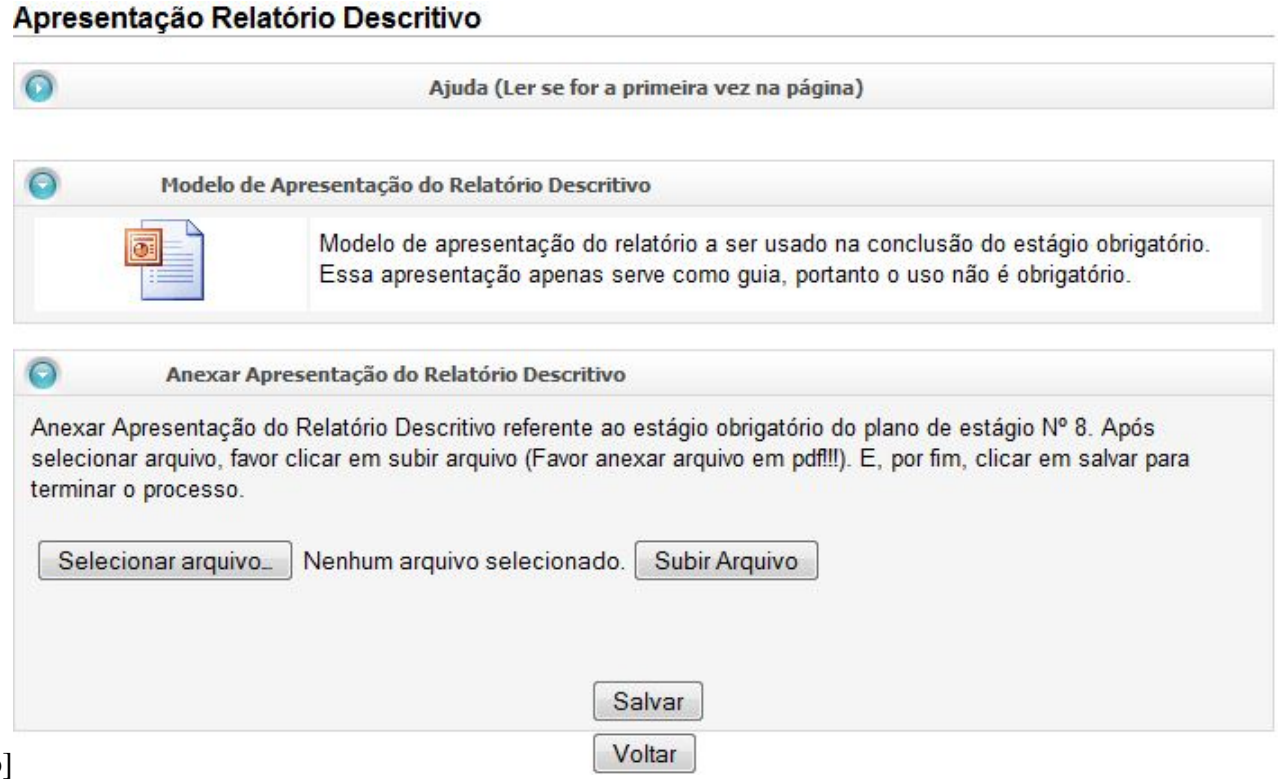
4.3.2.4 Páginas de validação de estágio obrigatório

Para concluir o estágio obrigatório, o estagiário deve escrever um artigo, denominado relatório descritivo, que descreve detalhadamente como são a empresa, as atividades e o envolvimento com as pessoas durante o período de estágio. Também é necessário fazer uma apresentação para uma banca com três professores para validar o estágio obrigatório.

Para garantir essa etapa, existe uma página no aplicativo que visa orientar como o aluno deve escrever esse relatório e a apresentação, através de um modelo para cada tipo que pode ser baixado no site. Depois de seguir o modelo, o aluno deve anexar o documento no sistema, para futuras consultas (figura 29).



[a]



[b]

Figura 29: Na figura [a] temos a página onde há o modelo para o relatório descritivo, bem como o botão para anexá-lo ao sistema (só é possível anexar quando o estágio é obrigatório).A figura [b] mostra a página onde deve ser anexada a apresentação do relatório descritivo (para poder anexar, primeiramente deve ser anexado o relatório descritivo).

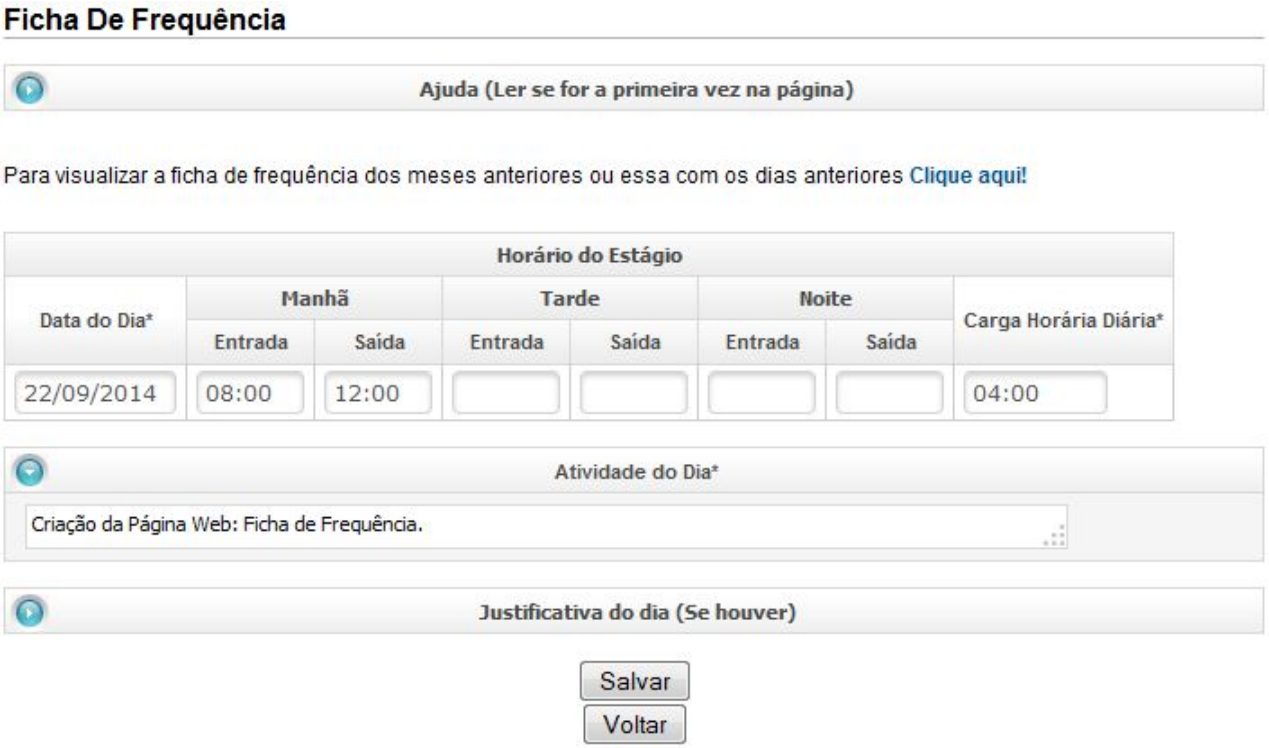
4.3.2.5 Página de ficha de frequência

A ficha de frequência serve para acompanhar diariamente a rotina do estagiário no trabalho. Na ficha gerada em papel, somente são vistos os dias em que o aluno foi ao estágio sem ter informações de quais atividades ele executou. Para suprimir essa necessidade, o aplicativo web tem uma página para gerar a ficha digitalmente onde o aluno, além de colocar as horas diárias trabalhadas, ele informa o que ele fez naquele dia, permitindo ter informações mais detalhadas do dia-a-dia do estagiário.

Essa função visa melhorar o preenchimento de relatórios de desempenho do estágio, através do enriquecimento de informações geradas na ficha de frequência. Como também, permite um controle maior sobre o estagiário e sobre suas atitudes no processo de estágio.

Uma representação das páginas de ficha de frequência pode ser vista na figura 30. O aplicativo Web possui duas páginas referentes à ficha de frequência, uma para adicionar diariamente as horas e as atividades trabalhadas naquele dia (nessa página há um link para a segunda), e uma segunda para gerar a ficha de frequência no final do mês ou visualizar as fichas de frequências anteriores.

[a]



[b]

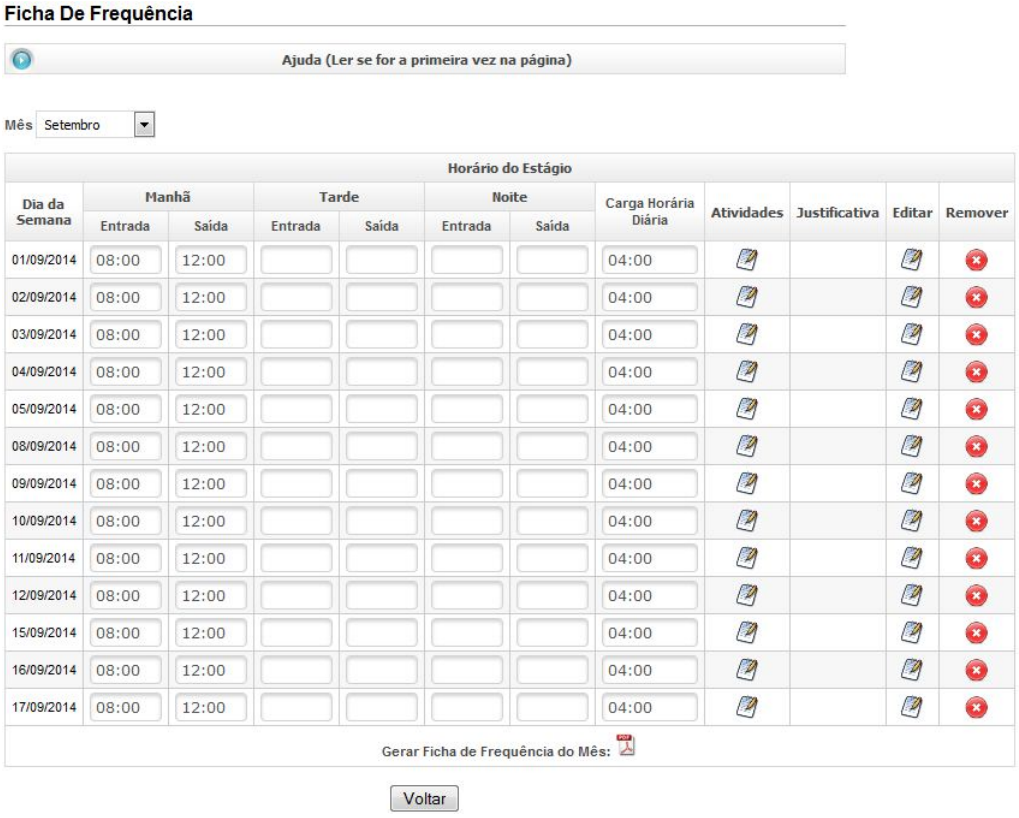


Figura 30: Na figura [a] é ilustrado o cadastro diário das horas e atividades exercidas num dia de estágio.A figura [b] mostra a página onde é possível gerar a ficha mensal ou visualizar as fichas anteriores.

4.3.2.6 Página de consulta

Ter uma página de consulta foi um dos requisitos mais explorados pelos clientes, pois no processo atual de estágio, os relatórios são gerados em papel, cada pessoa envolvida fica com uma cópia, porém esses documentos são engavetados e esquecidos. No aplicativo Web, é possível visualizar todos os relatórios gravados no sistema, porém há uma validação que só permite a consulta de relatórios vinculados ao código de acesso do usuário e ao tipo de usuário.

Outro aspecto útil da página, é que ela informa o andamento dos processos dos relatórios. O usuário pode ver quando um relatório já foi feito, se ele precisa ser aprovado (é possível aprovar pela página de consulta, se o usuário tiver essa permissão) e a situação do estágio, se está em andamento ou já foi concluído. Se o usuário for um professor responsável, ele poderá finalizar o processo de estágio. Na figura 31 é demonstrada uma página de consulta vista por um professor responsável.

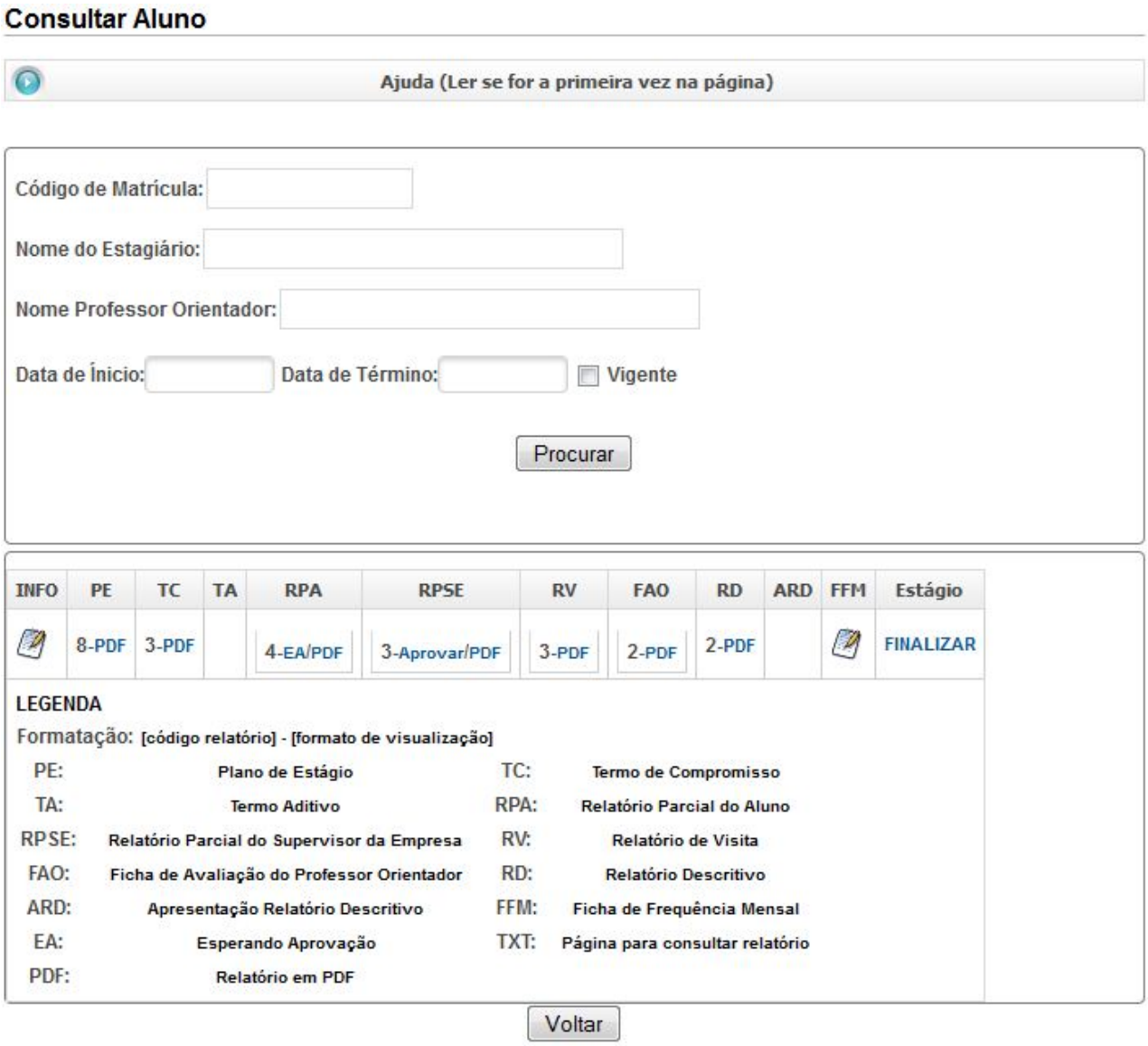


Figura 31: Exemplo de uma página de consulta

Fonte: Autoria Própria

5 CONCLUSÕES

Nesse trabalho de conclusão de curso foi apresentando o desenvolvimento de um aplicativo web para auxiliar no processo de estágio no departamento de informática na UTFPR. A criação do *software* veio da necessidade de um melhor controle e na automatização do gerenciamento de estágios. O sistema atual de estágio utiliza o preenchimento de relatórios em papéis para fazer o acompanhamento dos alunos nas empresas. Porém essa prática é pouca eficaz, pois há um grande intervalo entre as entregas dos relatórios o que afeta na verificação do desempenho do estágio. Outro aspecto que prejudica o processo é a falta de informação centralizada, que faz com que os alunos, até mesmo os professores e supervisores, se percam no controle dos documentos e nas datas de entregas.

O desenvolvimento do aplicativo visa suprir as falhas causadas pelo processo de estágio usado atualmente. Para isso, ele agrupa todas as etapas do processo ao digitalizar todos os documentos para serem preenchidos online. O site guarda informação sobre o estágio de forma centralizada e, em cada página, há um campo de ajuda que explica o que é feito naquela fase do processo. Cada usuário tem acesso a uma página exclusiva, sendo assim, cada pessoa envolvida pode saber qual é a sua função em cada etapa do processo de estágio. O aplicativo permite, também, a consulta dos relatórios guardados no sistema, portanto garante um controle mais eficiente pelos professores e supervisores sobre seus estagiários, nessa página cada usuário terá uma visão especifica que visa facilitar no gerenciamento e mostrar o andamento dos relatórios ao qual ele está ligado.

Para o gerenciamento do projeto é usado algumas práticas da programação extrema. O uso dos métodos ágeis é útil, pois facilita a interação com os clientes gerando requisitos mutáveis, sendo estes, o mais próximo das necessidades reais. Por ter pouca documentação, o foco no desenvolvimento é ampliado, o que facilita na demonstração do sistema para os clientes. Como o aplicativo foi desenvolvido por apenas um programador, algumas práticas da programação extrema não foram utilizadas, porém mesmo com essa limitação o uso dessa metodologia é mais indicado.

No desenvolvimento houve algumas dificuldades que serão explicadas a seguir. Como os métodos tradicionais são mais recorrentes na programação de um *software*, a mudança para outro paradigma afetou, inicialmente, alguns fatores no gerenciamento do projeto. Na programação extrema existe uma prática que visa ter reuniões cada duas semanas com o usuário para mostrar uma pequena versão do sistema, a fim de gerar ou alterar novos requisitos, nesse projeto, não foi possível garantir sempre o ciclo de duas semanas, pois houve um empecilho inicial com as ferramentas usadas no desenvolvimento, no próprio uso das práticas XP e, algumas vezes, até a disponibilidade dos clientes, mas isso não afetou na interação constante com os usuários. Outro ponto que proporcionou uma dificuldade foi a prática de codificar os testes antes de programar, inicialmente, houve uma estranheza com o seu uso, precisando gerar alguns testes depois da programação, porém com o tempo e a habilidade com o uso das ferramentas usadas no desenvolvimento, foi possível aplicar o TDD de forma completa.

Apesar dos contratempos, foi possível criar um sistema que atendeu as necessidades propostas pelos clientes. O sucesso na realização deste projeto somente foi possível com a utilização dos conhecimentos adquiridos no decorrer do curso de Bacharelado em Sistemas de Informação. Esse trabalho pode ser útil para outros estudantes que desenvolvam projetos com métodos ágeis e no uso da prática de TDD, pois mostra um exemplo concreto de desenvolvimento de *software* que soluciona um problema do mundo real.

5.1 Trabalhos Futuros

O *software* apresentando neste trabalho de conclusão de curso ainda não é o sistema ideal a ser aplicado no mundo real, e diversas melhorias podem ser realizadas para aumentar sua qualidade e eficiência.

- A primeira delas seria integrar totalmente os usuários no sistema, para que todos, a partir do seu código de identificação, possam acessar sua página exclusiva e interagir com o que o aplicativo possa oferecer.

- Há a necessidade de adquirir as informações das empresas e agentes de integração que estão aptos para garantir estágios, através da exportação das tabelas do banco de dados da universidade. Isso permite proporcionar mais dados na hora de preencher os relatórios online.

- Cada semestre letivo, os dados dos alunos são alterados no sistema acadêmico, isso afeta na hora de gerenciar o estágio, portanto é preciso atualizar o banco de dados periodica-

mente para sempre ter os dados atualizados.

- Quando o aluno faz estágio na própria universidade e está para conclui-lo deve ser feito um requerimento de conclusão, que é um relatório que formaliza o término do estágio. Essa função não foi aplicada no aplicativo Web.

- Para fazer um gerenciamento melhor das etapas no processo de estágio seria útil ter um gerenciamento de e-mails, cujo objetivo seria avisar às pessoas envolvidas quando um relatório foi gerado, quando um relatório está pendente ou sobre as datas de entrega dos documentos.

- E, por fim, o sistema foi testado apenas num servidor local, portanto para garantir a eficiência do aplicativo seria ideal colocá-lo online e ver como ele funciona com os usuários reais.

Referências

ABRAHAMSSON, P. et al. **Agile Software Development Methods. Review adn Analysis**. [S.l.]: Espoo, 2002.

ALGAWORKS. Desenvolvimento web com javaserver faces. Setembro 2010. Disponível em: <http://www.algaworks.com/downloads/apostilas/algaworks-dwjsf-desenvolvimento-web-com-javaserver-faces-2a-edicao.pdf>.

BECK, K. **Extreme Programming Explained: Embrace Change**. [S.l.]: Addison-Wesley, 1999.

BECK, K. **Test Driven Development by Example**. [S.l.]: Addison-Wesley, 2002.

BECK, K. et al. Manifesto for agile software development. 2001. Disponível em: <http://www.agilemanifesto.org>.

BOEHM, B. W.; GROUP, T. D. S. A spiral model of software development and enhancement. 1988.

COCKBURN, A.; HIGHSMITH, J. Agile software development: The business of innovation. **IEEE Computer**, 2001.

DIAS, M. V. B. **Um Novo Enfoque para o Gerenciamento de Projetos de Desenvolvimento de Software**. Dissertação (Mestrado) — Universidade de São Paulo, 2005.

DIEEM. Ficha de frequência. Setembro 2014. Disponível em: <http://www.utfpr.edu.br/servidores/estagiointerno/EstagioFichadefrequencia2setembro2012.pdf>.

DIEEM. Plano de estágio. Setembro 2014. Disponível em: <http://estagio.utfpr.edu.br/arquivos/plano_de_estagio.pdf>.

DIEEM. Regulamento de estágio utfpr. Setembro 2014. Disponível em: <http://www.utfpr.edu.br/servidores/estagio-interno/Regulamento_Estagio_UTFPR.pdf>.

DIEEM. Relatório de visita do professor orientador. Setembro 2014. Disponível em: <http://estagio.utfpr.edu.br/arquivos/RelatoriodeVisitaaUnidadeConcedentedeEstagio.pdf>.

DIEEM. Relatório parcial do aluno. Setembro 2014. Disponível em: <http://estagio.utfpr.edu.br/arquivos/RelatorioParcialdeEstagio.pdf>.

DIEEM. Relatório parcial do supervisor da empresa. Setembro 2014. Disponível em: <http://estagio.utfpr.edu.br/arquivos/RelatorioParcialdeSupervisaodeEstagio.pdf>.

FILHO, E. G. da C. et al. Padrões e métodos Ágeis: agilidade no processo de desenvolvimento de software. 2005.

IT improve. Scrum: metodologia ágil para gestão e planejamento de projetos. Acessado em Março de 2013. 2009. Disponível em: <http://www.improveit.com.br/scrum>.

K19. Relatórios em java jasperreports e ireport. Setembro 2010. Disponível em: <http://www.k19.com.br/artigos/relatorios-em-java-jasperreports-e-irepor/>.

MAINART, D. de A.; SANTOS, C. M. Desenvolvimento de software: Processos Ágeis ou tradicionais? uma visão crítica. 2010.

NETO, G. U. M. **Métodos Tradicionais versus Ágeis: Um estudo comparativo através do trainingcad**. Dissertação (Mestrado) — Faculdade de Ciência e Tecnologia de Caruaru, 2009.

PRANGE, H. F. **Uma Avaliação Empírica de um Ambiente Favorável para o Desenvolvimento Dirigido por Testes**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2007.

PRESSMAN, R. S. **Engenharia de Software**. [S.l.]: MAKRON Books, 1995.

ROCHA, T. Águila da; OLIVEIRA, S. R. B.; VASCONCELOS, A. M. L. Adequação de processos para fábricas de soft. In: **VI Simpósio Internacional de Melhoria de Processos de Software**. [S.l.: s.n.], 2004.

SANTOS, R. L. **Emprego de Test Driven Development no desenvolvimento de aplicações**. Dissertação (Mestrado) — Universidade de Brasília, 2010.

SCHWABER, K. Agile project management with scrum. **Microsoft Press**, 2004.

SILVA, E. R. P. da. **Desenvolvimento Dirigido por Testes**. Dissertação (Mestrado) — Faculdade de Tecnologia de Sõ Paulo, 2012.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.: s.n.], 2003.

UTIDA, K. H. **Metodologias Tradicionais e Metodologias Ágeis: Análise Comparativa entre rational unified process e extreme programming**. Dissertação (Mestrado) — Faculdade de Tecnologia do Estado de São Paulo, 2012.

APÊNDICE A – Cronograma de Reuniões

A tabela abaixo mostra as reuniões feitas durante o processo de desenvolvimento do aplicativo Web para controle e automatização de estágios.

Tabela 2: Cronograma de Reuniões

REUNIÃO	ASSUNTOS ABORDADOS
1	•Definição do Sistema •Estórias de Usuário
2	•Estrutura do banco de dados •Criação da página de cadastro de termo de compromisso
3	•Estrutura do processo de estágio
4	•E-mail do supervisor da empresa é obrigatório ao cadastrar plano de estágio •Mostrar o ramal do professor orientador em cadastrar plano de estágio •Na página de cadastro de termo de compromisso, possibilidade de buscar o código do plano de estágio através da matrícula do aluno •Na página de termo aditivo, possibilitar busca do código de termo de compromisso através da matrícula do aluno

Continua na próxima página

REUNIÃO	ASSUNTOS ABORDADOS
5	•Para conclusão de um estágio obrigatório, o sistema deve permitir o preenchimento do relatório descritivo pelo aluno e da ficha de avaliação pelo professor orientador, além do relatório parcial do supervisor •A última etapa do estágio obrigatório, o sistema deve permitir que o aluno anexe uma apresentação cujo objetivo é avaliar o estágio perante uma banca com 3 professores
6	•O Professor orientador pode consultar todos os seus orientados, sendo possível escolher determinado aluno por filtros na consulta •O PRAE pode fazer uma consulta de estágios filtrando por professores orientadores •Cada usuário deve ter uma página inicial com os links das páginas condizentes a ele •O sistema deve permitir a criação de relatórios em pdf, sendo que os únicos a serem anexados são termo de compromisso e aditivo •Cada usuário terá acesso a sua página através de um login e senha
7	•Ao cadastrar o plano de estágio, identificá-lo somente como obrigatório ou não obrigatório, o código da disciplina fica somente nos relatórios parciais •Possibilitar alteração dos relatórios durante o cadastramento
8	•Ao gerar os relatórios em pdf, retirar linha abaixo dos campos de assinatura e colocar os nomes de cada assinante •Possibilitar duas maneiras de consultar um relatório: em arquivo pdf ou na própria página do sistema
9	•Criar página para poder anexar relatórios assinados

Continua na próxima página

REUNIÃO	ASSUNTOS ABORDADOS
10	•Criar ajuda em todas as páginas e nos campos preenchíveis colocar títulos •Criar paginação nas tabelas de consulta
11	•Possibilidade de anexar relatórios gerados externamente •Criar página de ficha de frequência
12	•Gerar um plano de estágio por empresa •Criar as seguintes funções: cadastrar ficha de avaliação do professor orientador, anexar relatório descritivo e anexar apresentação do relatório descritivo
13	•Lapidar as funções criadas até agora
14	•Finalização das funções do site •Revisão de todas funções criadas

APÊNDICE B – Relatório Descritivo

Relatório criado para servir como modelo ao escrever o relatório Descritivo.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO XXX
CURSO DE XXX

NOME ESTAGIÁRIO

RELATÓRIO ESTÁGIO OBRIGATÓRIO 1

CURITIBA 201X

SUMÁRIO

CAPÍTULO 1 – PLANO DE ESTÁGIO

Capítulo introdutório do relatório descritivo, portanto deverá conter uma breve descrição do estágio, os principais envolvidos nesse processo e os objetivos do estágio. Apontar também, os principais conhecimentos adquiridos ou utilizados no desenvolvimento das atividades e escrever um curto levantamento bibliográfico sobre esses.

CAPÍTULO 2 – ORGANOGRAMA DA EMPRESA

Escrever sobre a empresa em si, sua hierarquia, os processos utilizados e, em especifico, os utilizados na área onde ocorre o estágio. Se possível, anexar uma imagem para facilitar a compreensão do organograma.

CAPÍTULO 3 – RECURSOS DISPONÍVEIS PARA O ESTÁGIO

Mostrar todos os recursos utilizados no desenvolvimento das atividades no estágio, sendo este hardware ou software.

CAPÍTULO 4 – ATRIBUÍCOES NO ESTÁGIO

Escrever todas as atribuições no estágio, bem como a responsabilidade dessas atribuições na empresa e quais as pessoas envolvidas e que te auxiliam nas realizações das atribuições.

CAPÍTULO 5 – ATIVIDADES DESENVOLVIDAS

Detalhar todas as atividades desenvolvidas no período do estágio obrigatório, quais os processos, recursos e pessoas envolvidas nessas atividades. Criar Diagrama de Gantt para gerar o cronograma de atividades.

CAPÍTULO 6 – COMENTÁRIOS SOBRE ESTÁGIO

Comentar sobre as tarefas e processos da empresa, bem como seu envolvimento com eles, as atividades desenvolvidas e como o estágio te ajudou no seu crescimento profissional.

CAPÍTULO 7 - CONCLUSÕES

Concluir escrevendo sobre a experiência adquirida no estágio, bem como o seu desenvolvimento profissional e social.

ANEXO A – Plano de Estágio

Relatório de Plano de Estágio utilizado na UTFPR (DIEEM, 2014b).

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA

Plano de Estágio

☐ Obrigatório ☐ Não Obrigatório

Nome do Estagiário(a):			Código de Matrícula:
Curso:	Código de Disciplina de Estágio:	Ano:	Período:
Telefone:	Celular:	E-mail:	
Nome da Empresa:			
Nome do Supervisor de Estágio da Empresa:			
Cargo / Setor:	Telefone:	Celular:	
E-mail:			
Horário do Estágio:		Horas Semanais:	
Data de Início:	Data de Término:		

Atividades a serem desenvolvidas pelo estagiário:
(a ser preenchido em conjunto com o Supervisor da Empresa)

Professor Orientador do Estágio

Nome:			Departamento
Telefone:	Celular:	E-mail:	
( ) Estagiário(a)	( ) Funcionário(a)	( ) Empresário(a)	( ) Autônomo(a)
Número do Termo de Compromisso: _____/_____			

Aprovação *(para efeito de carga horária da disciplina de estágio obrigatório)*

Contrato de Estágio assinado apresentado em : ___/___/___.

Data de Início contagem carga horária: ___/___/___.

Data de Término previsto carga horária: ___/___/___.

Estagiário	Supervisor de Estágio da Empresa (Assinatura e Carimbo)	Prof. Responsável pela atividade de Estágio (Assinatura e Carimbo)	Prof. Orientador do Estágio (Assinatura e Carimbo)

Curitiba, ___/___/___

Emitir em três vias

ANEXO B – Relatório de Visita do Professor Orientador

Relatório de Visita do Professor Orientador utilizado na UTFPR (DIEEM, 2014d)

Relatório Parcial de Supervisão de Estágio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba
Departamento Acadêmico de Informática

RELATÓRIO DE VISITA A UNIDADE CONCEDENTE DE ESTÁGIO

( ) Estágio Obrigatório ( ) Estágio Não Obrigatório

Unidade Concedente de Estágio (UCE): _____

Endereço: _____

Bairro: _____ Cidade: _____ Telefone: _____

Supervisor de Estágio na UCE: _____

Estagiário: _____ Código: _____

E-mail: _____ Telefone: _____

Curso: _____ Período: _____

Vigência do Estágio: _____/____/____ a ____/____/____ Horário do Estágio: _____

Professor: _____

O professor deve realizar reunião com o Supervisor de Estágio e o Estagiário na UCE para subsidiar o preenchimento deste relatório.

As atividades realizadas pelo estagiário:	SIM	NÃO
a) São compatíveis com o curso.		
b) Estão previstas no Plano de Estágio.		
c) Permitem que aplique os conhecimentos teóricos e práticos obtidos no curso.		
d) Permitem a aquisição de novos conhecimentos.		
e) Satisfazem as expectativas da UCE.		
O ambiente em que estão sendo desenvolvidas as atividades de estágio:		
a) Possui recursos e materiais para o desenvolvimento das atividades.		
O Supervisor de Estágio:		
a) Acompanha as atividades realizadas pelo estagiário.		
b) Auxilia o estagiário na solução de problemas ou dificuldades.		
O Estágio pode continuar:		
a) Sem modificação nas atividades previstas no Plano de Estágio.		
b) Pois o ambiente fornece condições para o desenvolvimento das atividades.		

Quando assinalado NÃO, apresente os motivos: _____

_____

_____

Descrever o ambiente e discorrer, minuciosamente, sobre as atividades desempenhadas pelo estagiário.

_____

_____

_____

Apêndice A da Instrução Normativa Conjunta 03/2011 – PROGRAD/PROREC 1

Continuação da descrição do ambiente e das atividades desempenhadas pelo estagiário.

Opcional: Apresentar sugestões para modificações no curso em função da visita realizada.

Local e Data

Estagiário	Supervisor de Estágio	Professor

Caso a UCE possua mais de um estagiário em determinada área, o professor, em uma única visita, poderá verificar as atividades realizadas por todos os estagiários O professor deve preencher um Relatório de Visita para cada estagiário. Após a visita, o relatório deve ser entregue para o Professor Responsável pela Atividade de Estágio para lançamento desta informação no Sistema Integrado de Estágio.

ANEXO C – Relatório Parcial do Aluno

Relatório Parcial do Aluno utilizado na UTFPR (DIEEM, 2014e)

Relatório Parcial de Supervisão de Estágio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba
Departamento Acadêmico de Informática

RELATÓRIO PARCIAL DE ESTÁGIO

( ) Estágio Obrigatório ( ) Estágio Não Obrigatório ( ) 1º ( ) 2º ( ) 3º ( ) 4º Relatório

Relatório referente ao período de: _____/____/____ a ____/____/____ (máximo 6 meses)
Estagiário: _____ Código: _____
E-mail: _____ Telefone: _____
Curso: _____ Período:_____
Unidade Concedente de Estágio (UCE): _____
Endereço: _____
Bairro: _____ Cidade: _____ Telefone: _____
Supervisor de Estágio na UCE: _____
Vigência do Estágio: _____/____/____ a ____/____/____ Horário do Estágio: _____
Professor Orientador na UTFPR: _____

Descrever as atividades desenvolvidas no estágio: _____

As atividades realizadas no estágio:	SIM	NÃO
a) Estão previstas no Plano de Estágio.		
b) São compatíveis com o curso que faço.		
c) São compatíveis com o período do curso que faço.		
d) Permitem que aplique os conhecimentos teóricos e práticos obtidos no curso.		
O ambiente em que estão sendo desenvolvidas as atividades de estágio:		
a) Possui recursos e materiais para o desenvolvimento das atividades.		
Conhecimentos para a formação profissional:		
a) Estou evoluindo na aquisição de novos conhecimentos.		
b) Tenho percebido minhas potencialidades e limitações.		
Relacionamento e Sociabilidade:		
a) Estou melhorando meu senso de responsabilidade.		
b) Estou melhorando a convivência e a integração com outras pessoas.		

Apêndice C da Instrução Normativa Conjunta 03/2011 – PROGRAD/PROREC 1

Quando assinalado NÃO, apresente os motivos: _____

Cite as disciplinas que se relacionam com o estágio: _____

Apresente as dificuldades encontradas no estágio: _____

Avalie seu estágio para sua formação profissional e como experiência de trabalho e vida: _____

Data: ____/____/____ _____ _____
Estagiário Supervisor de Estágio

Considerações e aprovação do Professor Orientador: _____

Data: ____/____/____ _____
Professor Orientador

A cada 6 (seis) meses, o estagiário deve entregar o Relatório Parcial de Estágio para o Professor Orientador que, após, entregará para o Professor Responsável pela Atividade de Estágio para lançamento desta informação no Sistema Inte

ANEXO D – Relatório Parcial do Supervisor da Empresa

Relatório Parcial do Supervisor da Empresa utilizado na UTFPR (DIEEM, 2014f)

Relatório Parcial de Supervisão de Estágio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba
Departamento Acadêmico de Informática

RELATÓRIO PARCIAL DE SUPERVISÃO DE ESTÁGIO

( ) Estágio Obrigatório ( ) Estágio Não Obrigatório ( ) 1º ( ) 2º ( ) 3º ( ) 4º Relatório

Relatório referente ao período de: _____/____/____ a ____/____/____ (máximo 6 meses)

Unidade Concedente de Estágio (UCE): _____

Endereço: _____

Bairro: _____ Cidade: _____ Telefone: _____

Supervisor de Estágio na UCE: _____

Curso de formação do Supervisor de Estágio: _____

Vigência do Estágio: _____/____/____ a ____/____/____ Horário do Estágio: _____

Área do Estágio: _____

Estagiário: _____ Código: _____

E-mail: _____ Telefone: _____

Curso: _____ Período: _____

Professor Orientador na UTFPR: _____

Avaliação do Estagiário
Assinale com "X" o desempenho do estagiário, utilizando:
A = acima da expectativa, **B** = de acordo com a expectativa, **C** = abaixo da expectativa, **D** = não se aplica

	A	B	C	D
Aplicação de Conhecimentos *Aplica os conhecimentos adquiridos no curso no desenvolvimento das atividades.*				
Autocrítica *Demonstra capacidade de reconhecer suas dificuldades e erros.*				
Autodesenvolvimento *Demonstra interesse na aquisição de conhecimentos e na participação em treinamentos e eventos, visando o aperfeiçoamento profissional.*				
Compreensão *Observa e analisa os elementos de uma situação, chegando à compreensão do todo.*				
Comprometimento *Conhece e compartilha dos objetivos e metas da empresa.*				
Cooperação *Oferecer auxílio e solicita a colaboração do grupo de trabalho nas atividades.*				
Criatividade *Apresenta sugestões criativas e inovadores ou propõe melhorias nas atividades.*				
Exigência de Qualidade e Eficiência *Procede de forma a executar atividades que satisfazem ou excedem os padrões de excelência estabelecidos pela empresa.*				
Iniciativa *Busca solucionar ou encaminhar problemas e dificuldades encontradas.*				
Planejamento *Sistematiza os meios para a realização das atividades.*				
Relacionamento *Contribui para a harmonia do ambiente, relacionando-se bem com o grupo.*				
Responsabilidade *Cumpre as tarefas nos prazos, respeita os horários de estágio e as normas.*				
Zelo *Prima pela limpeza, organização e segurança dos recursos e dos ambientes.*				

Apêndice D da Instrução Normativa Conjunta 03/2011 – PROGRAD/PROREC 1

As atividades desenvolvidas no estágio:	SIM	NÃO
a) Estão de acordo com o Plano de Estágio.		
b) São compatíveis com o curso e o período do estagiário.		
c) Satisfazem as expectativas da unidade concedente.		

Quando assinalado NÃO, apresente os motivos: _____

A formação que o estagiário está recebendo na UTFPR atende as necessidades da UCE?

Considerações do Supervisor de Estágio: _____

Data: ____/____/____ _____ _____
Supervisor de Estágio Estagiário

Considerações do Professor Responsável pela Atividade de Estágio: _____

Data: ____/____/____ _____
Professor Responsável pela Atividade de Estágio

A cada 6 (seis) meses, o Supervisor de Estágio deve preencher o Relatório Parcial de Supervisão de Estágio e enviar para o Professor Responsável pela Atividade de Estágio do curso do estagiário ou solicitar que o estagiário entregue. O Professor deve lançar esta informação no Sistema Integrado de Estágio.

ANEXO E – Ficha da Avaliação do Professor Orientador

Ficha de Avaliação do Professor Orientador utilizado, especificamente, no DAINF na UTFPR.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO
Ficha de Avaliação pelo Prof. Orientador

Nome do Estagiário:

Nome do Orientador de Estágio na UTFPR:

Atividades realizadas no período de / / a / /

Obs.: Os aspectos devem ser classificados com notas de 1 a 10, sendo 1 o menor grau e 10 o grau máximo. A ficha de avaliação deve ser entregue depois de preenchida e assinada pelo Orientador de Estágio na UTFPR.

Avaliação para o Relatório Final apresentado	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1 - Clareza de idéias e linguagem correta										
2 - Capítulos 1 e 2: Plano de estágio e Organograma da Empresa										
3 - Capítulos 3 e 4: Recursos Disponíveis para a Realização do Estágio e Atribuições do Estagiário										
4 - Capítulo 5: Atividades Desenvolvidas										
5 - Capítulos 6 e 7: Comentários do Estagiário e Conclusões										

Total de pontos para o Relatório Final apresentado: _____

Avaliação para o estágio realizado	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1 - Prazos: cumpriu os prazos regulamentados para a entrega dos relatórios parciais (20 horas e 180 horas do início do estágio) e para a entrega do Relatório Final (300 horas)?										
2 – Conhecimentos: aplicou conhecimentos teóricos e práticos obtidos no curso e também evoluiu tecnicamente com os conhecimentos adquiridos no estágio?										
3 - Qualidade no que foi produzido no estágio: a qualidade dos resultados obtidos durante o estágio é condizente com a formação do aluno?										
4 – Adaptação ao trabalho: o estágio, através das atividades desenvolvidas e de novos desafios, auxiliou na preparação para a futura carreira na engenharia?										
5 - Sociabilidade: o aluno demonstrou cordialidade no trato com o Orientador de Estágio?										

Total de pontos para o estágio realizado: _____

Média final = ((Total de pontos para o Relatório Final apresentado) + (Total de pontos para o estágio realizado))/10 : _____

_____ _____
Prof. Orientador de Estágio na UTFPR Estagiário

Curitiba, / /

ANEXO F – Ficha de Frequência

Ficha de frequência utilizada no estágio interno na UTFPR (DIEEM, 2014a)

Ministério da Educação
Universidade Tecnológica Federal do Paraná

FICHA DE CONTROLE DE FREQÜÊNCIA

Mês/Ano	_____/_____	**Data de Entrega:** _____/_____/_____	**Setor**:
Estagiário			**Ramal:**
Supervisor			**Ramal:**
Escolaridade		**Obrigatório ( ) Não-Obrigatório ( )**	**Carga Horária:**

Orientações para o(a) ESTAGIÁRIO(A):

1. Turnos Manhã e Tarde, entregar no último dia útil do mês. Turno da Noite, entregar no primeiro dia útil do mês subsequente.
2. Não rubrica-la nas ausências (sábados, domingos, feriados, faltas e recessos).
3. O atraso na entrega deste formulário acarretará suspensão provisória da Bolsa-auxílio, que ficará acumulada para o próximo mês.
4. **Fichas rasuradas não serão aceitas!**

Orientações para o(a) SUPERVISOR(A):

1. Memorando n° _____ (Obrigatório preencher este campo caso o estagiário realize estágio em horários diferentes àquele estipulado em seu contrato, de forma contínua)
2. Utilizar este campo para justificar as ausências ou reposições dos estagiários:

_____

_____

_____

DIA	MANHÃ		TARDE		NOITE		Rúbrica – Estagiário
	Entrada	Saida	Entrada	Saida	Entrada	Saida	
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							

_____ Estagiário

_____ Supervisor


UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA