



CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Santhiago Dionizio Pinto

PROPOSTA DE UMA METODOLOGIA DE USO DAS CERIMÔNIAS DO SCRUM PARA A FÁBRICA DE SOFTWARE DO CEULP/ULBRA

Palmas – TO

2019

Santhiago Dionizio Pinto

PROPOSTA DE UMA METODOLOGIA DE USO DAS CERIMÔNIAS DO SCRUM
PARA A FÁBRICA DE SOFTWARE DO CEULP/ULBRA

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Me. Jackson Gomes de Souza.

Palmas – TO

2019

Santhiago Dionizio Pinto

PROPOSTA DE UMA METODOLOGIA DE USO DAS CERIMÔNIAS DO SCRUM
PARA A FÁBRICA DE SOFTWARE DO CEULP/ULBRA

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Me. Jackson Gomes de Souza.

Aprovado em: ____/____/____

BANCA EXAMINADORA

Prof. Me. Jackson Gomes de Souza

Orientador

Centro Universitário Luterano de Palmas – CEULP

Prof. Esp. Fábio Castro Araújo

Centro Universitário Luterano de Palmas – CEULP

Profa. Me. Cristina D'Ornellas Filipakis

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2019

Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida, autor de meu destino, a minha esposa Dally Palmer por sempre estar presente e me incentivando, minha mãe Beatris Dionizio por acreditar em meu potencial e a toda minha família.

AGRADECIMENTOS

Ao Prof. Jackson, pela orientação sempre produtiva e seu interesse em meu aprendizado, e aos amigos que de alguma forma contribuíram para a conclusão deste trabalho.

“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes”. (Marthin Luther King)

RESUMO

PINTO, Santhiago Dionizio. **Proposta de uma metodologia de uso das cerimônias do *Scrum* para a Fábrica de *Software* do CEULP/ULBRA**. 2019. 73 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2019.

As metodologias ágeis têm o propósito de organizar e agilizar não só o desenvolvimento de *software*, como também qualquer outro tipo de negócio que demande organização e processos a serem seguidos. A utilização de metodologias ágeis em times de desenvolvimento é vantajosa no que se refere à qualidade e à rapidez com que os *softwares* são desenvolvidos, pois a interação entre artefatos é de grande utilidade para o processo. A fábrica de *Software* do CEULP/ULBRA adota algumas práticas da metodologia ágil *Scrum* e segue as recomendações sobre cerimônias detalhadas no guia de utilização do *Scrum*. Quando o time de desenvolvimento utiliza o *Scrum*, por exemplo, não há padrões ou formalismo para registro e gerenciamento das cerimônias, ocasionando problemas com a confiabilidade de documentos redigidos ou até mesmo a falta deles. Com isso, o presente trabalho propôs uma metodologia de uso das cerimônias do *Scrum* em que o objetivo principal é o registro das cerimônias, utilizando um *software* que se comunica com o sistema de gerenciamento de repositórios de documentos e de projetos *Gitlab*. A metodologia proposta faz uso de etapas para a contínua atualização e organização do repositório *Gitlab* e de um *software* para o gerenciamento das atas redigidas em reuniões.

Palavras-chave: *Scrum*; Cerimônias; *API Gitlab*; *Gitlab*, *GLScrum*, *GLSMeeting*.

LISTA DE FIGURAS

Figura 1 - Grupos de processos de gerenciamento de projetos	15
Figura 2 - Relacionamento entre os processos de controle	16
Figura 3 - Valores do Manifesto Ágil.....	19
Figura 4- O Processo da Extreme Programimng (XP)	20
Figura 5 - Fluxo do processo Scrum.....	22
Figura 6 - Exemplo de Product Backlog	25
Figura 7 - Exemplo de Sprint Backlog	26
Figura 8 - Exemplo de Burndown chart	27
Figura 9 - Exemplo de Task Board.....	28
Figura 10 - Etapas de desenvolvimento dos resultados.....	33
Figura 11 - Cadastro de Aplicação para utilização do Gitlab como Provedor OAuth	35
Figura 12 - Arquitetura de acesso a dados do GLSMeeting.....	36
Figura 13 - Fluxo de autenticação OAuth Gitlab	36
Figura 14- Diagrama de caso de uso do GLSMeeting.....	38
Figura 15 - Arquitetura do Django	41
Figura 16 - Estrutura das principais pastas e principais arquivos do GLSMeeting.....	42
Figura 17 - Diagrama de dependência do model do software GLSMeeting	43
Figura 18 - Página para login do software GLSMeeting.....	48
Figura 19 - Página de login no Gitlab	49
Figura 20 - Processo da metodologia de uso das cerimônias	52
Figura 21 - Localização "cadastro de projeto" no processo da metodologia.....	54
Figura 22 - Página de cadastro de projeto no Gitlab	54
Figura 23 - Localização "cadastro de issues" no processo da metodologia	55
Figura 24 - Página de cadastro de Issue no Gitlab	56
Figura 25 - Localização "cadastro de Workspace" no processo da metodologia	56
Figura 26 - Página de cadastro de novo Workspace no GLSMeeting.....	57
Figura 27 - Página do workspace no GLSMeeting	58
Figura 28 - Página de cadastro de integrante no GLSMeeting.....	59
Figura 29 - Localização "realizar Planning Poker" no processo da metodologia.....	60
Figura 30 - Localização "Definir Sprint Backlog" no processo da metodologia	60
Figura 31 - Localização "cadastro de Milestone" no processo da metodologia.....	61
Figura 32 - Página de cadastro de nova Milestone no Gitlab.....	61
Figura 33 - Localização "cadastro de ATA" no processo da metodologia.....	62

Figura 34 - Página do workspace no GLSMeeting	62
Figura 35 - Ação após clicar no botão " nova cerimônia"	63
Figura 36 - Página de cadastro de reunião de Planejamento no GLSMeeting	63
Figura 37 - Exemplo de visualização/impressão de ata de planejamento	64
Figura 38 - Localização "Daily Scrum" no processo da metodologia.....	65
Figura 39 - Localização "cadastro de ata diária" no processo da metodologia	65
Figura 40 - Página de cadastro de reunião diária no GLSMeeting.....	66
Figura 41 - Localização "cadastro de ata final" no processo da metodologia.....	67
Figura 42 - Página de cadastro de reunião final no GLSMeeting	68

LISTA DE TABELAS

Tabela 1- Detalhamento dos casos de uso	39
Tabela 2 – Definição dos models implementados em gls/models.py.....	44
Tabela 3 – Descrição dos métodos implementados em gls/views.py.....	46

LISTA DE TRECHOS DE CÓDIGOS

Trecho de código 1 – Exemplo de URL de redirecionamento após usuário autenticar no OAuth Gitlab	37
Trecho de código 2 - Model Cerimonia.....	44
Trecho de código 3 – View “login_user” para apresentação da página de login	48
Trecho de código 4 – Criar autenticação do usuário no GLSMeeting	49
Trecho de código 5 - Criando usuário no GLSMeeting com vínculo ao OAuth Gitlab.....	50
Trecho de código 6 – View “nova_diaria” para apresentação da página de cadastro de reunião diária.....	50

SUMÁRIO

SUMÁRIO	10
1. INTRODUÇÃO	11
2. REFERENCIAL TEÓRICO	13
2.1. GUIA DO CONHECIMENTO EM GERENCIAMENTO DE PROJETOS	13
2.1.1. GERENCIAMENTO DAS COMUNICAÇÕES	14
2.1.2. GERENCIAMENTO DE TEMPO	14
2.1.3. GRUPOS DE PROCESSOS DE GERENCIAMENTO DE PROJETOS	15
2.2. METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE	17
2.2.1. MANIFESTO ÁGIL	18
2.2.2. <i>SCRUM</i>	21
3. METODOLOGIA	32
3.1. MATERIAIS	32
3.2. MÉTODOS	32
4. RESULTADOS	34
4.1. VISÃO GERAL DA APLICAÇÃO ATUAL DO SCRUM NA FÁBRICA DE SOFTWARE DO CEULP ULBRA	34
4.2. <i>SOFTWARE GLSMEETING</i>	34
4.2.1. ESTRUTURA DE ACESSO AOS DADOS	34
4.2.2. DIAGRAMA DE CASO DE USO	38
4.2.3. ARQUITETURA DO SOFTWARE	40
4.3. GLSCRUM - METODOLOGIA DE USO DAS CERIMÔNIAS	51
4.3.1. PAPÉIS	52
4.3.2. FERRAMENTAS UTILIZADAS	53
4.3.3. PRODUCT BACKLOG	53
4.3.4. SPRINT PLANNING MEETING	59
4.3.5. SPRINT	64
4.3.6. DAILY SCRUM	65
4.3.7. SPRINT REVIEW & RETROSPECTIVE	66
4.3.8. NOVA SPRINT	68
4.3.9. PRODUTO PRONTO	69
5. CONSIDERAÇÕES FINAIS	70
REFERÊNCIAS	71

1. INTRODUÇÃO

De acordo com estudos de Souza (2014), são previstos que os lançamentos de produtos feitos por equipes que utilizaram metodologias ágeis sejam mais precoces e com mais qualidade, quando se leva em consideração a satisfação do cliente com os objetivos do produto.

Neste contexto de metodologias ágeis, o *software* pode ser modificado inúmeras vezes em seu desenvolvimento. Além de toda a documentação que é gerada, é necessário o engajamento da equipe e o entendimento por completo do produto em desenvolvimento. Os métodos ágeis surgiram com o propósito de melhorar o trabalho da equipe, dando ênfase ao código e as alterações de documentações (UTIDA, 2012). Para que uma metodologia ágil seja implantada em um ambiente, é importante saber que será necessária uma explicação e um entendimento real do método por parte da equipe, além de revisões constantes sobre ferramentas, *softwares* e divisões de trabalho.

O *Scrum* é uma metodologia ágil para gerenciar o desenvolvimento de softwares e manter produtos complexos, e tem como principais características: (i) a ênfase na comunicação em tempo real; (ii) a documentação reduzida comparando com outras metodologias; (iii) e, tratando também cada nova iteração no *software* como um novo projeto que será realizado (SOUZA, 2014).

O Guia *Scrum*, desenvolvido e mantido por Ken Schwaber e Jeff Sutherland, demonstra conceitos, eventos, regras e procedimentos a serem tomados quando se utiliza o *Scrum*. Dentre os eventos, é importante destacar as cerimônias, que são uma série de reuniões, que garantem a transparência através da comunicação e proporcionam momentos de planejamento, inspeção e adaptação do projeto. As cerimônias devem acontecer para melhor planejamento e entendimento de cada *Sprint* e podem ser adaptadas para cada equipe porque, de acordo com Leidemer (2013), “os processos existentes nos métodos ágeis têm o papel de dar suporte à equipe de desenvolvimento”.

As cerimônias do *Scrum*, se aplicadas conforme o Guia, não são bem detalhadas e tem o entendimento de forma genérica para melhor aplicação em cada ocasião/organização. Os itens que o *Guia Scrum* deixa de detalhar são especificamente os termos de ritos e processos de reuniões, de registros delas, mas trata de identificar, previamente, possíveis impedimentos para a conclusão do projeto conforme planejado. Frente a este contexto, pensou-se na proposta de uma metodologia de uso das cerimônias, para que sejam realizadas com suporte informatizado e consultas atualizadas pela *API Gitlab*, referenciando os projetos e suas tarefas com o propósito de redigir atas para maior facilidade no gerenciamento e realização de cerimônias.

A Fábrica de *Software* do CEULP/ULBRA adota o *Scrum* como metodologia no desenvolvimento dos seus projetos. Atualmente, são conduzidas reuniões diárias (*daily scrum*) e reuniões de revisão (*Sprint review*). As reuniões são registradas na forma de atas que descrevem informações como quais foram os participantes, os assuntos tratados e os relatos da equipe a respeito de questões práticas, como conclusões de tarefas e impedimentos.

A Fábrica de *Software* utiliza o *Gitlab* como ferramenta para o gerenciamento de projetos e versionamento de códigos-fontes. No sentido do gerenciamento, utiliza recursos como *issues* e *board (kanban)* para descrever, respectivamente, as funcionalidades a serem desenvolvidas ou tarefas a serem executadas e as situações das tarefas: a fazer, em andamento e concluídas.

Nesse sentido, o objetivo deste trabalho foi propor uma metodologia de uso das cerimônias do *Scrum (GLScrum)* aplicada no contexto da Fábrica de *Software* do CEULP/ULBRA, que adota o *Gitlab* e seus recursos. Isso foi alcançado por meio da identificação das necessidades e particularidades, definição das etapas ou fases da metodologia de uso da cerimônia do *Scrum* e o desenvolvimento de um *software* de apoio nomeado como *GLSMeeting*.

Este documento está organizado da seguinte forma: a seção 2 apresenta o referencial teórico (Guia PMBOK, Metodologias de desenvolvimento de *Software*, *Scrum*); a seção 3 apresenta os materiais e a metodologia; a seção 4 apresenta os resultados, descrevendo a metodologia e o *software* de apoio; por fim, a seção 5 apresenta considerações finais sobre os resultados.

2. REFERENCIAL TEÓRICO

Essa seção irá abordar conceitos de gerenciamento de projetos, metodologias de desenvolvimento e apresentação geral da metodologia *Scrum* permitindo, assim, uma visão conceitual e detalhada de cada abordagem.

2.1.GUIA DO CONHECIMENTO EM GERENCIAMENTO DE PROJETOS

Segundo Mangelli (2013), o gerenciamento de projetos é característico pelo emprego de conhecimentos, habilidades, ferramentas e técnicas para planejar, programar, executar e controlar as atividades do projeto a fim de uma aplicação adequada de seus processos lógicos e organizacionais. Um processo é um conjunto de ações e atividades inter-relacionadas, que são executadas para produzir um produto, um serviço predefinido ou um resultado (MANGELLI, 2013).

Além disso o gerenciamento de projetos é, normalmente, o campo de responsabilidade de um gerente de projeto individual. Esse indivíduo raramente participa nas atividades que produzem o resultado, mas se esforça para manter o progresso e a interação produtiva das várias partes, reduzindo o risco geral de fracasso (BOMFIN; NUNES; HASTENREITER, 2012).

Baseado nisso, o Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK) é uma norma definida para a profissão de gerenciamento de projetos (PMI, 2008). O Guia do PMBOK fornece uma introdução aos principais conceitos na área de gerenciamento de projetos. Além do guia conter campos sobre padrão, escopo e entre outros processos para gerenciamentos de projetos, ele fornece diretrizes para gerenciar projetos individuais, define o gerenciamento de projetos e conceitos relacionados e descreve o ciclo de vida destes.

O Guia PMBOK também fornece e promove um vocabulário comum dentro da profissão de gerenciamento de projetos para discutir, escrever e aplicar conceitos de gerenciamento de projetos. Tal vocabulário padrão é um elemento essencial de uma disciplina profissional (PMI, 2008).

O Guia é caracterizado como um padrão para gerenciar grande parte dos projetos em vários tipos de indústrias. Este padrão descreve, portanto, os processos, as ferramentas e as técnicas de gerenciamento de projetos usados para gerenciar um projeto.

O PMBOK é considerado pelo *Project Management Institute* (PMI) como padrão de referência de gerenciamento de projetos para seus programas e certificações de desenvolvimento profissional. A crescente aceitação do gerenciamento de projetos indica que a aplicação de conhecimento, processos, habilidades, ferramentas e técnicas apropriados pode ter um impacto significativo no sucesso do projeto (PMI, 2008).

A 4ª. edição do Guia PMBOK define os processos de gerenciamento e estes foram divididos em nove áreas de conhecimento: gerenciamento de integração, de escopo, de tempo, de custos, de qualidade, de recursos humanos, de comunicações, de riscos e de aquisições. Como parte importante para o contexto deste trabalho, a seguir serão detalhadas as áreas de conhecimento gestão de comunicações e gestão de tempo.

2.1.1. Gerenciamento das Comunicações

O gerenciamento das comunicações inclui os processos necessários para assegurar que as informações do projeto sejam geradas, coletadas, distribuídas, armazenadas, recuperadas e organizadas de maneira oportuna e apropriada (BOMFIN; NUNES; HASTENREITER, 2012).

Uma comunicação eficaz cria uma ponte entre as partes interessadas envolvidas do projeto. Nisso, os gerentes de projetos gastam a maior parte de seu tempo se comunicando com os membros da equipe e outras partes interessadas ou outras partes externas.

As principais atividades que o processo de gerenciamento das comunicações do projeto assegura são:

Identificar os Stakeholders: identificação das pessoas ou organizações que podem ser afetadas pelo projeto, como também de documentação das informações relevantes relacionadas aos seus interesses, ao seu envolvimento e ao seu impacto no sucesso do projeto;

Planejar as comunicações: determinação das necessidades de informação dos *stakeholders* e definição de uma abordagem de comunicação do projeto;

Distribuir informações: disponibilizar para os *stakeholders* as informações necessárias do projeto, conforme planejado;

Gerenciar as expectativas dos stakeholders: comunicar e interagir com os *stakeholders* para atender às suas necessidades e solucionar as questões à medida que ocorrerem;

Reportar o desempenho: coletar e distribuir informações sobre o desempenho, incluindo relatórios de andamento e medições do progresso do(s) projeto(s) (MANGUELLI, 2013).

A comunicação pode conectar vários ambientes culturais e organizacionais, de diferentes níveis de conhecimento e de diversas perspectivas nos resultados do projeto.

2.1.2. Gerenciamento de Tempo

Segundo Bomfin, Nunes e Hastenreiter (2012), o gerenciamento de tempo deve incluir o necessário para gerenciar o término pontual do projeto. Existe ainda a distinção entre a informação do cronograma do projeto e os dados que são produzidos, fazendo referência ao mecanismo de agendamento do projeto como o modelo de cronograma. Na prática, o cronograma e o modelo são chamados somente de cronograma.

As principais atividades no gerenciamento de tempo são:

Definir as atividades: ações a serem realizadas para produzir as entregas do projeto;
Sequenciar as atividades: identificação e documentação dos relacionamentos entre as atividades do projeto;

Estimar os recursos da atividade: estimativa dos tipos e quantidades de material, pessoas, equipamentos que serão necessários para realizar cada atividade;

Estimar a duração das atividades: estimativa do número de períodos de trabalho que serão necessários para terminar atividades específicas com os recursos estimados;

Desenvolver o cronograma: análise das sequências das atividades, suas durações, recursos necessários e restrições visando a criação do cronograma;

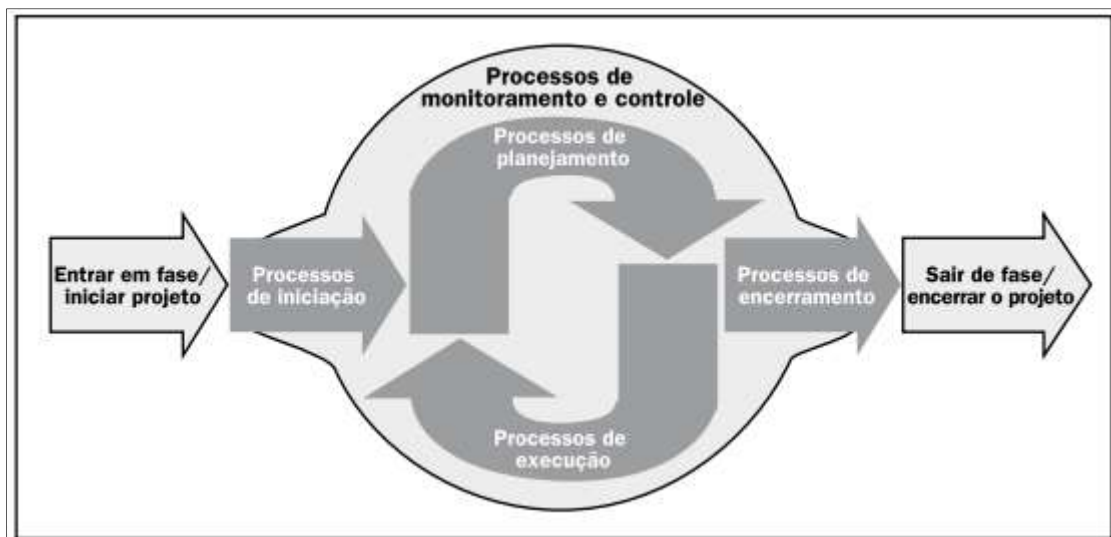
Controlar o cronograma: monitoramento do andamento do projeto, para atualização do seu progresso e gestão das mudanças realizadas no cronograma (MANGUELLI, 2013).

Portanto, o gerenciamento de tempo tem por objetivo gerenciar o término preciso do projeto.

2.1.3. Grupos de processos de gerenciamento de projetos

As áreas de conhecimentos são agrupadas, ao longo do projeto, em cinco grandes grupos de processos: **iniciação, planejamento, execução, monitoramento e controle e encerramento**, conforme é apresentado na *Figura 1*.

Figura 1 - Grupos de processos de gerenciamento de projetos



Fonte: Adaptado de PMI (2008)

Esses grandes grupos de processos são descritos de forma breve a seguir:

O grupo de **processos de iniciação** define um novo projeto ou uma nova fase de um projeto existente, pela autorização de início do projeto.

O grupo de **processos de planejamento** define o escopo do projeto, refinamento dos objetivos e desenvolve o curso de ação necessário para concretizar os objetivos para os quais ele foi criado.

O grupo de **processos de execução** realiza o trabalho definido no plano de gerenciamento do projeto e satisfaz suas especificações.

O grupo de **processos de monitoramento e controle** acompanha, revisa e regula o progresso e o desempenho do projeto, identificando todas as áreas nas quais serão necessárias mudanças e iniciando as mudanças correspondentes.

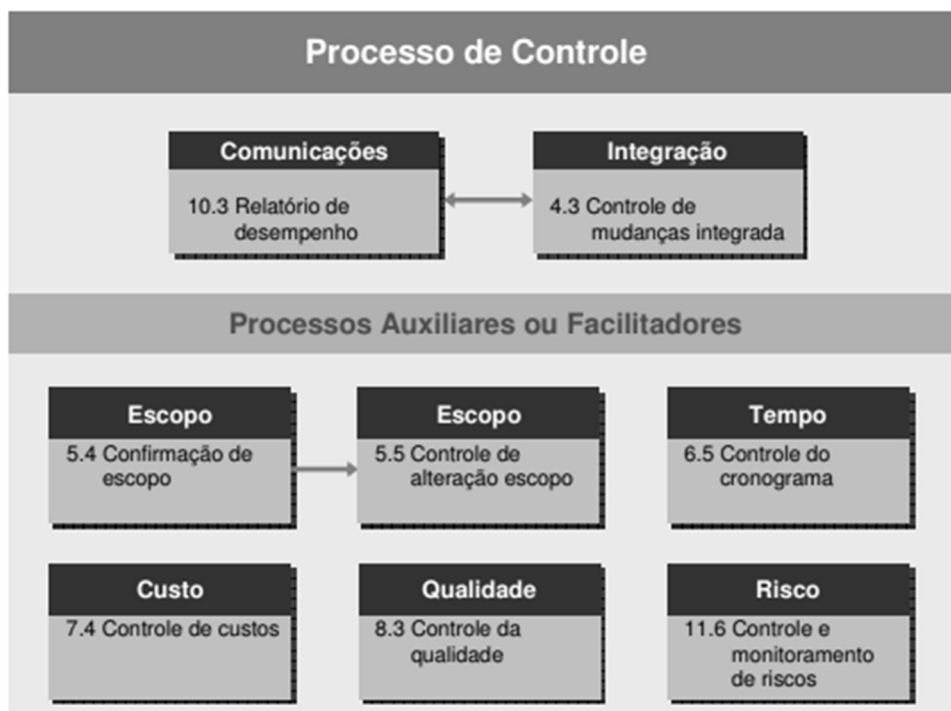
O grupo de **processos de encerramento** finaliza todas as atividades de todos os grupos de processos, visando encerrar formalmente o projeto (BOMFIN; NUNES; HASTENREITER, 2012).

Dentre os grupos de processos apresentados, para o contexto deste trabalho, destaca-se o grupo de monitoramento e controle, que será detalhamento a seguir.

Monitoramento e Controle

Segundo Oliveira (2007), o grupo de monitoramento e controle tem por atividade assegurar que o objetivo do projeto seja cumprido através do monitoramento e medição do progresso regular, a fim de identificar variações que possam ocorrer durante o projeto e que possam ser tomadas ações corretivas quando necessário. O grupo de processos de monitoramento e controle do guia PMBOK é constituído pela área de execução do projeto. A *Figura 2* demonstra os processos que compõem esta etapa e seus relacionamentos.

Figura 2 - Relacionamento entre os processos de controle



Fonte: Oliveira (2007)

Como processo essencial, o processo de controle é composto por relatórios de desempenho e controle de mudanças integrada. Processos auxiliares e/ou facilitadores são compostos por: confirmação de escopo; controle de mudanças do escopo; controle dos custos; controle da qualidade; controle do cronograma; e controle e monitoramento dos riscos. Estes processos são característicos do grupo de execução da grande área de monitoramento e controle do guia PMBOK.

O desempenho deste grupo está baseado na investigação das variações em relação ao plano de gerenciamento do projeto. Este monitoramento engloba todo o esforço do projeto, como também toda a fase do projeto, no que tange ao planejamento de ações corretivas e *feedbacks*, processos, mudanças e revisões (OLIVEIRA, 2007).

Para Reis (2011), outra característica importante associada ao monitoramento e controle está na sua relação com os demais processos, uma vez que durante todo ciclo de vida do projeto, é proposto que “seja avaliado se a realização do mesmo está sendo cumprida de acordo com o que foi planejado ou se existem desvios de execução, que podem sugerir ações de correção de rumo” (PMI, 2008).

Diante da explicação sobre o gerenciamento de projetos, é importante entender sobre as metodologias de desenvolvimento de software. O próximo tópico apresentará informações sobre o início das metodologias ágeis e detalhamento da metodologia *SCRUM*.

2.2.METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

O desenvolvimento de *Software* é um processo do qual pode-se afirmar que a implementação do código é dependente, pois para se desenvolver um *software* de computador, é necessário, além de outros procedimentos, a programação com implementações de códigos. Se utilizadas metodologias para o gerenciamento do desenvolvimento desses *softwares*, o resultado poderá ser mais eficiente, no sentido de rapidez, qualidade, entendimento e padrões pré-definidos (SOUZA, 2014).

Uma metodologia de desenvolvimento de *software* pode ser entendida como a definição de objetivos e papéis a todos os envolvidos, evitando que o conhecimento do sistema seja de poucos, bem como a instrução de normas e princípios a fim de garantir o padrão e integração entre os sistemas desenvolvidos (SOUZA, 2014).

No desenvolvimento de *software*, o método tradicional exige que seu processo seja mais planejado, ou seja, deve ser realizada a definição de uma documentação de todos os requisitos no processo de desenvolvimento do projeto. Há diversas metodologias tradicionais, sendo algumas delas: *CleanRoom*, *Domain Driven Design*, *Joint Application*, *Model Driven*

Architecture, Model Waterfall, Spiral Model, Prototyping Model e Rational Unified Process. As metodologias tradicionais são práticas mais burocráticas, e têm por objetivo, além de auxiliar/orientar no processo de gerenciamento de projetos, tornar o desenvolvimento mais previsível.

De acordo com Stoica, Mircea e Ghilic-micu (2013) é importante levar em consideração alguns fatores ao adotar uma prática de desenvolvimento de *software*, como: **abordagem** na forma de desenvolvimento; **maturidade da empresa** ao adotar determinado método de gerenciamento de projeto; **tamanho da equipe**, que se refere a como a mesma será gerenciada; **domínio**, que diz respeito a conhecimento de cada projeto; **perspectiva de mudança**, que se refere a intensidade do impacto ocasionando no andamento do projeto; **comunicação**, a forma que a mesma é explorada; **cultura**, no que tange à política de comportamento e organização da equipe; **documentação**, que refere-se à composição da documentação, em características extensiva ou não; e **investimento**, que diz respeito ao retorno geral do projeto.

Em meados dos anos 90, conforme Alcântara (2009), o termo "metodologias ágeis" passou a se tornar mais conhecido, o que pode ter ocorrido devido à chamada "crise do *software*", um período de dificuldade no desenvolvimento de *software* por causa do rápido crescimento de demandas por *softwares*. A partir dos anos 2001, com o surgimento do Manifesto Ágil, escrito por Kent Beck e outros dezesseis profissionais de software, o conceito de metodologias ágeis passou a se tornar mais popular (SOUZA, 2014). Por questões de escopo, na próxima seção serão abordados os fatores tanto técnicos quanto conceituais das metodologias ágeis.

2.2.1. Manifesto Ágil

No ano de 2001, um grupo de dezessete profissionais de *software* se reuniram para estabelecer uma metodologia de desenvolvimento em comum, mas, devido às particularidades sobre conceitos e métodos que cada um deles tinham sobre desenvolvimento de *software*, não chegaram a uma metodologia (ALCÂNTARA, 2009). Então, segundo Silva (2016), como resultado dessa reunião, os membros do grupo listaram itens que geravam bons resultados e que eram comuns a todos, criando e assinando o “Manifesto Ágil” que se baseia em valores, como demonstrado na Figura 3.

Figura 3 - Valores do Manifesto Ágil

Valor_1	mais que	Valor_2
indivíduos e interações	>	processos e ferramentas
software em funcionamento	>	documentação abrangente
colaboração com o cliente	>	negociação de contratos
responder a mudanças	>	seguir um plano

Fonte: Adaptado de Alcântara (2009)

De acordo com Souza (2014) os valores foram definidos como menor e maior importância, ou seja, valorize mais o **Valor_1**, mas não deixe de valorizar o **Valor_2**.

Com base nos valores definidos, foram elaborados os 12 princípios das metodologias ágeis, segundo Souza (2014):

1. Uma das maiores prioridades é pela satisfação do consumidor por meio de entregas contínuas de valor e em um curto período de tempo;
2. Mudanças de requisitos são bem-vindas mesmo em estágios avançados do desenvolvimento. Processos ágeis aproveitam as mudanças em benefícios da vantagem competitiva do cliente;
3. Entregar o produto funcionando em curto período;
4. Desenvolvedores e gestores devem trabalhar diariamente em conjunto;
5. Criar projetos com as pessoas motivadas. Confie nelas e dê suporte e ambiente para que o trabalho seja feito;
6. O método mais eficiente e eficaz de transmitir informações em um projeto é sempre informando diretamente ao cliente;
7. Produto funcionando é a principal medida para o progresso;
8. Processos ágeis promovem o desenvolvimento sustentável. As empresas contratantes, os desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
9. Atenção contínua excelência técnica e ao design melhoram a agilidade;
10. Simplicidade. A arte de deixar de fazer trabalhos desnecessários é essencial;
11. Os melhores requisitos, arquiteturas e design surgem de equipes que praticam a autogestão;
12. Em um determinado período de tempo a equipe deve repensar sobre como se tornar mais eficaz. Após a reflexão deve reajustar-se de acordo com as necessidades percebidas.

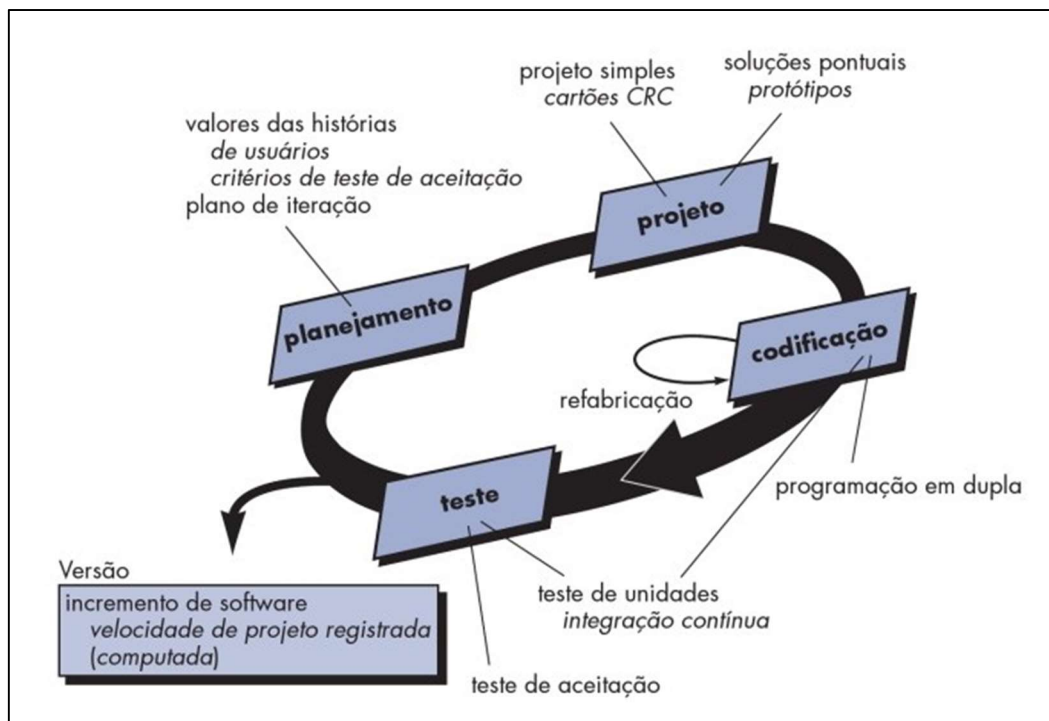
Com isso, atualmente, a diversidade de metodologias ágeis de gerenciamento e desenvolvimento de *software* é considerada vasta, algumas das quais, segundo Ambler (2002) são: *Agile Modeling*, *Agile Unified Process*, *Behavior Driven Development*, *Crystal*, *Dynamic Systems Development Methodology*, *Extreme Programming (XP)*, *Feature Driven Development*, *Lean Software Development*, *Microsoft Solutions Framework*, *Rapid Application Development*, *Scrum* e *Test Driven Development*. Dentre essas as mais conhecidas são *Extreme Programming (XP)* e *Scrum* (SANTOS, 2014).

Segundo Pressman (2011), o XP é a abordagem mais amplamente utilizada para desenvolvimento de *software* ágil, empregando uma abordagem orientada a objetos como base para seu paradigma de desenvolvimento e envolvendo um conjunto de regras e práticas constantes no contexto de quatro atividades metodológica: planejamento, projeto, codificação e testes.

A

Figura 4 ilustra o processo da *Extreme Programming* (XP), com destaque de alguns conceitos chaves para compreensão do processo de acordo com Pressman (2011).

Figura 4- O Processo da *Extreme Programimng* (XP)



Fonte: Santos (2014)

Conforme é apresentado na Figura 4, o processo XP é composto por quatro atividades, que são: (i) **Planejamento**: é uma atividade de levantamento de requisitos, a fim de possibilitar uma percepção ampla sobre o que foi solicitado; (ii) **Projeto**, deve consistir de uma representação que preserve a simplicidade, segundo Pressman (2011); (iii) **Codificação**, essa atividade trabalha com o conceito de programação em dupla – Pressman (2011) traz para discussão que o processo XP recomenda que duas pessoas trabalhem juntas para desenvolver um objetivo; e (iv) **Testes**, esta atividade tem por objetivo explorar um conjunto de testes, estes para benefício do desenvolvimento dos requisitos solicitados.

Pressman (2011) alerta que todo processo de *software* tem suas falhas. Portanto, o segredo para se utilizar um processo é saber apresentar fraquezas e adaptá-las de acordo com as necessidades da organização.

2.2.2. SCRUM

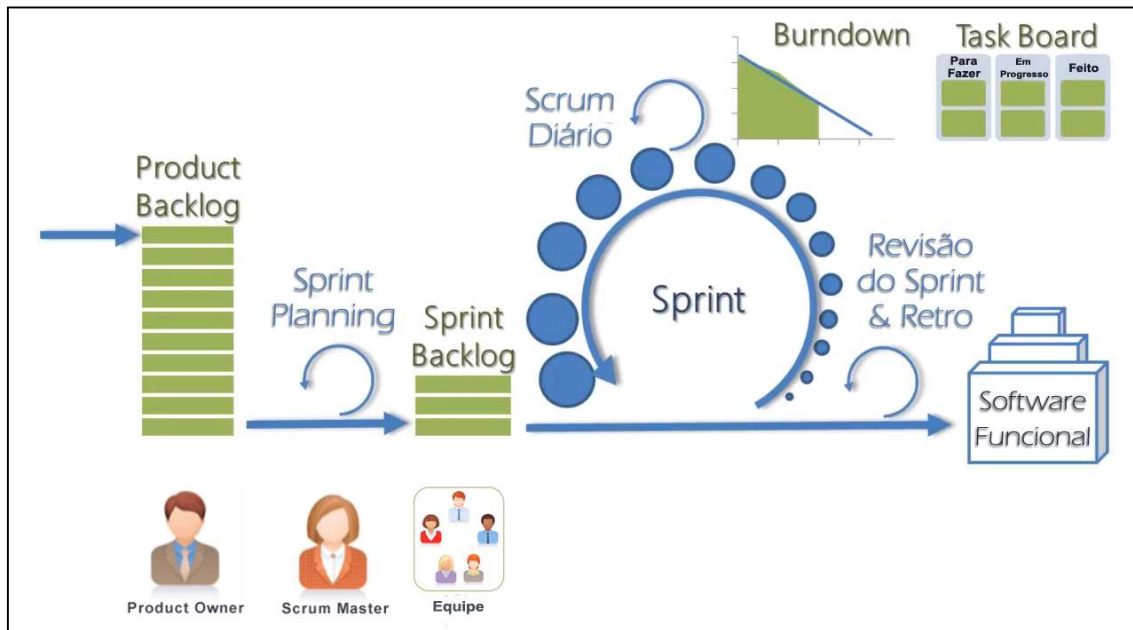
Segundo Pressman (2011), o *Scrum* é uma metodologia de desenvolvimento ágil de *software* concebido por Jeff Sutherland e sua equipe nos anos 90. É um método iterativo e incremental, baseado num conjunto de valores que visa explorar o sucesso do projeto e entrega de valor para o cliente.

De acordo com Sutherland e Schwaber (2011) o *Scrum* teve como base um artigo de 1986 da *Harvard Business Review* que explorava sobre as práticas associadas a grupos de desenvolvimento de produtos de sucesso. Ainda afirmavam que neste artigo foram introduzidos os termos “*Rugby*” e “*Scrum*”, que conceitualmente estavam se relacionando a desenvolvimento bem-sucedido ao jogo de *Rugby* no qual uma equipe auto organizada move-se em conjunto no campo do desenvolvimento de produtos. Em 1995, este foi então formalizado por Ken Schwaber e Dr. Jeff Sutherland. Além disso, o *Scrum* trabalha em consistência com os princípios do manifesto ágil.

Segundo Soares (2004), o objetivo desta metodologia é fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto. Portanto, o *Scrum* apresenta uma abordagem empírica que aplica algumas ideias da teoria de controle de processos industriais para o desenvolvimento de *softwares*, reintroduzindo as ideias de flexibilidade, adaptabilidade e produtividade.

Pressman (2011) ainda aponta as seguintes atividades que são usadas para orientar no desenvolvimento dentro do processo, de acordo com o manifesto, que incorporam: requisitos, análise, projeto, evolução e entrega. Para tanto, o *Scrum* compreende de um ciclo de execução baseado em *Sprints*, ou seja, esse ciclo é orientado de acordo com cada produto e planejado por atividades, conforme é apresentado na Figura 5.

Figura 5 - Fluxo do processo *Scrum*



Fonte: Jose Junior (2019)

O fluxo apresentado na Figura 5 será detalhado nas próximas seções, tendo em vista os principais eixos de execução, que consistem em: eventos, artefatos e os papéis do processo.

2.2.2.1. Papéis do *Scrum*

No *Scrum* são atribuídos um conjunto de três papéis, formando um time de desenvolvimento. Estes papéis serão explorados nas seções a seguir, a fim de possibilitar uma visão conceitual detalhada do ciclo de funcionamento e/ou resultados que podem ser gerados pelo fluxo da equipe.

Product Owner

O papel do *Product Owner* é caracterizado como o proprietário do produto, ou seja, aquele que representa a empresa do respectivo projeto. Também conhecido como PO, é o responsável pelas decisões de negócios e por ter o contato direto com as partes interessadas do projeto.

Além disso, segundo Marçal et al. (2007), o PO representa os interesses de todos no projeto, como também: define as causas do projeto criando requisitos, assim nomeados no *Product Backlog*; objetivos e planos de entregas; prioriza o *Product Backlog* a cada *Sprint* do processo, garantindo que as funcionalidades de maior valor de entrega sejam construídas prioritariamente. Baseado nisso, o *Product Owner* deve estar a caráter disponível para a equipe

de desenvolvimento, a fim de certificar que o projeto esteja de acordo com as necessidades estabelecidas na fase inicial.

Scrum Team

O *Scrum Team*, também conhecido como Time de Desenvolvimento, consiste de profissionais que realizam o trabalho de entregar uma versão que potencialmente incrementa o produto “pronto” ao final de cada *Sprint*.

De acordo com Schwaber e Sutherland (2014) o *Scrum Team* é estruturado e autorizado pela organização para organizar e gerenciar seu próprio trabalho. A cooperação resultante aperfeiçoa a eficiência e a eficácia do Time de Desenvolvimento como um todo. Além disso, os autores afirmam que o tamanho ideal do time de desenvolvimento deve se caracterizar como pequeno o suficiente para se manter ágil e grande o suficiente para completar uma parcela significativa do trabalho dentro dos limites da *Sprint*.

Segundo Schwaber e Sutherland (2014), para a realização do trabalho do time de desenvolvimento é necessário compor-se das seguintes características: auto-organização, multifuncionalidades; especialidades nas áreas de habilidades; e um time de desenvolvimento não contém sub-times, ou times secundários, dedicados a domínios específicos de conhecimento, tais como teste ou análise de negócios.

Portanto, *Scrum Team* desenvolve as funcionalidades desejadas do produto; define como modificar o *Product Backlog* em incremento de funcionalidades numa iteração gerenciando seu próprio trabalho, sendo responsáveis coletivamente pelo sucesso da iteração e consequentemente pelo projeto como um todo (MARÇAL et al., 2019).

Scrum Master

O *Scrum Master* é exercido pelo gerente de projetos e tem como papel facilitar o trabalho do *Scrum team*. O seu principal objetivo é garantir que todos sigam as regras e práticas do *Scrum*; e é responsável por remover os impedimentos do projeto.

Schwaber e Sutherland (2014) afirmam que o *Scrum Master* se dispõe ao *Scrum Team* de modo a: treinar em autogerenciamento e interdisciplinaridade; ensinar na concepção de valores; remover impedimentos no processo de desenvolvimento; liderança; e facilitar nos eventos do *Scrum*. Além disso, contribui para a Organização de várias maneiras: liderança e treinamento na prática do *Scrum*; produtividade do time e eficácia na aplicação do *Scrum* nos processos da Organização.

O papel do *Scrum Master* no processo do *Scrum* está direcionado a gerenciar o processo do *Scrum*, ensinando-o a todos os envolvidos e partes interessadas e implementando de modo que esteja adequado a cultura da Organização (MARÇAL et al., 2019).

2.2.2.2. *Sprint*

“O *Scrum* trabalha com desenvolvimento incremental, dividindo seus processos em *Sprints*. *Sprints* é o nome como são chamadas as interações que ocorrem no *Scrum*, ou seja, o período de trabalho para cada fase incremental” (SILVA; SOUZA; CAMARGO, 2013).

No *Scrum*, o período de duração de uma *Sprint* é caracterizado como *Timebox*. O *Scrum* é baseado em entrega de valor a cada *Sprint*, logo é recomendável que durante o desenvolvimento do produto exista um padrão, um mesmo período de entrega para as *Sprints*. Segundo Marçal et al. (2007), isto é importante para se conseguir medir o progresso e a produtividade da equipe no projeto, na finalidade de gerar valor de entrega para o cliente.

2.2.2.3. Artefatos

Os artefatos definidos pelo *Scrum* são um conjunto de tarefas concebidas durante a realização dos eventos do processo. Os artefatos serão explorados nas seções a seguir, a fim de possibilitar uma visão conceitual detalhada do ciclo de funcionamento e/ou entrega dos artefatos do processo.

Product Backlog

O *Product Backlog* é uma lista ordenada de atividades de tudo que deve ser necessário no produto. O *Product Owner* é o responsável pelo *Product Backlog*, incluindo seu conteúdo, disponibilidade e ordenação da lista das atividades. O *Product Backlog* lista as características, funções, os requisitos, as melhorias e correções que formam as alterações que devem ser feitas no produto (SCHWABER e SUTHERLAND, 2014).

Após a definição do *Product Backlog* é necessário determinar os itens da lista de tarefas que serão executados pelo *Scrum Team* durante a *Sprint*. Essa lista de tarefas tem como base a ordem das atividades definidas pelo *Product Owner*. A Figura 6 exemplifica o modelo de *Product Backlog* fornecido pelo *Product Owner*.

Figura 6 - Exemplo de *Product Backlog*

Backlog do produto	
Id	História
1	O cliente deve conseguir pesquisar livros por autor
2	O cliente deve conseguir pesquisar livros por título
3	O cliente deve conseguir pesquisar livros por editora
4	O cliente deve conseguir comprar um livro escolhido (compra 1 item por vez)
5	O cliente deve conseguir incluir o item em um carrinho de compras
6	O cliente deve conseguir pagar sua compra via boleto bancário
7	O cliente deve conseguir pagar sua compra via cartão de crédito
8	O cliente deve conseguir rastrear sua encomenda
9	O cliente deve ser notificado por e-mail das etapas de sua compra (pagamento, aprovação, correio)
10	O cliente deve conseguir enviar o feedback de seu atendimento após o recebimento do produto

Fonte: Tavares (2019)

A Figura 6 detalha as funcionalidades que o *Product Owner* entende serem necessárias para o projeto, portanto essa lista é tratada e atualizada para que seja feito o *Sprint Backlog*, conforme é apresentado posteriormente.

Sprint Backlog

O *Sprint Backlog* é um conjunto de itens do *Produto Backlog* selecionados para a *Sprint*, juntamente com o plano para entregar o produto e atingir o objetivo da *Sprint*. O papel responsável por este artefato é o *Scrum Team*, que define tempo e a quantidade de itens para a primeira *Sprint Backlog*. Nessa etapa é necessária a participação constante do *Scrum Master*, com o intuito de liderar e monitorar a execução das atividades (SCHWABER e SUTHERLAND, 2014).

O *Sprint Backlog* é também caracterizado como a previsão do *Scrum Team* sobre quais funcionalidades estarão aptas a serem entregues como também sobre o que é necessário realizar para entregar essa funcionalidade em um incremento “pronto”. Como exemplo básico de um *Sprint Backlog*, é apresentada a Figura 7.

Figura 7 - Exemplo de *Sprint Backlog*

Sprint Backlog (20/04/2015)			
ID Product Backlog	Descrição Product Backlog	Sprint	Tempo
1	Pesquisar livro por autor	Sprint 01	7 dias
2	Pesquisar livro por titulo		
3	Pesquisar livro por editora		
5	Incluir um item no carrinho de compras		
4	Comprar um livro escolhido (um item por vez)	Sprint 02	9 dias
6	Pagamento via Boleto		
7	Pagar com Cartão de crédito		
8	Rastrear encomenda	Sprint 03	6 dias
9	Notificações de etapas das compras		
10	Feedback de atendimento após entrega		
...	...	Sprint 04	x dias
...	...		

Fonte: Adaptado de Tavares (2019)

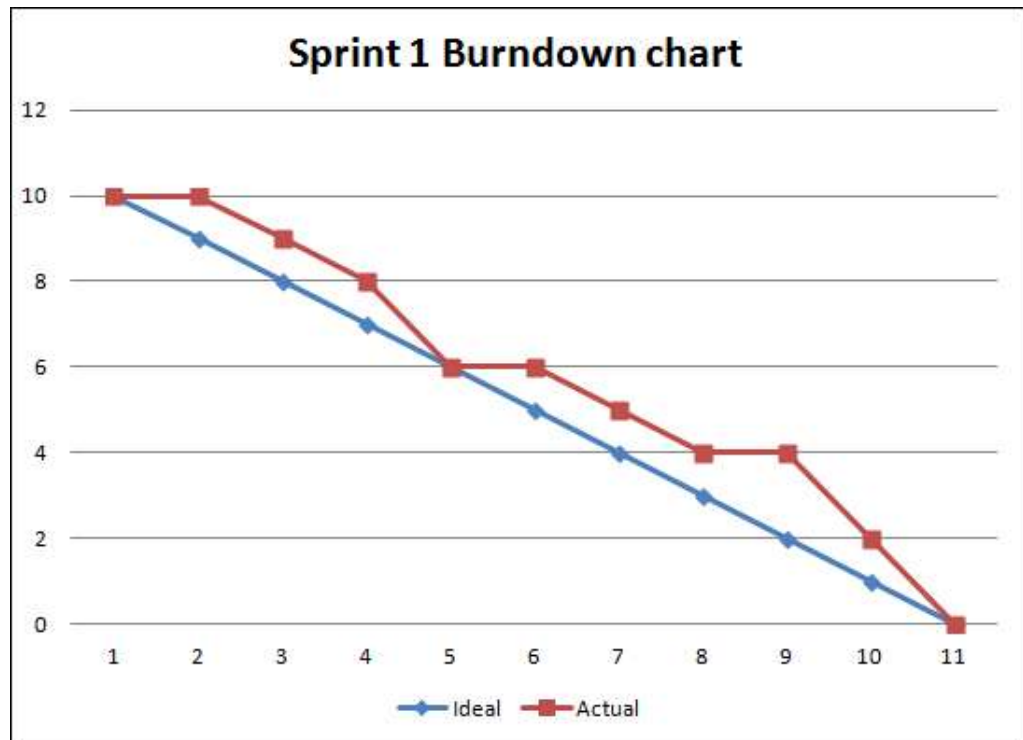
A Figura 7 representa a classificação do *Product Backlog*, de modo que os itens da lista sejam divididos em *Sprints* (*sprint01*, *sprint02*, etc.) para que sejam desenvolvidos em um determinado tempo e em sequência de prioridades definidas na cerimônia *Sprint Planning Meeting*. Cada *Sprint* tem seu tempo para ser concluída e esse progresso é medido visualmente através do *Burndown Chart*, que será apresentado a seguir.

Burndown Chart

O *Burndown Chart* mede o progresso da *sprint* e fornece indicativos do processo de trabalho da equipe. Além disso, é utilizado pelo *Scrum Team* para apresentar o progresso diário das equipes na execução da *Sprint*. Após cada dia de trabalho o *Burndown Chart* apresenta a porção de trabalho do time finalizada em comparação com o trabalho total planejado (SCHWABER e SUTHERLAND, 2014).

O *Burndown Chart* tem como principal objetivo medir os pontos das atividades finalizadas ao longo dos dias de execução da *Sprint* e ter uma visibilidade do ritmo de trabalho, verificando se o ritmo está adequado para atingir a meta da *sprint*, cumprindo com o que foi planejado. A Figura 8 demonstra um modelo de gráfico utilizado (LEIDEMER, 2013).

Figura 8 - Exemplo de *Burndown chart*



Fonte: Burndown (2019)

O gráfico de *Burndown chart* apresentado na Figura 8 demonstra o progresso ideal na cor azul e o progresso desenvolvido pelo *Scrum team* na cor vermelha. Esse progresso leva em consideração a quantidade de dias (na horizontal) e a quantidade de tarefas (na vertical) da *Sprint 1*.

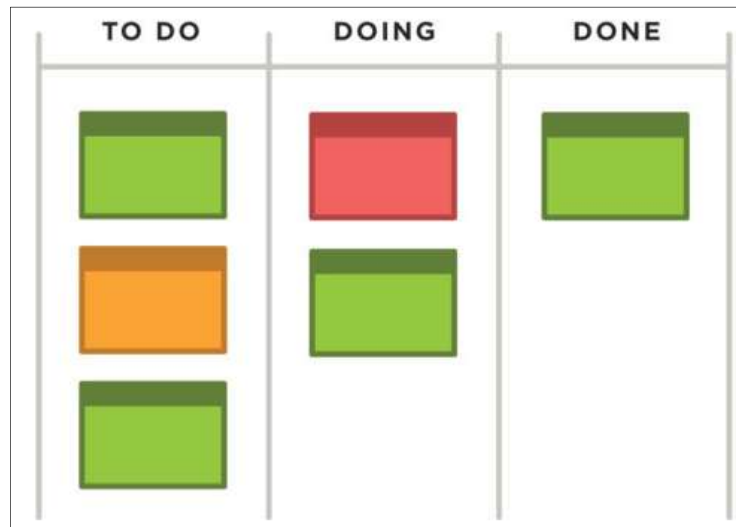
Task Board

O *Task Board* é uma visualização, em modelo de quadro, de todo progresso do trabalho que o *Scrum Team* está desenvolvendo durante uma *Sprint* (FUNARI, 2015).

Embora não faça parte do Guia do *Scrum* como artefato, o *Task board* é a visualização do processo de *Sprint Backlog*. Ou seja, quando necessita atualizar a *Sprint backlog*, seja adicionando alguma tarefa, removendo ou atualizando o seu status, o *Scrum Team* pode, por exemplo, simplesmente arrastar a tarefa para uma nova posição do quadro.

De acordo com Augusto (2011), o *Task Board* pode ser formado por três colunas, sendo elas: (i) **fazer**, que são as tarefas a que ainda não foram iniciadas; (ii) **fazendo**, referenciando as tarefas que estão em desenvolvimento pelo *Scrum Team*; e (iii) **feito**, que refere-se as tarefas concluídas sem pendências e/ou impedimentos. Exemplo de *Task Board* é apresentado na Figura 9 em seu modelo básico.

Figura 9 - Exemplo de *Task Board*



Fonte: Planview (2019)

A Figura 9 demonstra o exemplo do *Task Board* com três tarefa a “fazer” (*to do*), duas tarefas em “fazendo” (*doing*) e uma tarefa “feito” (*done*).

2.2.2.4. Cerimônias

O processo *Scrum* é formado por um conjunto de eventos de cerimônias, os quais serão detalhados nas seções a seguir a fim de possibilitar uma visão conceitual detalhada do ciclo de funcionamento dos eventos do processo.

Schwaber e Sutherland (2014) explicam que todo evento de cerimônia do *Scrum* tem uma duração máxima a ser cumprida, podendo terminar assim que o propósito do evento for alcançado, mas não depois do tempo máximo estabelecido.

Segundo MARÇAL et al. (2007) o ciclo do *Scrum* tem o seu progresso baseado em uma série de interações definidas, cada uma com duração em média de 2 a 4 semanas, chamadas *Sprints*. Inicialmente,

Antes de cada *Sprint*, realiza-se a *Sprint Planning Meeting* (Reunião de planejamento) onde o time de desenvolvedores tem contato com o cliente para priorizar o trabalho que precisa ser feito, selecionar e estimar as tarefas que o time pode realizar dentro da *Sprint*. A próxima fase é a Execução da *Sprint*. Durante a execução da *Sprint*, o time controla o andamento do desenvolvimento realizando as *Daily Scrum* (Reuniões Diárias), não mais que 15 minutos de duração, e observando o seu progresso. Ao final da *Sprint*, deve-se realizar a *Sprint Review* (Reunião de Revisão), onde o time demonstra o produto gerado na *Sprint* e valida se o objetivo foi atingido. Logo em seguida, realiza-se a *Sprint Retrospective* (Reunião de Retrospectiva), uma reunião de lições aprendidas, com o objetivo de melhorar o processo/time e/ou produto para a próxima *Sprint* (MARÇAL et al., 2007).

Baseado nisso, ao final de cada *Sprint* o produto que foi definido para ser desenvolvido deve estar pronto, codificado e testado.

Para demonstrar as funcionalidades e detalhes de cada cerimônia do *Scrum*, a seguir serão citadas e explicadas.

Sprint Planning Meeting

Sendo a primeira cerimônia realizada quando se inicia um novo projeto, a *Sprint Planning Meeting* tem *timebox* máximo de oito horas e o dever de planejar o trabalho a ser realizado durante uma determinada *Sprint*.

Conforme Marçal et al. (2007), a atividade que antecede o início da *Sprint* chama-se *Sprint Planning Meeting* (reunião de planejamento da *Sprint*). É uma atividade caracterizada como importante e que se deve ter atenção no seu objetivo principal durante o andamento da reunião. A equipe se prepara e tem por necessidade um período para planejar e estimar as atividades que serão desenvolvidas.

Segundo Schwaber e Sutherland (2014), a *Sprint Planning Meeting* tem como objetivo principal o entendimento e a definição do que pode ser entregue como incremento da próxima *Sprint*, além de decidir como serão realizadas e divididas as responsabilidades para a conclusão da *Sprint*.

Augusto (2011) descreve que durante o *Sprint Planning Meeting*, o PO explica e detalha os itens do *Product Backlog*, salientando também os itens que entendem ser relevante para a *Sprint* em planejamento. A equipe então deve determinar quais itens sejam possíveis de completar na referida *Sprint* e assim elaborar o *Sprint Backlog*, quebrando cada item do *Product Backlog* em uma ou mais tarefas detalhando o tempo de conclusão de cada uma.

A realização da *Sprint Planning Meeting* deve ser observada com clareza e os seus objetivos devem ser decididos com máximo de presteza e certeza possível, pois pode ser motivo de erros posteriores. Segundo Carvalho e Mello (2010), durante a aplicação da metodologia *Scrum* em uma empresa de Software, ao realizar a *Sprint Review Meeting* a equipe apontou vários erros, mais o considerado maior foi o erro ao definirem a *Sprint Backlog* com muitos itens, tal que resultou no atraso da *Sprint*.

Daily Scrum

Segundo Schwaber e Sutherland (2014) o acompanhamento do progresso da *Sprint* é feito realizando reuniões diárias. Essas reuniões devem ser rápidas, não mais que quinze minutos e devem ser objetivas, tendo como participantes o *Scrum Master* e o *Team Scrum*. No

Daily Scrum a equipe tem por objetivo responder três perguntas, que são respondidas por cada membro:

- 1) **O que foi feito desde ontem?**
- 2) **O que você planeja fazer para amanhã?**
- 3) **Teve algum impedimento durante ou nas atividades realizadas?**

Respondendo essas perguntas, todo e qualquer problema identificado durante o *Daily Scrum* deve ser tratado em uma outra reunião, à parte, apenas com os envolvidos.

De acordo com Carvalho e Mello (2010), o *Scrum Team* deve ter a habilidade de se auto gerenciar quando não for necessário a presença do *Scrum Master* para realização da reunião, pois assim pode evitar atrasos ou faltas de reuniões por motivos de ausência de algum membro. Carvalho e Mello (2010) também esclarece que a falta da *Daily Scrum* deve ser evitada o máximo possível.

Sprint Review

“Ao final da *Sprint* deve ocorrer uma *Sprint Review Meeting* (Reunião de Revisão da *Sprint*), onde a equipe demonstra o produto e faz a validação para verificar se o objetivo foi atingido” (CASTRO et al., 2017).

Marçal et al. (2007) também afirma que, ao final de cada *Sprint*, o objetivo é entregar o produto testado e revisado realizando uma demonstração prática. Este é o momento do *Product Owner* (cliente) inspecionar o produto final e verificar se o mesmo está como solicitado.

Carvalho e Mello (2010) esclarece que a demonstração deve ser informal, feita em formato de demo e com no máximo duas horas de duração, devendo se atentar as anotações do *Product Owner* e convidados que poderão se transformar em novos itens para o *Product Backlog*.

Sprint Retrospective

Após a execução da *Sprint*, ocorre a *Sprint Retrospective* (reunião de retrospectiva), cujo objetivo é melhorar o processo da equipe, é caracterizada como “lições aprendidas”. Nesta, a equipe levanta tudo que ocorreu durante a execução das tarefas e procura estabelecer pontos de melhoria. (SCHWABER E SUTHERLAND, 2014)

Marçal et al. (2007) enfatiza que o ciclo do *Scrum* é repetido até que todas as atividades tenham sido finalizadas, atendidas e/ou o produto tenha sido aceito pelo cliente. Portanto, se a *Sprint* for executada de acordo com o planejado, o resultado ao final implica em valor para o cliente.

Carvalho e Mello (2010) explica que a *Sprint Retrospective* é a última cerimônia do *Scrum* e é considerada uma reunião de lições aprendidas, onde pode-se responder as seguintes perguntas: O que foi bom? O que deve ser melhorado? E o que melhoramos? Assim, ao iniciar uma nova *Sprint*, deve-se sempre tentar melhorar cada vez mais.

Vale ressaltar que em nenhum dos eventos é previsto o uso de atas ou outro modelo de anotação e registro dos eventos, e isso é uma característica do *Scrum*, pois segundo Schwaber e Sutherland (2014), as reuniões são uma forma de reduzir a complexidade, fazer com que o *Scrum Team* trabalhe em conjunto e auto organizados para completar os objetivos propostos.

3. METODOLOGIA

A seguir serão descritos os materiais e métodos utilizados para o desenvolvimento deste trabalho.

3.1.MATERIAIS

Os materiais que foram utilizados no desenvolvimento deste trabalho são: *framework Django, PostgreSQL, Gitlab e API do Gitlab*.

O *Django* é um *Framework MVC (Model – View – Controller)* escrito em *Python*, de código aberto, publicado sob a licença BSD em 2005, e é utilizado para o desenvolvimento de *software web* (ERLO, 2012). *Django* foi utilizado para o desenvolvimento do *software GLSMeeting*, auxiliando na autenticação dos usuários, gerenciamento do sistema, formulários, *uploads*, consultas *backend* e manipulação de páginas HTML.

O *PostgreSQL* é um SGBD (Sistema Gerenciador de Banco de Dados) de código aberto, extensível e robusto. Segundo *The Postgresql Global Development Group* (2019) o *PostgreSQL* possui muitos recursos que ajudam os administradores proteger a integridade dos dados. O *PostgreSQL* foi utilizado para o armazenamento de dados do *GLSMeeting*.

O *GitLab* é um gerenciador de repositórios de *software* e tem como principal característica a permissão de armazenamento de códigos dos usuários diretamente em seus próprios servidores, ao invés de servidores de terceiros (GITLAB, 2019). *Gitlab* foi utilizado como sistema gerenciador de repositório vinculado a metodologia de uso das as cerimônias, onde será necessário a atualização e cadastros frequentes de todas as ações desenvolvidas acerca dos projetos no *Gitlab*, para que os dados a serem consumidos pelo *GLSMeeting* com o auxílio da *API do Gitlab* possam ser genuínos e atuais.

A *API* fornecida pelo *Gitlab* tem a função de acesso aos dados autorizados através de chave de acesso (*token*) fornecido pelo usuário no *Gitlab*. Neste trabalho, a *API* foi utilizada para o *login* e consumo de dados referente aos projetos da Fábrica de *Softwares* do CEULP/ULBRA.

3.2.MÉTODOS

O desenvolvimento do presente trabalho foi conduzido pelas etapas: reunião com especialista, análise do contexto da Fábrica de *software* definição da metodologia de uso das cerimônias e desenvolvimento do *software GLSMeeting*.

Figura 10 - Etapas de desenvolvimento dos resultados



As etapas ilustradas na Figura 10 são detalhadas a seguir:

- **Reuniões com Especialista:** inicialmente foram realizadas reuniões e entrevistas com o gerente de projetos da Fábrica de *Software* do CEULP/ULBRA para entendimento do contexto e da metodologia utilizada atualmente pela equipe. A reunião com especialista foi contínua, geralmente uma vez por semana, apresentando e buscando informações e sugestões para melhor utilização das cerimônias do *Scrum*;
- **Análise do contexto da Fábrica de Software:** foi a parte de análise das informações obtidas na reunião com especialista, verificando a utilização do *Scrum* na fábrica de *software*; o uso do *Gitlab* durante o desenvolvimento por um todo; o formato das reuniões e os registros (atas);
- **Definição da Metodologia *GLScrum*:** envolveu a definição das etapas e regras da metodologia de uso das cerimônias do *Scrum* com base na análise do contexto atual da Fábrica de *Software* do CEULP/ULBRA;
- **Desenvolvimento do *software GLSMeeting*:** em consonância com a definição da metodologia, o desenvolvimento do software envolveu as etapas de caso de uso, estrutura de acesso aos dados, arquitetura do *software* e definição dos *models* e *views* com a lógica do sistema, bem como os *templates* para a visualização por parte do usuário.

Com a execução das etapas citadas nessa secção, foi possível a obtenção de resultados conforme apresentado na secção seguinte.

4. RESULTADOS

Essa seção aborda informações sobre os resultados obtidos na realização deste trabalho. Para tanto é apresentada uma visão geral da utilização do *Scrum* na Fábrica de *Software* do CEULP/ULBRA. Na sequência é apresentada a visão geral do processo de desenvolvimento do *software GLSMeeting* e da metodologia *GLScrum*.

4.1. VISÃO GERAL DA APLICAÇÃO ATUAL DO SCRUM NA FÁBRICA DE SOFTWARE DO CEULP ULBRA

A equipe da fábrica de *Software* é responsável pelo desenvolvimento dos *softwares* e *websites* relacionados ao CEULP/ULBRA. Para tanto, faz uso de processos relacionados a metodologia de desenvolvimento *Scrum*, não se limitando a tal, no que se refere a seguir todos os processos sugeridos pela metodologia.

O modelo de desenvolvimento da fábrica de *software* é relacionado ao *Scrum* quanto às *Sprints*, cerimônias e aos *Backlogs*. Os papéis são os mesmos do *Scrum*, ou seja: *Scrum Master*, *Scrum Team* e *Product Owner*.

O *Scrum Team* e o *Scrum Master* possuem conhecimento do *Scrum* e desenvolvem os projetos procurando estar sempre de acordo com a metodologia. As reuniões são rápidas, pois o *Scrum Team* é composto por quatro pessoas, minimizando o processo de relatos sobre dúvidas e informações pertinentes. O *Product Backlog* e o *Sprint Backlog* são desenvolvidos e cadastrados no *Gitlab* como *issues* e *milestones*, respectivamente. As atas das reuniões são redigidas utilizando a plataforma *Google Docs* disponibilizada pela *Google*.

4.2. SOFTWARE GLSMEETING

Nesta seção serão apresentados os artefatos gerados para o desenvolvimento do *software GLSMeeting*, a ser utilizado como apoio as cerimônias. A criação dos artefatos seguiu o modelo UML (para os diagramas) como base para o projeto e para o desenvolvimento do sistema. Especificamente, os artefatos desenvolvidos foram: Arquitetura do *software*, estrutura de comunicação, diagrama de casos de uso e diagrama de classes. As seções seguintes apresentam o *GLSMeeting* e os artefatos de desenvolvimento.

4.2.1. Estrutura de acesso aos dados

Os dados do *GLSMeeting* são manipulados com o uso do SGBD *PostgreSQL* e pela *API Gitlab*. A *API Gitlab* é acessada pelo *GLSMeeting* através de solicitações HTTP, obtendo como

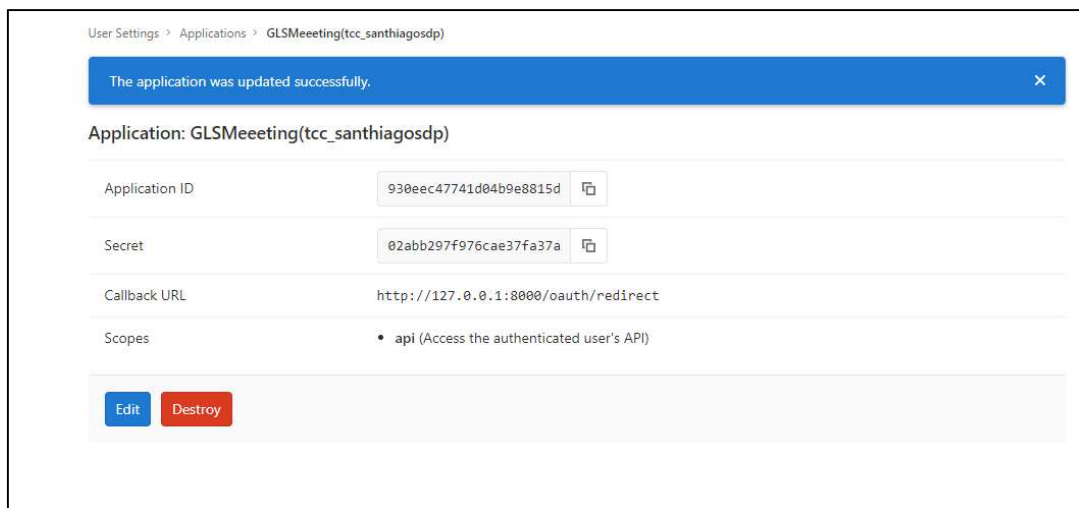
retorno objetos no formato *Json*, facilitando a manipulação dos dados no projeto. Os dados do *PostgreSQL* são acessados por meio do ORM do *Django*.

4.2.1.1. Configuração da aplicação no *Gitlab*

O *Software GLSMeeting* utiliza como forma de autenticação o login delegado pelo *Gitlab*. Para configuração do login através da plataforma *Gitlab*, é necessário o cadastro do *software GLSMeeting* como aplicativo no *Gitlab* para utilização do provedor *OAuth Gitlab*. O cadastro foi realizado por meio da tela de cadastro de aplicativo no *Gitlab*, informando os dados: (a) nome, representado o nome do aplicativo que acessará a conta do usuário ("*GLSMeeting*"); (b) URL de redirecionamento, que indica para qual link deve ser retornada a resposta de autenticação do usuário ("*http://127.0.0.1:8000/oauth/redirect*") e (c) escopos, que indicam quais dados podem ser acessados pelo software. Neste caso, o escopo "*api*" foi utilizado para permitir acesso completo aos dados, tanto leitura como gravação.

Na conclusão do cadastro o *Gitlab* informa dados necessários para o funcionamento do *software*, conforme a Figura 11.

Figura 11 – Exemplo de cadastro de Aplicação para utilização do *Gitlab* como Provedor *OAuth*

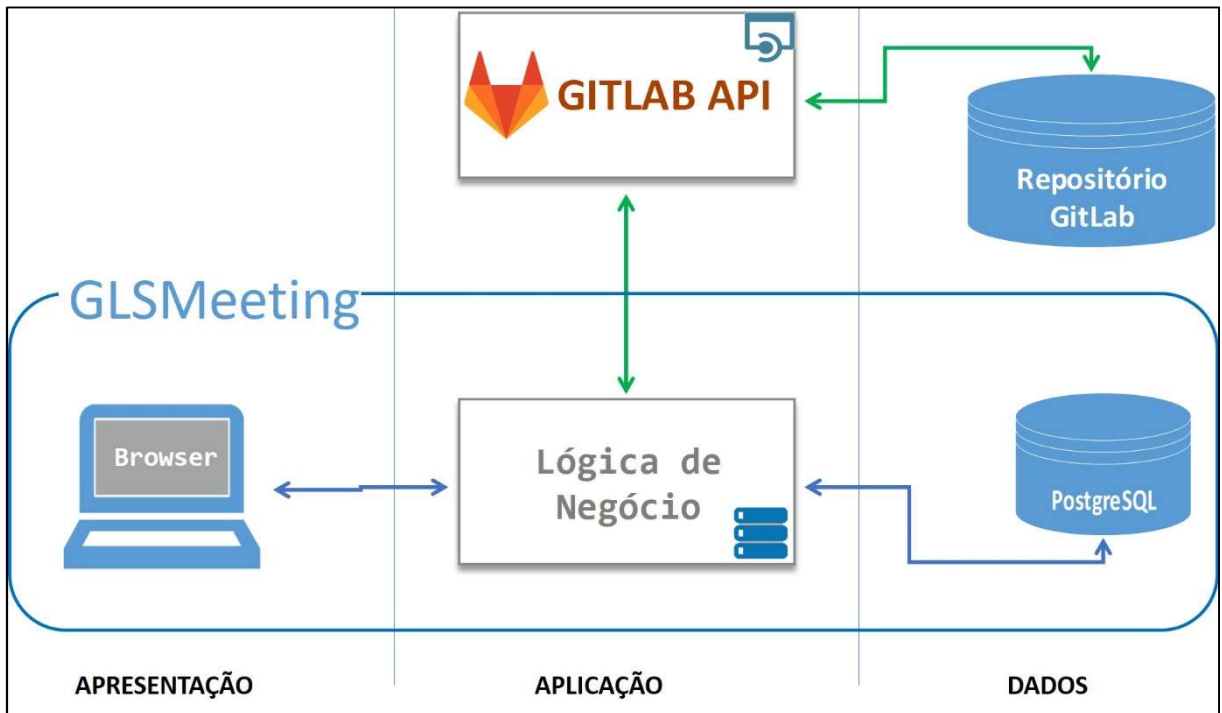


Os dados apresentados na Figura 11 indicam: a) o identificador da aplicação (*Application ID*); b) uma senha da aplicação (*Secret*); c) o *link* de redirecionamento (*Callback URL*); e d) os escopo que o usuário deve permitir que a aplicação acesse (*Scopes*).

Visão Geral da Integração entre *GLSMeeting* e *Gitlab*

A integração com o *Gitlab* é realizada com a utilização da *API Gitlab*, conforme ilustrado na Figura 12.

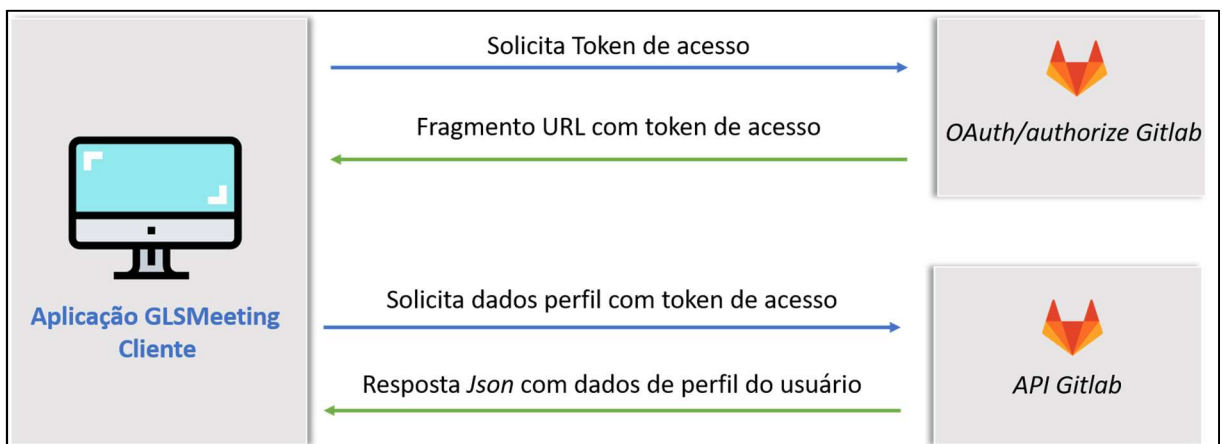
Figura 12 - Arquitetura de acesso a dados do *GLSMeeting*



A Figura 12 apresenta a estrutura de acesso aos dados armazenados no gerenciador de banco de dados *PostgreSQL* e no repositório *Gitlab*.

O primeiro acesso feito pelo *GLSMeeting* à *API Gitlab* é a autenticação por meio do *OAuth*, conforme fluxo apresentado na Figura 13.

Figura 13 - Fluxo de autenticação *OAuth Gitlab*



A Figura 13 apresenta o fluxo de autenticação do usuário no *GLSMeeting*, no qual o usuário é redirecionado para página de login do *Gitlab* com parâmetros de configuração da aplicação *GLSMeeting*, solicitando token de acesso, que é um código alfanumérico utilizado

como chave secreta para acesso aos dados de repositório e perfil do usuário no *Gitlab*. Depois de o usuário se autenticar no *Gitlab* com sucesso, ele é redirecionado para a URL informada como parâmetro, acompanhada de token de acesso. O Trecho de código 1 ilustra a URL de redirecionamento.

Trecho de código 1 – Exemplo de URL de redirecionamento após usuário autenticar no *OAuth Gitlab*

```
http://127.0.0.1:8000/oauth/redirect/#access_token=d62cee3ffe4dsd4ks74gffgd8sf
```

O Trecho de código 1 apresenta o fragmento “*access_token*” retornado na *URL*, que é usado para o *software* ter acesso aos dados do perfil e de repositório do usuário através da *API Gitlab*. O acesso do usuário ao *software GLSMeeting* é autorizado após a requisição de perfil do usuário utilizando o token de acesso retornado através da *API Gitlab*. Ao obter o perfil, é feito cadastro no *PostreeSQL* que é vinculado ao token de acesso para obter dados do *gitlab*.

As requisições de informações do repositório *gitlab* do usuário também são feitas com a utilização da *API Gitlab* informando o token de acesso, solicitando os grupos, projetos e *issues* que o usuário tem acesso.

Quando se faz a requisição de um grupo específico, é necessário informar o “*id*” do grupo e o token de acesso do usuário, tendo como resposta as informações do grupo. As principais informações de um grupo são:

- (a) *Id*, indica o código identificador do grupo;
- (b) *Name*, que apresenta o nome do grupo,
- (c) *Description*, que apresenta a descrição dos detalhes do grupo;
- (d) *Projects*, que apresenta um *array* com todos os projetos cadastrados no grupo.

Os projetos são solicitados de acordo com o grupo específico, pois estará presente no *array* retornado dentro do grupo. Portanto, após o retorno do grupo específico, é necessário acessar o *array* e buscar o projeto. As principais informações de um projeto são:

- (a) *Id*, indica o código identificador do projeto;
- (b) *Name*, que apresenta o nome do projeto;
- (c) *Description*, que apresenta a descrição dos detalhes do projeto.

As *issues* também são solicitadas de acordo com o grupo específico. As principais informações de uma *issue* são:

- (a) *Id*, indica o código identificador da *issue*;

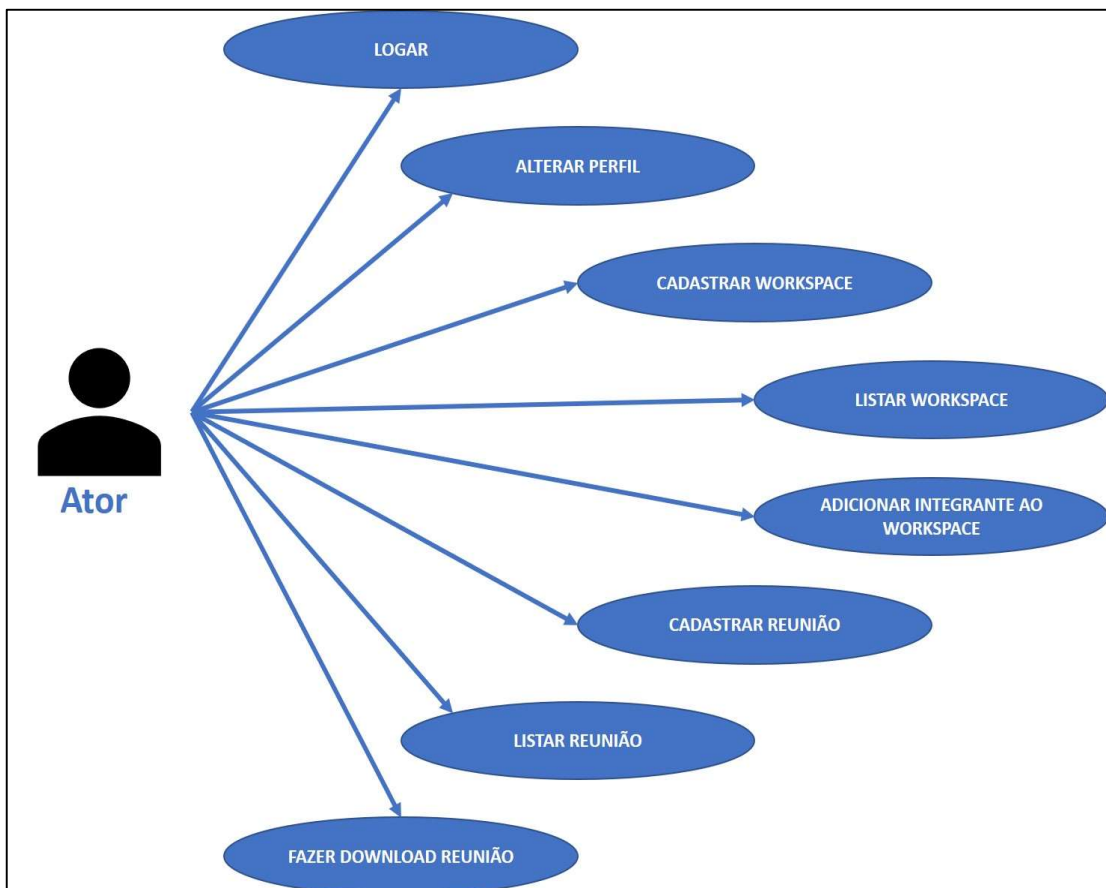
- (b) *project_id*, indica o identificador do projeto que a *issue* pertence;
- (c) *title*, que apresenta o nome/título da *issue*;
- (d) *state*, que apresenta o estado da *issue* (aberta/fechada).

As principais informações apresentadas aqui foram utilizadas no *software GLSMeeting* para cadastros dos *Workspaces* e das atas de reunião. Na sequência serão apresentados os artefatos, a estrutura e as funcionalidades do *GLSMeeting*.

4.2.2. Diagrama de Caso de Uso

O diagrama de caso de uso apresenta as funcionalidades que compõe o *GLSMeeting* e as interações dessas funcionalidades com os usuários, como demonstrado na Figura 14.

Figura 14- Diagrama de caso de uso do *GLSMeeting*



A Figura 14 apresenta o diagrama de caso de uso, no qual pode-se observar que o usuário possui as funcionalidades de *logar*, *cadastrar* e *listar* procedimentos relacionados ao *software GLSMeeting*. O detalhamento de cada caso de uso está demonstrado na Tabela 1:

Tabela 1- Detalhamento dos casos de uso

CASO DE USO	TIPO	ATORES	DESCRIÇÃO
Logar	Primário	Usuário	O <i>software</i> deve permitir que o usuário possa logar usando a sua conta do Gitlab através do Oauth da API Gitlab .
Alterar perfil	Primário	Usuário	O <i>software</i> deve permitir que o usuário altere seu perfil após se logar no sistema. Deve ser possível alterar o nome que será manipulado pelo GLSMeeting .
Cadastrar <i>workspace</i>	Primário	Usuário	O <i>software</i> deve permitir que o usuário efetue o cadastro de <i>Workspace</i> no GLSMeeting . Esse cadastro deve solicitar que usuário informe o “nome” do <i>Workspace</i> e o código “id” do grupo cadastrado no Gitlab . Workspace é o espaço onde as atas serão criadas, e com o código “id” do grupo no Gitlab deve permitir que dados sejam solicitados de forma indexada.
Listar <i>workspace</i>	Primário	Usuário	O <i>software</i> deve permitir que o usuário possa listar os <i>Workspaces</i> no GLSMeeting . A lista deve apresentar o nome de cada <i>Workspace</i> .
Adicionar integrante ao <i>Workspace</i>	Primário	Usuário	O <i>software</i> deve permitir que o usuário cadastre convites de integrantes para um <i>Workspace</i> no GLSMeeting . Esse cadastro deve solicitar que o usuário informe o “e-mail” do convidado.
Cadastrar Reuniões	Primário	Usuário	O <i>software</i> deve permitir que o usuário efetue o cadastro de atas de reuniões no GLSMeeting . Esse cadastro deve solicitar que usuário informe os seguintes campos: data de início, lista de presentes, projeto, Assuntos/Descrição e data de encerramento.
Listar reuniões	Primário	Usuário	O <i>software</i> deve permitir que o usuário liste as atas de reuniões no GLSMeeting . A lista deve ter os seguintes campos: data de início, Tipo da Reunião e Projeto.

Fazer <i>Download</i> de reunião	Primário	Usuário	Na tela de lista das atas de reunião, o usuário poderá clicar na opção de <i>download</i> da ata de reunião.
----------------------------------	----------	---------	--

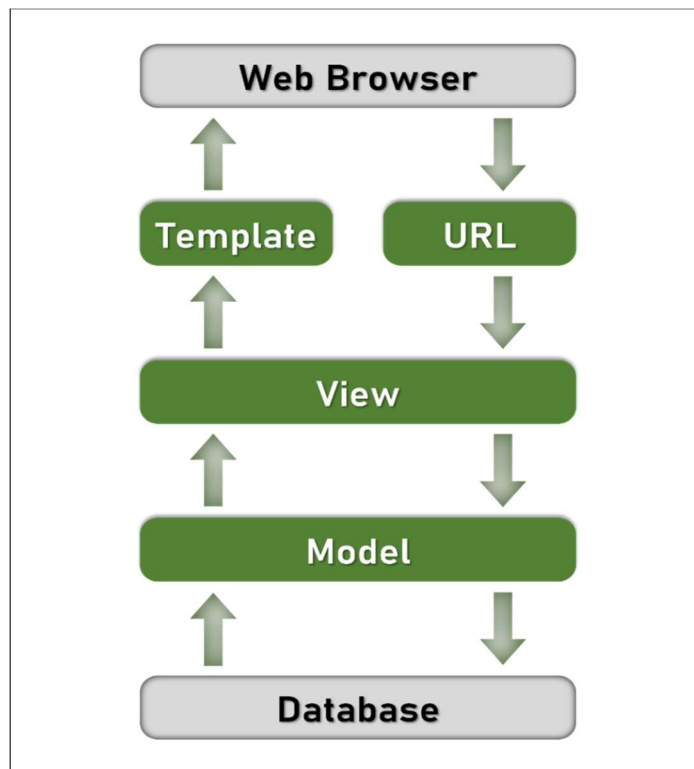
A Tabela 1 apresenta o diagrama de caso de uso que representa a interação do usuário com o *software GLSMeeting*. Foi importante o entendimento das funcionalidades do *software* para o planejamento e implementação dos requisitos, pois o caso e uso apresenta os cenários a serem desenvolvidos. Portanto, desenvolveu-se a arquitetura de software conforme apresentado na seção seguinte.

4.2.3. Arquitetura do software

O *framework Django* utiliza o padrão de desenvolvimento MTV (*Model, Template e View*) que disponibiliza benefícios para facilidade de desenvolvimento do projeto. A arquitetura MTV é um padrão de desenvolvimento que o separa em três camadas, *models.py* onde é implementada a camada que representa o banco de dados, *templates*, camada responsável por armazenar a interface do sistema (páginas *html*) e *views.py* que é responsável pelas regras do negócio, ou seja, processamento dos dados vindo da camada *model* e direcionando-os para a camada *template*.

A utilização do *Django* faz com que o projeto possua uma estrutura de pastas com os principais arquivos necessários para a execução.

Figura 15 - Arquitetura do *Django*

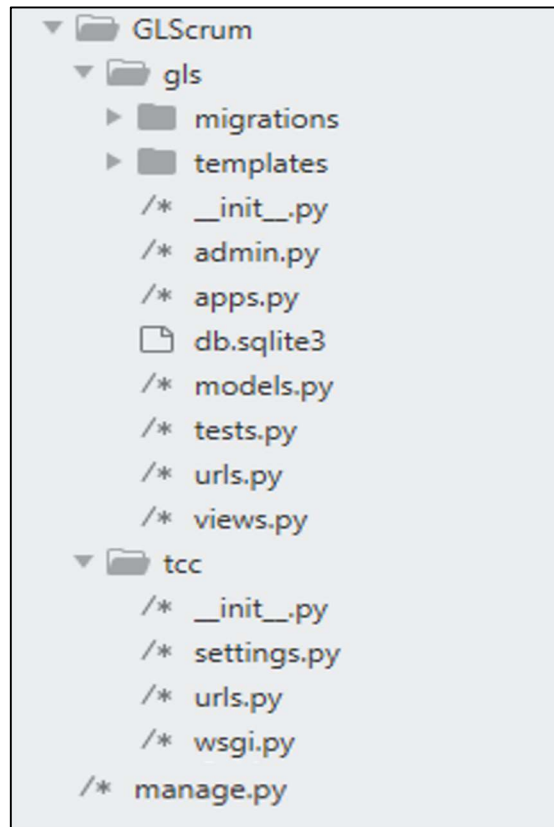


Fonte: Adaptado de Furtado (2016)

A Figura 15 apresenta a arquitetura do *Django* apresentando as camadas MTV desde uma comunicação com o *Web Browser* até o acesso ao *Database*. O **URL** identifica o endereço requisitado pelo usuário e realiza o redirecionamento para a *view* correspondente, a *view* por sua vez, de acordo com a implementação, faz o acesso ao banco de dados através do *model* e retorna para o *template* as informações pertinentes a apresentação. O acesso ao banco de dados só é feito caso seja necessário a busca de informações, previamente implementadas na *view* em questão.

A estrutura das principais pastas e dos principais arquivos do *GLSMeeting* é apresentada na Figura 16.

Figura 16 - Estrutura das principais pastas e principais arquivos do *GLSMeeting*

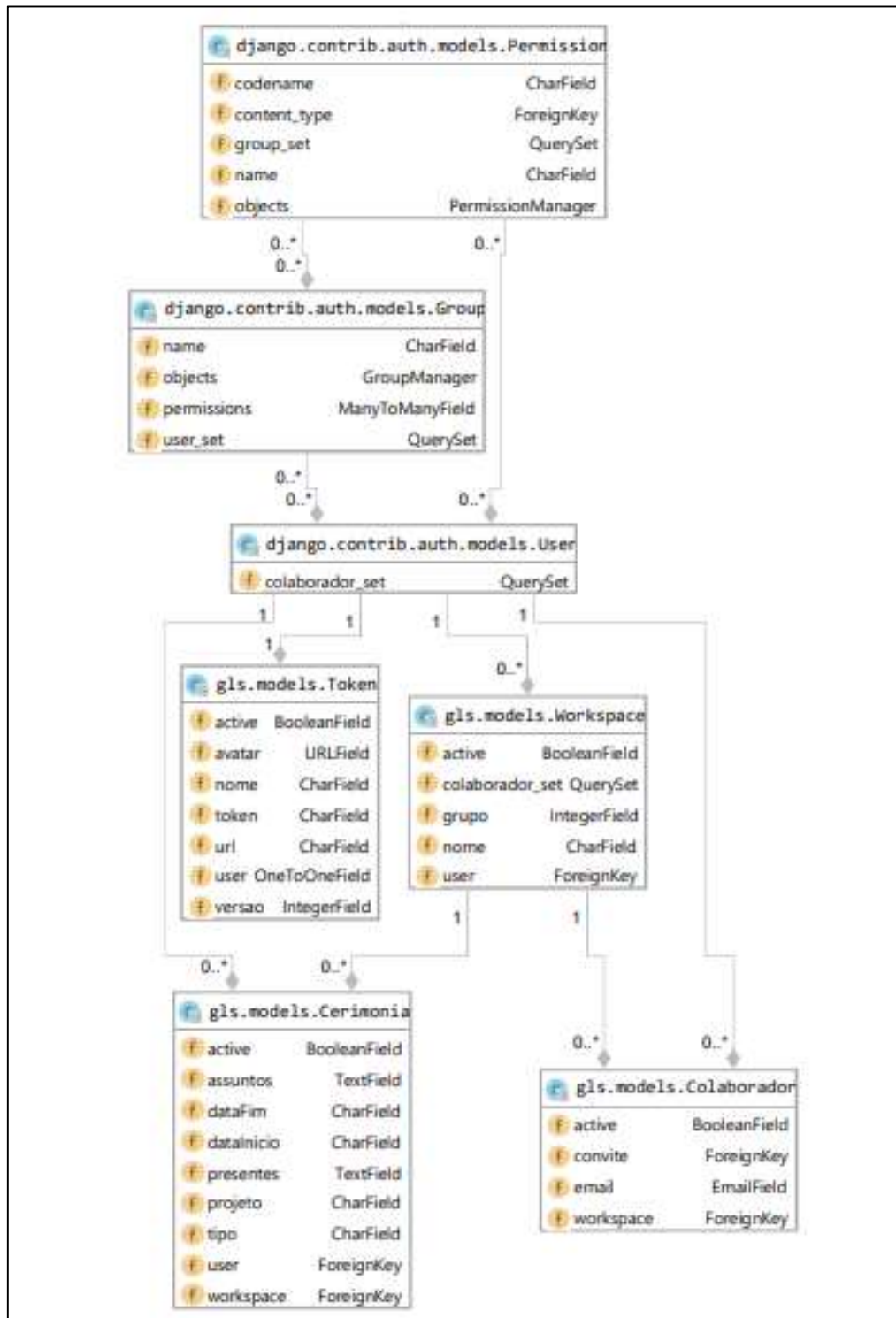


Conforme a Figura 16, é importante ressaltar os arquivos *gls/models.py* e *gls/views.py*, conforme abordado nos tópicos seguintes.

4.2.3.1. Models

O arquivo *gls/models.py* contém as informações dos dados que estão sendo armazenados e é o responsável pelo mapeamento objeto-relacional, portanto não há necessidade de comandos da linguagem SQL.

Figura 17 - Diagrama de dependência do *model* do software *GLSMeeting*



A Figura 17 apresenta o diagrama de dependências do *model* do *software* que é composto por 7 *models* que se relacionam. A Tabela 2 define cada um dos *models*.

Tabela 2 – Definição dos *models* implementados em *gls/models.py*

Model	Descrição
<code>django.contrib.auth.models.Permission</code>	O <i>model Permission</i> está disponível no pacote de autenticação nativo do <i>Django</i> e fornece uma maneira de atribuir permissões a usuários e grupos de usuários específicos.
<code>django.contrib.auth.models.Group</code>	O <i>model Group</i> está disponível no pacote de autenticação nativo do <i>Django</i> e tem a função de categorização dos usuários para aplicação de permissões para um grupo em específico.
<code>django.contrib.auth.models.User</code>	O <i>model User</i> está disponível no pacote de autenticação nativo do <i>Django</i> . Fornece a estrutura necessária para cadastro de usuário no sistema e tem o relacionamento muitos para muitos já definido com os <i>models</i> de <i>Permission</i> e <i>Groups</i> .
<code>gls.models.Token</code>	O <i>model Token</i> foi implementado no sistema <i>GLSMeeting</i> com a função de vincular um usuário do <i>Oauth Gitlab</i> com um <i>User</i> criado no sistema para gerenciamento local. Esse relacionamento é de “um para um” no qual cada <i>User</i> do sistema pode ter relacionamento com somente um <i>Token</i> , e vice-versa. Os atributos do <i>model Token</i> são armazenados conforme dados recebidos do <i>Oauth Gitlab</i> e somente o atributo “ <i>nome</i> ” pode ser alterado.
<code>gls.models.Workspace</code>	O <i>model Workspace</i> foi implementado no sistema <i>GLSMeeting</i> com a função do relacionar um <i>User</i> com muitos <i>Workspaces</i> , enquanto um <i>Workspace</i> só pode ser criado por um <i>User</i> . Um <i>Workspace</i> com muitos <i>Colaboradores</i> . Uma <i>Cerimonia</i> com uma <i>Workspace</i> , enquanto uma <i>Workspace</i> com muitas <i>Cerimonias</i> .
<code>gls.models.Colaborador</code>	O <i>model Colaborador</i> foi implementado no sistema <i>GLSMeeting</i> com a função de relacionar um <i>User</i> com muitos <i>Colaborador</i> (convite) e um <i>Workspace</i> com muitos <i>colaborador</i> .
<code>gls.models.Cerimonia</code>	O <i>model Cerimonia</i> foi implementado no sistema <i>GLSMeeting</i> com a função de relacionar um <i>User</i> com muitas <i>Cerimonias</i> e um <i>Workspace</i> com muitas <i>Cerimonias</i> .

Para ilustrar o conteúdo de um *model*, o Trecho de código 2 apresenta o *model Cerimonia*.

Trecho de código 2 - *Model Cerimonia*

```
class Cerimonia(models.Model):
    workspace = models.ForeignKey(Workspace, on_delete=models.CASCADE)
    tipo = models.CharField(max_length=100)
    dataInicio = models.CharField(max_length=100)
    dataFim = models.CharField(max_length=100)
```

```
presentes = models.TextField()
projeto = models.CharField(max_length=100)
assuntos = models.TextField()
active = models.BooleanField(default=True)
user = models.ForeignKey(User, null = True, on_delete=models.SET_NULL)
```

O Trecho de código 2 apresenta o *model Cerimonia*, que utiliza o módulo *models*, do pacote *django.db*, já disponível com o próprio *django*. Neste *model* há os atributos:

- a) *workspace*, do tipo *ForeignKey*, que representa um relacionamento com o *model workspace* com deleção em cascata;
- b) *tipo*, do tipo *CharField*, que representa se é reunião de planejamento, diária ou final, tem limite de cem caracteres;
- c) *dataInicio*, do tipo *CharField*, que representa a data de início da cerimônia, tem limite de cinquenta caracteres;
- d) *dataFim*, do tipo *CharField*, que representa a data de encerramento da cerimônia, tem limite de cinquenta caracteres;
- e) *presentes*, do tipo *CharField*, que representa os participantes da cerimônia, tem limite de cem caracteres;
- f) *projeto*, do tipo *CharField*, que representa o projeto tratado na cerimônia, tem limite de cem caracteres;
- g) *assuntos*, do tipo *TextField*, indica os assuntos ou descrição tratadas na cerimônia.
- h) *active*, do tipo *BooleanField*, com valor padrão *True*, indica se o workspace está habilitado ou não;
- i) *user*, do tipo *ForeignKey*, que representa um relacionamento com o *model User* (do pacote *django-admin*) com deleção de relacionamento.

Os dados a serem armazenados de acordo com o *model* definido são manipulados no arquivo *gls/views.py*, que será explicado no tópico seguinte.

4.2.3.2. Views

A camada de *views* do *software GLSMeeting* é representada pelo arquivo *gls/views.py*, que é responsável pela lógica de negócio para o gerenciamento de dados a serem visualizados na camada. Para detalhamento de todos os métodos implementados no *GLSMeeting*, a Tabela 3 apresenta os nomes dos métodos que representam as *views* implementadas e a descrição de cada um.

Tabela 3 – Descrição dos métodos implementados em *gls/views.py*

Nome da view	Descrição	URL
login_user	Apresentação da página para o usuário logar no <i>software GLSMeeting</i>	/login/
submit_login	Ação de redirecionamento para URL de <i>OAuth</i> do <i>Gitlab</i> com parâmetros da aplicação.	/login/submit/
autenticar	Ação de autenticar/cadastrar usuário no <i>software GLSMeeting</i> .	/autenticar
logout_user	Ação de <i>logout</i> do usuário no <i>software GLSMeeting</i>	/logout
inicial	Apresentação da página inicial do <i>software</i> com as <i>Workspaces</i> habilitadas.	/ws
perfil	Apresentação da página para alterar perfil do usuário.	/perfil
submit_perfil	Ação de confirmar alteração do perfil do usuário.	/perfil/submit
nova_workspace	Apresentação da página de cadastrar <i>Workspace</i> .	/ws/new
submit_workspace	Ação de cadastrar <i>Workspace</i> no banco de dados.	/ws/submit
workspaces_desabilitadas	Apresentação da página com <i>Workspaces</i> desabilitadas com opção de habilitar.	/ws/desabilitadas
desabilitar_workspace	Ação de desabilitar uma <i>Workspace</i> .	/desabilitar_ws/<id>
habilitar_workspace	Ação de habilitar uma <i>Workspace</i> .	/habilitar_ws/<id>
lista_atas	Apresentação da página com lista das atas da <i>Workspace</i> selecionado.	/ws/<id>
deletar_ata	Ação de deletar uma ata do banco de dados.	/delete_ata/<id>/<id_ata>
novo_integrante	Apresentação da página de cadastro de um integrante para o <i>Workspace</i> selecionado.	/ws/<id>/new_integrante
submit_integrante	Ação de cadastrar um integrando no banco de dados.	/ws/<id>/new_integrante/submit
deletar_integrante	Ação de deletar um integrante do banco de dados.	/delete_integrante/<idw>/<idi>

resposta_convite	Apresentação da página para o usuário aceitar ou recusar o convite para participar de um <i>Workspace</i> .	/convite
submit_convite	Ação de aceitar ou recusar o convite para participar de um <i>Workspace</i> .	/resposta_convite/<id>
nova_planejamento	Apresentação da página de cadastro de uma nova ata de reunião de planejamento no <i>Workspace</i> selecionado.	/new/<id>/planejamento
nova_diaria	Apresentação da página de cadastro de uma nova ata de reunião diária no <i>Workspace</i> selecionado.	/new/<id>/diaria
nova_final	Apresentação da página de cadastro de uma nova ata de reunião final no <i>Workspace</i> selecionado.	/new/<id>/final
submit_reuniao	Ação de cadastro da ata de reunião no banco de dados.	/new/<id>/cerimonia_save
imprimir_pdf	Ação de renderização da página <i>html</i> com uma ata de reunião a ser impresso no formato PDF.	print/<id>
imprimir_pdf_data	Ação de renderização da página <i>html</i> com uma ou várias atas de reunião a ser impresso no formato PDF.	/data/print
render_to_pdf	Ação de impressão e visualização da página <i>html</i> em formato PDF.	Não se aplica

As informações contidas na Tabela 3 descrevem as funções de cada método do arquivo *gls/views.py*. Os métodos implementam as funções de gerenciamento de dados (adicionar, consultar, alterar ou excluir). A seguir, serão apresentadas as *views* “*login_user*” e “*autenticar*” para o demonstrar a forma de login no *GLSMeeting* e a *view* “*nova_diaria*”, demonstrando a visualização da página de cadastro da ata diária.

View *login_user*

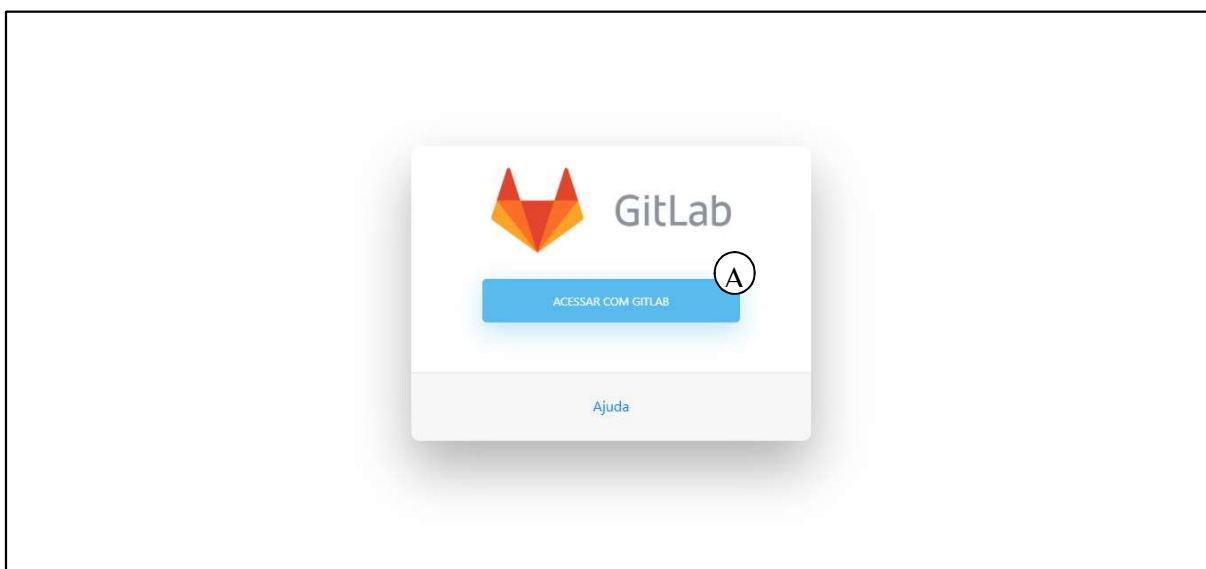
A *view login_user* é responsável por apresentar a página de *login* do *GLSMeeting* e seu código é apresentado no Trecho de código 3.

Trecho de código 3 – *View “login_user”* para apresentação da página de *login*

```
def login_user(request):  
    return render(request, 'login.html')
```

O Trecho de código 3 representa a *view login_user*, que utiliza o *template gls/templates/login.html*, e gera a página de login, conforme Figura 18.

Figura 18 - Página para *login* do software *GLSMeeting*



Conforme representa a Figura 18, o usuário deve fazer login no sistema *GLSMeeting* utilizando uma conta *Gitlab*. Ao clicar no botão “*acessar com Gitlab*” (Figura 18-A), o usuário será direcionado para plataforma do *Gitlab* onde realizará a autenticação.

Figura 19 - Página de login no *Gitlab*

GitLab.com

GitLab.com offers free unlimited (private) repositories and unlimited collaborators.

- [Explore projects on GitLab.com](#) (no login needed)
- [More information about GitLab.com](#)
- [GitLab.com Support Forum](#)
- [GitLab Homepage](#)

By signing up for and by signing in to this service you accept our:

- [Privacy policy](#)
- [GitLab.com Terms.](#)

Sign in Register

Username or email

Password

Remember me [Forgot your password?](#)

Sign in

Sign in with

Google Twitter

GitHub Bitbucket

Salesforce

A Figura 19 apresenta a página de *login* no *gitlab*, no qual usuário deve informar suas credenciais para autenticação *OAuth* e redirecionamento para a *view* “*autenticar*” do *GLSMeeting* que será explicada na sequência.

View autenticar

A *view* “*autenticar*” faz requisição de perfil do usuário através da *API Gitlab*. A requisição obtém como resposta informações em formato *Json* dos dados do usuário. Dentre as informações retornadas, as mais importantes são: “*id*”, contendo o número de identificação do usuário, “*name*”, contendo o nome completo do usuário, “*username*”, contendo o nome de usuário, “*avatar*”, contendo o link para acesso ao avatar do usuário e “*email*”, contendo o e-mail de cadastro do usuário.

Os dados de perfil de usuário são usados para autenticação do usuário no *GLSMeeting*, conforme apresentado no Trecho de código 4.

Trecho de código 4 – Criar autenticação do usuário no *GLSMeeting*

```
def autenticar(request):  
    ...  
    user = authenticate(username=username, password=senha)  
    if user is not None:
```

```
login(request, user)
return redirect('/ws')
...
```

O Trecho de código 4 define a variável “*user*” através do método *authenticate()* disponível no pacote *django.contrib.auth* nativo do *Django*, tendo como parâmetros “*nome_usuario*” e “*senha*”. Se o usuário autenticado no *Gitlab OAuth* já tiver acessado o *GLSMeeting* anteriormente, ele será autenticado e redirecionado para view “*workspaces*”. Caso seja o primeiro acesso, será feito o vínculo da autenticação no *OAuth Gitlab* com um usuário no *GLSMeeting*, conforme apresenta o Trecho de código 5.

Trecho de código 5 - Criando usuário no *GLSMeeting* com vínculo ao *OAuth Gitlab*

```
def autenticar(request):
...
    else:
        new_user = User.objects.create_user(first_name=name,
                                             username=username, email=email, password=senha)
        new_user.save()
        user = authenticate(username=nome_usuario, password=senha)
        login(request, user)
        return redirect('/perfil')
...
```

O Trecho de código 5 cadastra o usuário no *GLSMeeting* com os dados de perfil retornado e faz a autenticação. Para o cadastro é utilizado o *Model User* disponível no *Django* e o método *create_user()*, indicando o os dados de perfil e senha do usuário. Em seguida o cadastro do usuário é concluído utilizado o método *save()* e então é feito a autenticação através do método *authenticate()*.

View nova_diaria

O Trecho de código 6 apresenta o view “*nova_diaria*” responsável por gerar a página de cadastro de uma reunião diária.

Trecho de código 6 – View “*nova_diaria*” para apresentação da página de cadastro de reunião diária

```
def nova_diaria(request, id):
    colaboradores = Colaborador.objects.filter(workspace=id)
    workspace = Workspace.objects.get(id=id)
    grupo_especifico = grupo_gl(aces_token, workspace.grupo)
    projects = grupo_especifico.projects.list()
    issues = grupo_especifico.issues.list()
    return render(request, 'diaria.html',
                  {'time': colaboradores, 'projects': projects, 'issues': issues})
```

O Trecho de código 6 apresenta a *view* para cadastro de reunião diária, no qual primeiramente são obtidos os colaboradores do *Workspace* em questão. Isso é feito utilizando o *model* Colaborador e o método *filter()*, indicando o identificador do *Workspace*. Posteriormente é obtido o *Workspace* em questão utilizando o *model* *Workspace* e o método *get()*, indicando o identificador do *Workspace*. Em seguida é obtido o grupo utilizando o método *grupo_gl()* da *API Gitlab*, indicando o token de acesso do usuário e o identificador do grupo no repositório *Gitlab*. Após, é obtido os projetos e *issues* do grupo, utilizando o método *Project.list()* e *issues.list()*, respectivamente. Com a utilização do *template* *gls/templates/diaria.html*, é utilizado o método *render()* gerar a página de cadastro de ata da reunião diária.

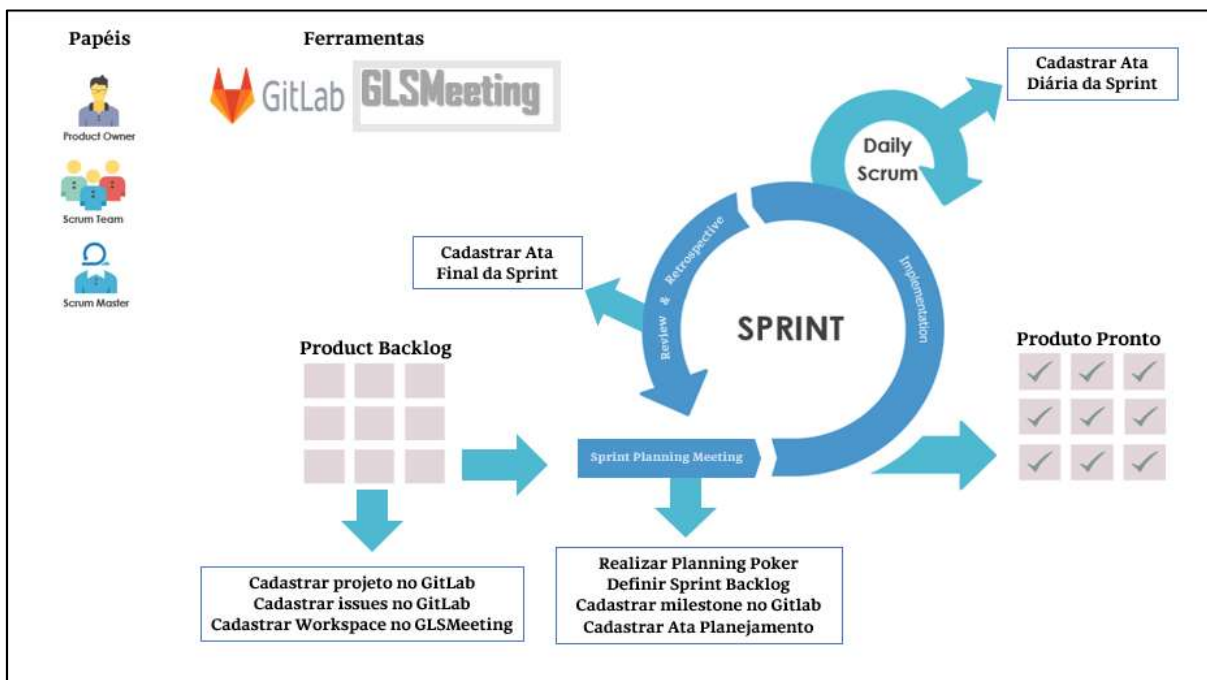
Portanto, o arquivo *gls/views.py* é responsável por toda a lógica de negócio no que se refere a solicitações de consultas, cadastros, alterações etc. do projeto *GLSMeeting*. As funcionalidades do *software GLSMeeting* serão abordadas de forma visual no próximo tópico de modo a demonstrar sua utilização com a proposta de metodologia elaborada neste trabalho.

4.3. GLSCRUM - METODOLOGIA DE USO DAS CERIMÔNIAS

Esse tópico abrange os procedimentos necessários para desenvolvimento das cerimônias, reduzindo as necessidades de se apresentar erros de registros ou ineficácia dos métodos a serem utilizados para realização da cerimônia relacionada a metodologia *Scrum*.

A *GLScrum* faz uso das abordagens definidas na metodologia *Scrum*, detalhando procedimentos que são necessários para a melhor realização das cerimônias e gerenciamento do projeto em desenvolvimento.

Figura 20 - Processo da metodologia de uso das cerimônias



A Figura 20 demonstra o processo de utilização da metodologia de uso das cerimônias. É necessário o uso das ferramentas *GLSMeeting* (desenvolvido como parte deste trabalho) e *Gitlab*, além da adoção da técnica de *Planning Poker* para determinar o consenso da equipe, estimando as metas do desenvolvimento de cada *milestone*.

A metodologia de uso das cerimônias consiste nos processos do *Scrum* associados a procedimentos de cadastros, atualizações e registros informatizados com uso do *Gitlab*, bem como a utilização do *GLSMeeting* para cadastros e gerenciamentos das atas de reuniões.

4.3.1. Papéis

Os três papéis atribuídos no *Scrum* se fazem presentes também na metodologia de uso das cerimônias, onde, além das responsabilidades já especificadas no *Scrum*, devem efetuar procedimentos necessários para a realização das cerimônias, conforme explorados a seguir.

Product Owner

O papel do *Product Owner* continua inalterado em relação a sua função do *Scrum*. Sendo opcionalmente convidado para participar da reunião de Planejamento e assim ser adicionado seu nome na ata de Planejamento da *Sprint*.

Scrum Team

O *Scrum team*, como já observado pela metodologia *Scrum*, tem a responsabilidade coletiva pelo sucesso da interação e do produto pronto. No entanto, considerando a necessidade de orientação para o “norte” da conclusão do projeto, devem reportar ao *Scrum Master*, buscando informações e relatando possíveis impedimentos.

Como procedimentos essenciais para o andamento da metodologia de uso das cerimônias, o *Scrum Team* também fica responsável pelos cadastros e atualizações do *GitLab*, a fim de sempre manter os projetos alimentados com informações pertinentes ao seu desenvolvimento e soluções de problemas, possibilitando as consultas atualizadas de informações no momento das cerimônias a serem realizadas.

Scrum Master

Como já mencionado, o *Scrum Master* é responsável por gerenciar o processo do *Scrum*, e a partir da utilização da metodologia de uso das cerimônias acumula a função de gestão do *GitLab* e do *GLSMeeting*, sendo opcional eleger membros do *Scrum Team* em certos momentos para realizar tarefas pertinentes aos *softwares*.

4.3.2. Ferramentas utilizadas

A utilização de ferramentas específicas é uma característica da metodologia de uso das cerimônias, sendo necessário que o *Scrum Master* e o *Scrum Team* tenham acesso constante ao *Gitlab* para cadastros e atualizações de atividades pertinentes ao produto em desenvolvimento, bem como o uso do *GLSMeeting* para gerenciamento das Atas de reuniões que devem ser elaboradas conforme explicado posteriormente.

4.3.3. Product backlog

Para iniciar um novo projeto é necessária a análise do *Product Backlog* apresentado pelo *Scrum Owner*. O *Product Backlog* deve ser detalhado em lista contendo as funcionalidades que o *Product Owner* entende ser necessário para o produto, assim como já explicado no item 2.2.2.3 deste trabalho.

Cadastrar Projeto no Gitlab

O processo “Cadastrar projeto no *Gitlab*” dentro da metodologia está localizado conforme Figura 21.

Figura 21 - Localização "cadastro de projeto" no processo da metodologia



O cadastro de projeto no *Gitlab* é uma tarefa a ser realizada pelo *Scrum Master* ou por alguém eleito por ele. Para isso é necessário o cadastro de conta ou login no *Gitlab* e assim prosseguir para o cadastro do Projeto.

Figura 22 - Página de cadastro de projeto no *Gitlab*

Blank project Create from template Import project CI/CD para um repositório externo

Nome do projeto
My awesome project

URL do projeto Slug do projeto
https://gitlab.com/ developer_sdp my-awesome-project

Want to house several dependent projects under the same namespace? [Create a group.](#)

Project description (optional)
Description format

Visibility Level ⓘ
 Privado
Acesso ao projeto deve ser concedido explicitamente para cada usuário.

Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project Cancelar

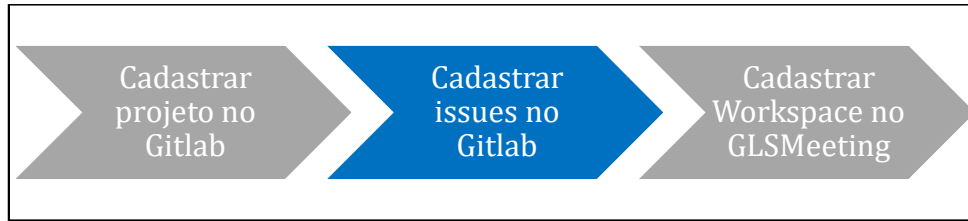
Fonte: Gitlab (2019)

A Figura 22 apresenta a página de cadastro do *Gitlab*, que disponibiliza os seguintes campos de preenchimento: nome, *URL* (personalizável), visibilidade e descrição. Os campos de preenchimentos são autoexplicativos sobre seus significados.

Cadastrar Issues no Gitlab

O processo “Cadastrar issues no *Gitlab*” dentro da metodologia está localizado conforme Figura 23.

Figura 23 - Localização "cadastro de issues" no processo da metodologia



As *issues* são tarefas a serem implementadas para que um item do *Product Backlog* seja desenvolvido. Portanto, cada item do *Product Backlog* pode corresponder a uma ou mais *issues* no *Gitlab*. Por exemplo:

Item do *Product Backlog*: “Usuário deve poder alterar foto e nome em seu perfil”

Issues:

- (1) Página de alteração do perfil do usuário
- (2) Upload de foto de perfil do usuário
- (3) Alteração de dados do usuário no BD

Os cadastros das *issues* são realizados pelo *Scrum Team*, pois como explicado no item 4.3.1, o *Scrum Team* deve manter os cadastros e atualizações do *Gitlab* e são responsáveis pelo sucesso da interação e do produto pronto.

Figura 24 - Página de cadastro de *Issue* no *Gitlab*

Departamento de Computação e Informática do CEULP ULBRA > Fábrica de Software > Portal-wp > Issues > Novo

Nova Issue

Título

Add description templates to help your contributors communicate effectively!

Description

Markdown and ações rápidas are supported

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee

Due date

Milestone

Labels

Weight

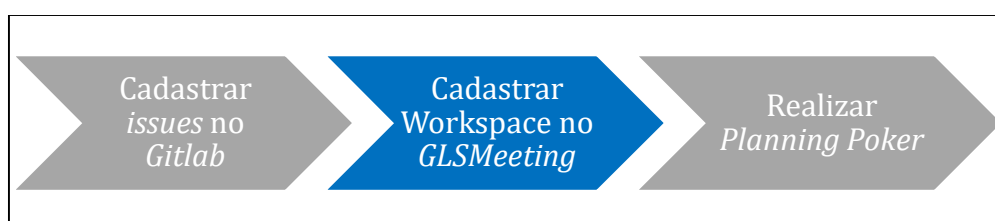
A Figura 24 apresenta a página de cadastro de uma *issue* no *Gitlab*, na qual é necessário informar as informações: título, descrição, *assignee* (responsável pela *issue*), previsão de término, *milestone* (Sprint a ser definida posteriormente), *label* (etiqueta, tag) e *weight* (peso de complexidade da *issue* para “*Plannig Poker*”).

É importante o cadastro e atualizações frequentes de informações pertinentes às *issues*, pois as mesmas podem ser referenciadas a qualquer momento nas reuniões diárias elaboradas com auxílio do *GLSMeeting*.

Cadastrar *Workspace* no *GLSMeeting*

O processo “Cadastrar *Workspace* no *GLSMeeting*” dentro da metodologia está localizado conforme Figura 25.

Figura 25 - Localização "cadastro de *Workspace*" no processo da metodologia



O *Workspace* é a página do *GLSMeeting* disponibilizada para o grupo dos projetos cadastrados e vinculados ao usuário no *Gitlab*. Caso o *Workspace* não esteja criado, deve-se criar e configurar para acesso aos projetos do *Gitlab* vinculados ao usuário.

Figura 26 - Página de cadastro de novo *Workspace* no *GLSMeeting*

INICIO santiagosdp (Sair)

Workspaces

Fábrica de Software

Home Office

Privado old

+Workspace

Desabilitados

Nova Workspaces

Nome

Digite um nome para o Workspace

Grupo

5445240 Departamento de Com

Adicionar

Desenvolvido por Santiagosdp

A Figura 26 apresenta a página referente a cadastro de nova *Workspace* no *GLSMeeting*. Para o cadastro é necessário informar o nome para o *Workspace* e selecionar o grupo que será vinculado. O grupo corresponde ao grupo de projetos ao qual o usuário tem acesso no *Gitlab* e são obtidos através da sincronização via API *Gitlab*, explicado anteriormente no tópico 4.2.1.

Após cadastro da *Workspace*, o usuário é redirecionado para a página de *Workspace*, na qual é possível verificar as atas criadas, conforme apresenta a Figura 27.

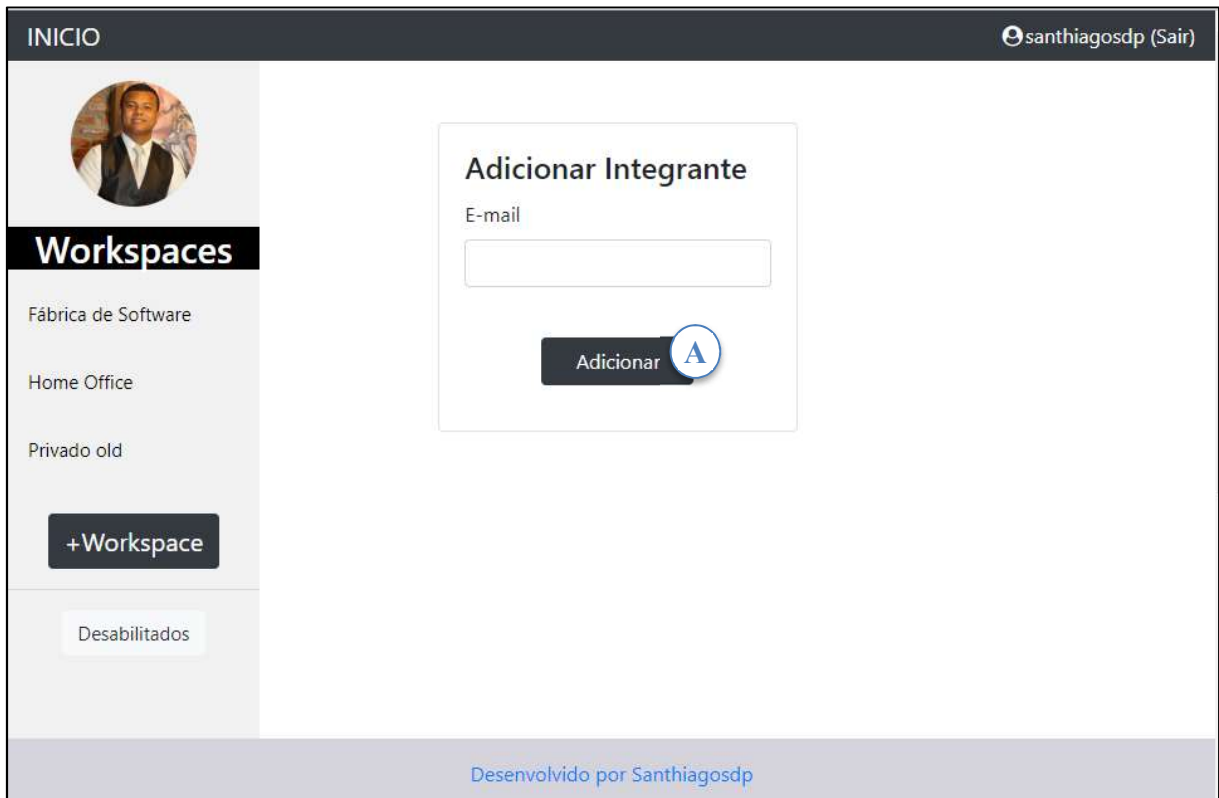
Figura 27 - Página do *workspace* no *GLSMeeting*

The screenshot shows the 'Fábrica de Software' workspace page. At the top, there is a header with 'INICIO' on the left and 'santiagosdp (Sair)' on the right. Below the header, a user profile picture is visible on the left. The main content area is titled 'Fábrica de Software' and includes a 'Desabilitar' link and a 'Nova Cerimônia' button. A table lists meeting records with columns for 'Inicio', 'Tipo', 'Projeto', and 'Ações'. The table contains four rows of meeting data. To the right of the table, there is a section for 'Integrantes' showing 'Santiago Dionizio Pinto' and a '+Integrante' button. At the bottom of the page, it says 'Desenvolvido por Santiagosdp'.

Inicio	Tipo	Projeto	Ações
dd/mm/aaaa	Imprimir Workspace		
12/11/2019 15:44	Reunião Final	14499983 Divulgação Vestibular 2020	
11/11/2019 17:40	Reunião Diária	14499983 - Divulgação Vestibular 2020	
11/11/2019 15:49	Reunião Diária	14499983 - Divulgação Vestibular 2020	
10/11/2019 15:41	Reunião Planejamento	14499983 Divulgação Vestibular 2020	

A página do *Workspace*, demonstrada na Figura 27, apresenta: a) as atas de reuniões realizadas (item que será explicado adiante); b) a lista dos integrantes; e c) o botão para convidar integrante ("+Integrante"), que pode realizar o gerenciamento das atas do *Workspace*. Ao clicar no botão "+Integrante" o usuário é direcionado para página de cadastro de novo integrante, conforme Figura 28.

Figura 28 - Página de cadastro de integrante no *GLSMeeting*



The screenshot displays the user interface for adding a new member to a workspace. At the top left, there is a navigation bar with 'INICIO' and a user profile 'santhiagosdp (Sair)'. The left sidebar contains a 'Workspaces' section with a user profile picture, a list of workspaces: 'Fábrica de Software', 'Home Office', and 'Privado old', a '+Workspace' button, and a 'Desabilitados' button. The main content area features a form titled 'Adicionar Integrante' with an 'E-mail' input field and an 'Adicionar' button with a blue circular icon containing the letter 'A'. At the bottom of the page, there is a footer that reads 'Desenvolvido por Santhiagosdp'.

Para concluir o cadastro de novo integrante ao *Workspace*, o usuário digita e-mail e clica no botão “*adicionar*” (Figura 28-A). O integrante adicionado tem a opção de aceitar ou não aceitar fazer parte do grupo.

4.3.4. Sprint Planning Meeting

Para iniciar a implementação do projeto solicitado pelo *Product Owner* é necessário o planejamento da *Sprint* a ser desenvolvida, e seguindo o processo da metodologia de uso das cerimônias na Figura 20, é necessária a utilização da técnica *Planning Poker* para a definição da *Sprint* a ser desenvolvida.

Realizar *Planning Poker*

O processo “Realizar *Planning Poker*” dentro da metodologia está localizado conforme Figura 29.

Figura 29 - Localização "realizar *Planning Poker*" no processo da metodologia

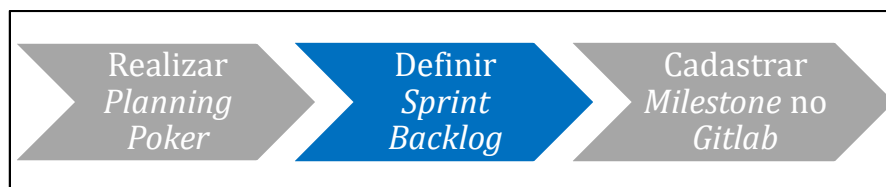


O *Planning Poker* é uma técnica de gamificação que se baseia no modelo de consenso entre os participantes para definir uma estimativa, no caso a *Sprint Planning Meeting*, a técnica é utilizada para definir a *Sprint Backlog* e o tempo de desenvolvimento previsto para o incremento. Nesta etapa o *Scrum Master*, em consenso com o *Scrum Team*, define quais os itens do *Product Backlog* serão desenvolvidos na *Sprint*. O tempo de reunião pode variar e depender da complexidade de cada projeto, portanto, fica a critério do *Scrum Master* a definição de início e término da reunião.

Definir a *Sprint Backlog*

O processo “Definir *Sprint Backlog*” dentro da metodologia está localizado conforme Figura 30.

Figura 30 - Localização "Definir *Sprint Backlog*" no processo da metodologia



Sendo resultado da *Sprint Planning Meeting*, o *Sprint Backlog* é a junção dos itens do *Product Backlog* selecionados para serem implementados na *Sprint*. Conforme já explicado no tópico 2.2.2.3, a implementação dos itens do *Sprint Backlog* é de responsabilidade do *Scrum Team*, ficando o *Scrum Master* com a obrigação de liderar e monitorar a execução das atividades.

Cadastrar Milestone no Gitlab

O processo “Cadastrar *Milestone* no *Gitlab*” dentro da metodologia está localizado conforme Figura 31.

Figura 31 - Localização "cadastro de *Milestone*" no processo da metodologia



A *Milestone no Gitlab* é uma maneira utilizada para filtrar as *issues* provenientes dos itens da *Sprint Backlog*. A *milestone* permite a junção de *issues* em um grupo coeso para apresentação, com a definição de data de início e data de término, facilitando assim o acompanhamento do andamento do desenvolvimento por parte do *Scrum Master*.

Com isso, após definir o *Sprint Backlog* durante a *Sprint Planning Meeting*, é necessário o cadastro da *milestone* no *GitLab*, além da atualização das *issues* pertinentes para adicioná-las a referida *milestone*.

Figura 32 - Página de cadastro de nova *Milestone* no *Gitlab*

developer_SDP > gamsprj > Marcos > Novo

Novo Marco

Título

Start Date [Clear start date](#)

Descrição

Due Date [Clear due date](#)

Como demonstrado na Figura 32, o cadastro de uma nova *milestone* deve ser feito acrescentando as informações de título, descrição, data de início e a data de término. É imprescindível a descrição detalhada, informando data da *Sprint Planning Meeting* e outros dados que julgar importante para o bom entendimento de toda equipe.

Cadastrar Ata de planejamento da *Sprint* no *GLSMeeting*

O processo “Cadastrar Ata de Planejamento” dentro da metodologia está localizado conforme Figura 33.

Figura 33 - Localização "cadastro de ATA" no processo da metodologia



A ata de Planejamento deve ser cadastrada no *GLSMeeting* durante a *Sprint Planning Meeting*. Para isso, o usuário designado deve selecionar o *Workspace* e criar uma reunião de Planejamento.

Figura 34 - Página do *workspace* no *GLSMeeting*

A captura de tela mostra a interface do usuário do *GLSMeeting*. No topo, há uma barra de navegação com 'INICIO' e o nome de usuário 'santiagosdp (Sair)'. O cabeçalho da página indica 'Desabilitar Fábrica de Software' e um botão 'Nova Cerimônia' (marcado com 'A').

À esquerda, há um menu 'Workspaces' com 'Fábrica de Software' selecionado e um botão '+Workspace'. Abaixo, há uma seção 'Desabilitados'.

O conteúdo principal é uma tabela com as seguintes colunas: 'Início', 'Tipo', 'Projeto' e 'Ações'. A tabela contém quatro registros:

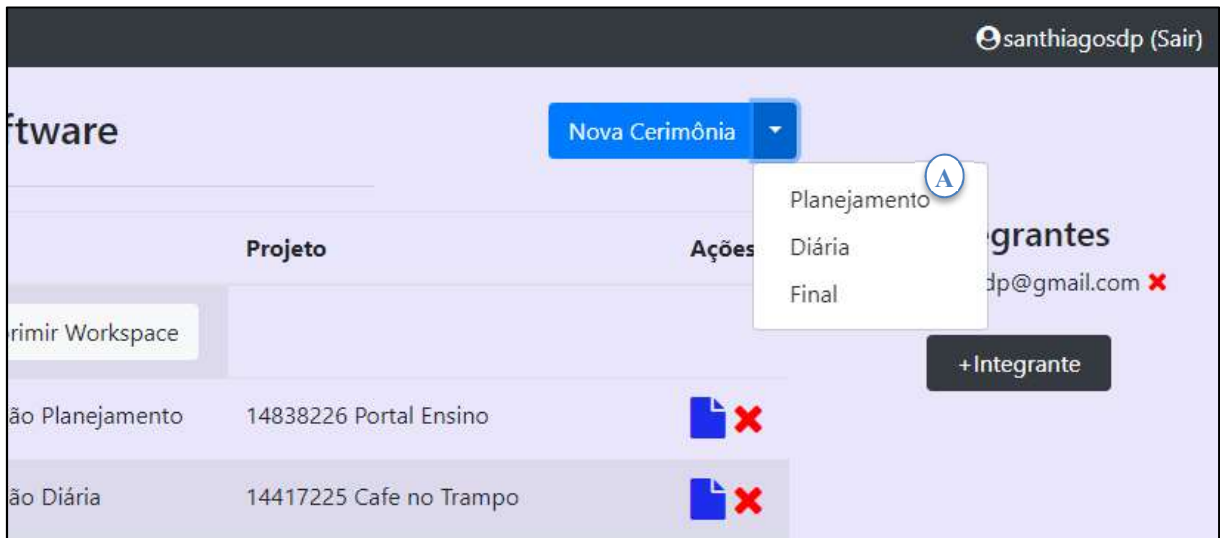
Início	Tipo	Projeto	Ações
dd/mm/aaaa	Imprimir Workspace		
12/11/2019 15:44	Reunião Final	14499983 Divulgação Vestibular 2020	
11/11/2019 17:40	Reunião Diária	14499983 - Divulgação Vestibular 2020	
11/11/2019 15:49	Reunião Diária	14499983 - Divulgação Vestibular 2020	
10/11/2019 15:41	Reunião Planejamento	14499983 Divulgação Vestibular 2020	

À direita da tabela, há uma seção 'Integrantes' com o nome 'Santiago Dionizio Pinto' e um botão '+Integrante'.

Na base da página, há o texto 'Desenvolvido por Santiagosdp'.

A Figura 34 apresenta a página do *Workspace* no *GLSMeeting* e para criar uma reunião de planejamento deve-se clicar no botão “nova cerimonia” (Figura 34-A) para aparecer as opções conforme a Figura 35.

Figura 35 - Ação após clicar no botão " nova cerimônia"



A Figura 35 apresenta o menu secundário após clicar no botão “nova cerimônia”, o qual é responsável pela escolha de qual tipo de reunião deseja iniciar. Portanto escolha-se a opção de reunião de planejamento para iniciar uma nova *Sprint Panning Meeting*.

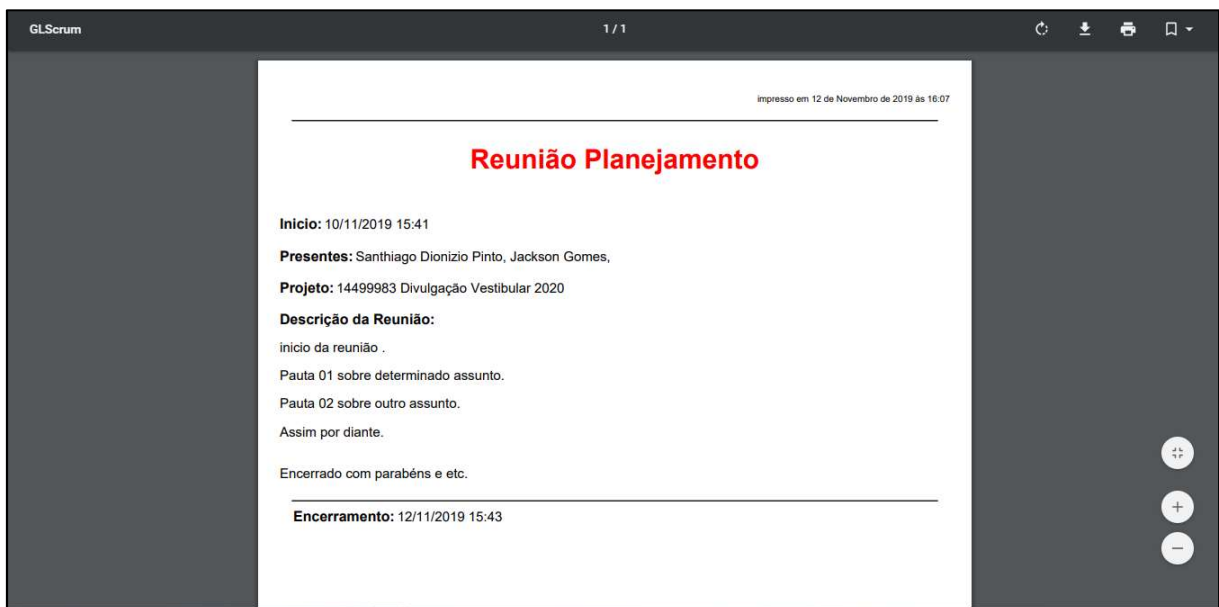
Figura 36 - Página de cadastro de reunião de Planejamento no *GLSMeeting*

A captura de tela mostra a página de cadastro de reunião de Planejamento. O título é 'Cadastrar Reunião Planejamento'. O formulário contém os seguintes campos: 'Início:' com o valor '19/10/2019 16:25'; 'Participantes:' com dois itens: 'x santiagosdp@gmail.com' e 'x Outra pessoa que não seja integrante'; 'Projeto:' com o valor '14417225 Cafe no Trampo'; 'Descrição:' com um campo de texto contendo o texto: 'Aqui deve ser descrito o que foi tratado na reunião e observações que o Scrum Master entender de utilidade para registro.'; e 'Término' com o valor '19/10/2019 --:--'. Um botão 'Salvar' está localizado no canto inferior direito.

A Figura 36 apresenta a página de cadastro de reunião de planejamento, na qual é possível adicionar e remover os participantes da reunião, sejam eles integrantes ou não do *workspace*, selecionar o projeto relacionado e também o campo para redigir texto corrido sobre os assuntos tratados na referida. Ao preencher os dados, o usuário deve clicar em “Salvar” para que a reunião seja cadastrada e listada na página do *Workspace*.

Para visualização/Impressão de qualquer ata, o usuário deve clicar no ícone de PDF (Figura 34-B). O *software* fará o redirecionamento para visualização da página de impressão da ata selecionada. Conforme Figura 37.

Figura 37 - Exemplo de visualização/impressão de ata de planejamento



A Figura 37 apresenta a visualização de impressão em PDF de um exemplo de ata de planejamento. As informações apresentadas são cadastradas no momento de criação da ata de planejamento e para impressão, é considerado um melhor *designer* de apresentação.

4.3.5. Sprint

A *Sprint* é o desenvolvimento do resultado planejado durante a *Sprint Planning Meeting*, pois trata da implementação com reuniões diárias (*Daily Scrum*), atualizações do *Gitlab* e a reunião final da *Sprint* (*Sprint Review & Retrospective*).

É importante ressaltar que durante o desenvolvimento da *Sprint*, o *Scrum Team* deve estar sempre com o *Gitlab* atualizado, inibindo possíveis erros de informação nas reuniões diárias e/ou análise de desempenho assistido pelo *Scrum Master*.

4.3.6. Daily Scrum

O processo “Daily Scrum” dentro da metodologia está localizado conforme Figura 38.

Figura 38 - Localização "Daily Scrum" no processo da metodologia



Durante a implementação, é necessária a realização de reuniões diariamente para o alinhamento de informações pertinentes ao desenvolvimento.

A *Daily Scrum* é realizada para análise do progresso do desenvolvimento, tendo o objetivo de troca de informações para o pleno entendimento do andamento do projeto por todos.

Durante a *Daily Scrum*, opcionalmente, o *Scrum Master* pode solicitar que o *Scrum Team* responda as três perguntas sugeridas do *Scrum* (1- O que foi feito desde ontem? 2- O que você planeja fazer para amanhã? 3- Teve algum impedimento durante ou nas atividades realizadas?).

O *Scrum Master* fica responsável por gerenciar o tempo das reuniões, sendo o registro de tais dados obtidos de forma automática no *GLSMeeting*, caso não informados. Durante essa etapa, é elaborada a ata diária da *sprint*.

Cadastro da Ata diária da Sprint

O processo “Cadastro da Ata Diária” dentro da metodologia está localizado conforme Figura 39.

Figura 39 - Localização "cadastro de ata diária" no processo da metodologia



Diariamente, durante o *Daily Scrum*, o *Scrum Master* deve elaborar a ata de reunião diária. O cadastro da ata deve ser realizado na ferramenta *GLSMeeting* pelo usuário designado, acessando o *Workspace*, clicar no botão “*criar cerimônia*” e na opção de reunião diária, de acordo com a Figura 35 apresentada anteriormente.

Figura 40 - Página de cadastro de reunião diária no *GLSMeeting*

Cadastrar Reunião Diária

Início:
19/10/2019 16:39

Participantes:
× santhiagosdp@gmail.com
× outra pessoa que nao seja integrante
× Diretor@exemplo

Projeto:
13069364 SIG

Assuntos (issues):
× 26098770 - SIG - ACADEMICO - Script para Importação das Ementas :opened
× 25866697 - SIG - CORE - API - Criar serializer de pessoa :opened

Término
19/10/2019 --:--

Salvar

A Figura 40 apresenta a página de cadastro de uma reunião diária, em que se deve informar o horário de início, participantes, selecionar o projeto, adicionar as *issues* que foram abordadas e informar o horário de término da reunião.

A ata de reunião diária apresentará como assunto somente o número e nome das *issues* abordadas e adicionadas no *input* correspondente, se for necessário a descrição de alguma informação, um dos participantes deve acessar a *Gitlab* e atualizar a *issue* com a informação pertinente.

4.3.7. Sprint Review & Retrospective

De acordo com a Figura 20, ao final do desenvolvimento da *Sprint*, é necessária a realização da *Sprint Review & Retrospective*, que visa fazer a validação da conclusão da *Sprint*, demonstrando o produto com os itens da *Sprint Backlog* desenvolvido, e estabelecer pontos de melhorias para o desenvolvimento das *Sprints* seguintes.

Considerada como reunião de validação e lições aprendidas, a *Sprint Review & Retrospective* deve ser realizado com a presença do *Scrum Team*, *Scrum Master* e opcionalmente o convite pode se estender ao *Scrum Owner*. A duração da reunião é de

responsabilidade do *Scrum Master*, que deve analisar as abordagens para definir tempo de início e término.

É importante ressaltar que a *Sprint Review & Retrospective* é realizada quando todas as *issues* da *milestone* (ou itens do *Sprint Backlog*) em desenvolvimento estejam implementados e testados, ficando o *Scrum Team* obrigados a se reportar ao *Scrum Master* sobre a conclusão. Durante a *Sprint Review & Retrospective* é elaborada a ata final da *Sprint*.

Cadastro da Ata Final da Sprint

O processo “Cadastro da Ata Final” dentro da metodologia está localizado conforme Figura 41.

Figura 41 - Localização "cadastro de ata final" no processo da metodologia



Durante a *Sprint Review & Retrospective*, o *Scrum Master* é responsável pelo cadastro da ata Final da *Sprint* no *GLSMeeting*, no qual o usuário designado deve acessar o *Workspace* clicar no botão “*criar cerimônia*” e na opção de reunião final, de acordo com a Figura 35 apresentada anteriormente

Figura 42 - Página de cadastro de reunião final no *GLSMeeting*

Cadastrar Reunião Final

Início:
19/10/2019 16:43

Participantes:
× santhiago dionizio pinto ×
× outra pessoa que esteja participando × etc

Projeto:
13069364 SIG

Descrição:
Assuntos tratados na reunião final devem ser digitados aqui.

Término
19/10/2019 --:--

Salvar

A Figura 42 apresenta a página de cadastro de uma reunião final, através da qual é necessário informar o horário de início da reunião e os participantes, selecionar o projeto relacionado, descrever os assuntos tratados na reunião e informar o horário de término da reunião.

Para a ata final, a descrição deve abordar problemas e dificuldades encontradas no desenvolvimento da *Sprint*, sugestões e possíveis impedimentos para o restante do desenvolvimento.

4.3.8. Nova *Sprint*

Após conclusão da implementação da *Sprint* em andamento, é necessária uma nova *Sprint Planning Meeting* para definir novo planejamento para elaborar o *Sprint Backlog* através dos itens em aberto no *Product Backlog* e realizar os processos explicados para o pleno desenvolvimento da *Sprint*. O Processo de *loop* é feito até que não haja mais itens em aberto no *Product Backlog* e assim, o projeto esteja totalmente implementado e testado, conforme explicado em Produto Pronto.

4.3.9. Produto Pronto

O Produto Pronto é caracterizado pela conclusão da implementação de todos os itens do *Product Backlog*, além da realização de reunião final da última *Sprint* implementada, sendo necessária a presença do *Product Owner* para fazer validações e testes do produto, confirmando a conclusão do projeto.

5. CONSIDERAÇÕES FINAIS

Este trabalho consistiu no desenvolvimento de uma proposta de metodologia para auxiliar os procedimentos de realização das cerimônias do *Scrum* na Fábrica de *Software* do CEULP/ULBRA, com o objetivo de melhorar o desempenho da equipe e automatização do processo. Desta forma a metodologia *GLScrum* apresenta as etapas a serem executadas para realização das cerimônias e faz uso de um *software* para gerenciamento das atas de reuniões cadastradas (*GLSMeeting*).

A utilização da metodologia *GLScrum* contribui para o melhor aproveitamento do tempo e gerenciamento das reuniões, atualizações das informações no gerenciador de repositório *Gitlab* e o aprimoramento constante do *Scrum*, considerado base para o desenvolvimento e aplicação da metodologia *GLScrum*.

Com a aplicação de técnicas como o gerenciamento de comunicação e gerenciamento de tempo, é possível que a equipe tenha um melhor empenho para realização das tarefas e cumprimento com os objetivos de produção da organização, bem como o autogerenciamento do seu tempo de desenvolvimento.

O *software GLSMeeting* desenvolvido para utilização na metodologia *GLScrum*, utiliza a *API GITLAB* fornecida pela empresa *Gitlab* para acesso ao repositório e informações de perfil de seus usuários. Com o uso da *API Gitlab* foram possíveis a integração de autenticação dos usuários e a busca em tempo real das informações dos projetos a serem abordados nas reuniões, garantindo transparência e confiabilidade no processo.

Dada a importância de ter transparência, confiabilidade e autenticidade dos processos de gerenciamento das atas de reuniões, recomenda-se para trabalhos futuros a implementação de mais funcionalidades no *GLSMeeting*, principalmente a opção de edição das atas cadastradas e um modelo de validação para confirmar a veracidade das informações. Além disso, recomenda-se também a implantação da metodologia *GLScrum* na Fábrica de *Software* do CEULP/ULBRA, monitoramento, análise das informações e aplicação de melhorias que sejam consideradas relevantes ao contexto.

REFERÊNCIAS

ALCÂNTARA, Túlio de Souza. **Um Paralelo Entre O Manifesto Ágil e o Sistema Toyota De Produção**. 2009. 78 f. Monografia (Especialização) - Curso de Engenharia da Computação, Universidade de Pernambuco, Recife, 2009. Disponível em: <https://tcc.ecomp.poli.br/20092/TCC_final_Tulio.pdf>. Acesso em: 12 mar. 2019.

AMBLER, Scott. **Agile Modeling: Effective Practices for Extreme Programming and the Unified Process**. New York: John Wiley & Sons, Inc, 2002. 402 p. Disponível em: <<http://msoo.pbworks.com/f/Scott+W.+Ambler+-+Agile+Modeling.pdf>>. Acesso em: 18 mar. 2019.

AUGUSTO, Marcus Vinícius. **Desenvolvimento de Software com Apoio de Práticas Scrum**. 2011. 40 f. TCC (Graduação) - Curso de Tecnologia em Processamento de Dados, Faculdade de Tecnologia de São Paulo, São Paulo, 2011. Disponível em: <<http://www.fatecsp.br/dti/tcc/tcc0011.pdf>>. Acesso em: 31 mar. 2019.

BOMFIN, David Ferreira; NUNES, Paula Cristine de Ávila; HASTENREITER, Flávio. Gerenciamento de Projetos Segundo o Guia PMBOK: Desafios para os Gestores. **Revista de Gestão e Projetos**, São Paulo, v. 3, n. 3, p.58-87, 1 nov. 2012. *University Nove de Julho*. <http://dx.doi.org/10.5585/gep.v3i3.78>. Disponível em: <<http://www.revistagep.org/ojs/index.php/gep/article/view/78/307>>. Acesso em: 13 abr. 2019.

BURNDOWN chart. Disponível em: <https://subscription.packtpub.com/book/application_development/9781849699730/6/ch06lv11sec55/burndown-chart>. Acesso em: 30 maio 2019.

CARVALHO, Bernardo Vasconcelos de; MELLO, Carlos Henrique Pereira. **Aplicação do método ágil Scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica**. 2010. 17 f. Tese (Doutorado) - Curso de Engenharia da Computação, Universidade Federal de Itajubá – UNIFEI, Itajubá, Mg, 2012.

CASTRO, Ronney Moreira de et al. **Agility Scrum: Um Jogo para Ensino da Metodologia Scrum**. 25º Wei - *Workshop* Sobre Educação em Computação, São Paulo, p.2217-2226, 2017.

ERLO, Eduardo. **Desenvolvimento de uma ferramenta WEB para extração customizada de informações de bancos de dados relacionais**. 2012. 188 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Computação e Tecnologia da Informação, Universidade de Caxias do Sul, Caxias do Sul, 2012. Disponível em: <<https://repositorio.ucs.br/xmlui/bitstream/handle/11338/1492/TCC%20Eduardo%20Erlo.pdf?sequence=1&isAllowed=y>>. Acesso em: 09 maio 2019.

FUNARI, Ricardo. **Uma Análise da Aplicação do Método Scrum na Gestão do Portfólio de Projetos em uma Empresa de Tecnologia da Informação**. 2015. 88 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Centro Universitário de Araraquara – Uniara, Araraquara, Sp, 2015. Disponível em: <<https://www.uniara.com.br/arquivos/file/ppg/engenharia-producao/producao-intelectual/dissertacoes/2015/ricardo-funari.pdf>>. Acesso em: 30 maio 2019.

FURTADO, Brenddon Gontijo. **Sistema Web para Monitoramento Energético**. 2016. 48 f. TCC (Graduação) - Curso de Engenharia de Software, Universidade de Brasília - UNB, Brasília, DF, 2016. Disponível em: <<https://fga.unb.br/articles/0001/6729/v1-tcc-brenddon-gontijo.pdf>>. Acesso em: 13 set. 2019.

GITLAB. **GitLab - About**. Disponível em: <<https://about.gitlab.com/>>. Acesso em: 18 maio 2019.

JOSE JUNIOR. **Framework Scrum em 5 minutos**. Disponível em: <<http://www.josejr.com.br/framework-scrum-em-5-minutos/>>. Acesso em: 30 maio 2019.

LEIDEMER, Rômulo Henrique. **Implantação De Scrum Em Uma Empresa De Desenvolvimento De Software**. 2013. 129 f. Monografia (Especialização) - Curso de Sistemas de Informação, Gerência de Projetos, Centro Universitário Univates, Lajeado, 2013. Disponível em: <<https://www.univates.br/bdu/bitstream/10737/361/1/RomuloLeidemer.pdf>>. Acesso em: 05 mar. 2019.

MANGELLI, Leonardo S. L. Passeri. **Gestão de Projetos e o Guia PMBOK: Um Estudo Sobre o Nível de Uso do Guia PMBOK nas Empresas Brasileiras**. 2013. 94 f. Dissertação (Mestrado) - Curso de Administração, Centro de Formação Acadêmica e Pesquisa, Escola Brasileira de Administração Pública e de Empresas, Rio de Janeiro, 2013. Disponível em: <<http://bibliotecadigital.fgv.br/dspace/handle/10438/11354>>. Acesso em: 13 abr. 2019.

MARÇAL, Ana Sofia Cysneiros et al. **Estendendo o Scrum segundo as Áreas de Processos de Gerenciamento de Projetos do CMMI**, 2007. Disponível em: <<http://luizhoffmann.com.br/docs/Projeto/artigo11.pdf>>. Acesso em: 31 mar. 2019.

OLIVEIRA, Alexandre Ferreira de. **Gestão de projetos estratégicos: um estudo de caso**. 2007. 197 f. Tese (Doutorado) - Curso de Mestrado em Engenharia Naval e Oceânica, Mestrado em Engenharia Naval e Oceânica, Universidade de São Paulo, São Paulo, 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/3/3135/tde-25062007-171931/pt-br.php>>. Acesso em: 13 abr. 2019.

PMI. *A guide to the project management body of knowledge (PMBOK Guide), Project Management Institute, 4th ed., Newton Square, PA, 2008*. Disponível em: https://www.academia.edu/37884999/Guide_to_Project_Management_Book_of_Knowledge_4th_edition

PRESSMAN, Roger S. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. Porto Alegre: Amgh, 2011. Tradução de: Ariovaldo Griesi.

REIS, Caio Almeida Arêas. **A Importância do Escritório de Projetos no Gerenciamento de Projetos: Um Estudo de Caso na MRS Logística S.A.** 2011. 61 f. TCC (Graduação) - Curso de Engenheiro de Produção, Universidade Federal de Juiz de Fora, Juiz de Fora, 2011. Disponível em: <http://www.ufjf.br/engenhariadeproducao/files/2014/09/2011_3_Caio.pdf>. Acesso em: 30 maio 2019.

SANTOS, Melquizedequi Cabral dos. **O Impacto do Uso das Metodologias Ágeis Scrum e XP na Satisfação dos Stakeholders**. 2014. 82 f. Dissertação (Mestrado) - Curso de Pós-graduação em Ciência da Computação, Centro de Informática da Universidade Federal de Pernambuco, UFPE - Universidade Federal de Pernambuco, Recife, 2014. Disponível em: <<https://repositorio.ufpe.br/handle/123456789/11646>>. Acesso em: 18 mar. 2019.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum: Um guia definitivo para o Scrum: As regras do jogo**. 2014. Disponível em: <<https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>>. Acesso em: 31 mar. 2019.

SILVA, Alessandra Galvão da. **A Importância dos Métodos Ágeis na Engenharia de Software**. 2016. 62 f. TCC (Graduação) - Curso de Tecnologia em Sistemas de Computação, Universidade Federal Fluminense, Niterói, 2016. Disponível em: <https://app.uff.br/riuff/bitstream/1/5488/1/TCC_ALESSANDRA_GALVAO_DA_SILVA%20%281%29.pdf>. Acesso em: 30 maio 2019.

SILVA, Daisy Eliana dos Santos; SOUZA, Ingredy Thaís de; CAMARGO, Talita. Metodologias Ágeis Para O Desenvolvimento De Software: Aplicação E O Uso Da Metodologia Scrum Em Contraste Ao Modelo Tradicional De Gerenciamento De Projetos. **Revista Computação Aplicada - UNG-SER**, Guarulhos, v. 2, p.39-46, 2013. Disponível em: <<http://revistas.ung.br/index.php/computacaoaplicada/article/view/1408/1194>>. Acesso em: 31 mar. 2019.

SOARES, Michel dos Santos. Metodologias Ágeis *Extreme Programming* e *Scrum* para o Desenvolvimento de Software. **Revista Eletrônica de Sistemas de Informação**, Conselho Lafaiete, v. 3, n. 1, 30 jun. 2004. IBEPES (Instituto Brasileiro de Estudos e Pesquisas Sociais). <http://dx.doi.org/10.21529/resi.2004.0301006>.

SOUZA, Diogo Rodrigues de. **Implantação da Metodologia ágil Scrum em um Ambiente de Desenvolvimento**. 2014. 63 f. TCC (Graduação) - Curso de Bacharel em Tecnologias da Informação e Comunicação, Universidade Federal de Santa Catarina, Araranguá, 2014. Disponível em: <<https://repositorio.ufsc.br/bitstream/handle/123456789/130043/TCC%20Final.pdf?sequence=1>>. Acesso em: 21 fev. 2019.

STOICA, Marian; MIRCEA, Marinela; GHILIC-MICU, Bogdan. *Software Development: Agile vs. Traditional*. **Informática Econômica**, [s.l.], v. 17, n. 4/2013, p.64-76, 30 dez. 2013. ECO-INFOSOC *Research Center*. <http://dx.doi.org/10.12948/issn14531305/17.4.2013.06>. Disponível em: <<http://www.revistaie.ase.ro/content/68/06%20-%20Stoica,%20Mircea,%20Ghilib.pdf>>. Acesso em: 06 mar. 2019.

SUTHERLAND, Jeff; SCHWABER, Ken. **The Scrum Papers. Nuts, Bolts and Origins of an Agile Process**, 2011.

TAVARES, Leonardo. **Definindo a prioridade das histórias utilizando o quadro Esforço x Valor**. Disponível em: <<https://www.synergia.dcc.ufmg.br/definindo-prioridade-das-historias-utilizando-o-quadro-esforco-x-valor/>>. Acesso em: 30 maio 2019.

The Postgresql Global Development Group. PostgreSQL: The World's Most Advanced Open Source Relational Database. Disponível em: <<https://www.postgresql.org/>>. Acesso em: 18 maio 2019.

UTIDA, Kleber Hiroki. **Metodologias Tradicionais e Metodologias Ágeis:** Análise Comparativa Entre *Rational Unified Process* e *Extreme Programming*. 2012. 48 f. Monografia (Especialização) - Curso de Tecnólogo em Processamento de Dados, Faculdade de Tecnologia do Estado de São Paulo, São Paulo, 2012. Disponível em: <<http://www.fatecsp.br/dti/tcc/tcc00055>>. Acesso em: 04 mar. 2019.