Trabalho de Conclusão de Curso

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

PGTBL: Plataforma de gerenciamento de Team-Based Learning

Autor: Victor Hugo Arnaud Deon
Orientador: Cristiane Soares Ramos
Coorientador: Ricardo Ajax Dias Kosloski

Brasília, DF
2019

Victor Hugo Arnaud Deon

PGTBL: Plataforma de gerenciamento de Team-Based Learning

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Cristiane Soares Ramos

Coorientador: Ricardo Ajax Dias Kosloski

Brasília, DF

2019

Victor Hugo Arnaud Deon

PGTBL: Plataforma de gerenciamento de Team-Based Learning/ Victor Hugo Arnaud Deon. – Brasília, DF, 2019

Orientador: Cristiane Soares Ramos

Coorientador: Ricardo Ajax Dias Kosloski

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2019.

1. TBL. 2. Ferramenta. 3. Automatização. 4. Software Livre. 5. Ensino. 6. Metodologia Ativa de aprendizado. I. Cristiane Soares Ramos. II. Ricardo Ajax Dias Kosloski. III. Universidade de Brasília. IV. Faculdade UnB Gama. V. PGTBL: Plataforma de gerenciamento de Team-Based Learning

CDU 02:141:005.6

Victor Hugo Arnaud Deon

PGTBL: Plataforma de gerenciamento de Team-Based Learning

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Cristiane Soares Ramos

Ricardo Ajax Dias Kosloski

Prof. DSC. André Luiz Peron Lanna

Brasília, DF
2019

Este trabalho é dedicado primeiramente a Deus,
por ser essencial em minha vida, ao meu pai Adalberto Jose Deon,
minha mãe Dyonne Arnaud Sampaio de Alencar Deon, meus irmãos
Jean Guilherme Arnaud Deon e Kaio Cesar Arnaud Deon, minha tia
Maria Soledade Arnaud meus amigos e agradeço também
os professores que me acompanharam durante a graduação, em especial
ao Prof. Ricardo Ajax Dias Kosloski, e as Profa. Cristiane Soares Ramos,
e Elaine Venson, responsáveis pela realização deste trabalho.
"Os limites só existem se você os deixar existir". Son Goku

Resumo

O mercado de engenharia exige algumas competências que devem ser enfatizadas, entre elas destacam-se: capacidade de trabalho em equipe, análise de dados, resolução de problemas reais. As metodologias ativas de aprendizagem tem como objetivo preencher essas lacunas que o mercado exige dos alunos, entre elas temos o Team-Based Learning (TBL) que é uma metodologia de aprendizagem colaborativa. Embora existam ferramentas para apoiar o uso do TBL como CMCs (Computer-Mediated Communication), não foi encontrado ferramentas para automatizar a execução e implantação do TBL no ambiênte acadêmico. O objetivo deste trabalho é a implementação de uma ferramenta chamada PGTBL para automatizar o processo de uso da metodologia ativa de aprendizado Team Based Learning. A metodologia utilizada para alcançar os objetivos foi uma adaptação ao SCRUM, XP e SAFe para um único desenvolvedor e pesquisa bibliográfica para o embasamento teórico do trabalho. A finalidade do software é que a aplicação do TBL se torne algo mais fácil, constante e automatizado, tornando o processo mais prazeroso, tanto para o aluno quanto para o professor. A ideia do sistema é ter um design atrativo e será responsável por todo o processo da metodologia de ensino TBL.

Palavras-chaves: TBL. metodologia ativa de aprendizado. aprendizado baseado em equipes. ferramenta. software livre. metodologia ágil. ensino. educação. automatização.

Abstract

The engineering profession requires some skills that should be emphasized, for example: teamwork, data analysis, real problem solving. The active learning methodologies have as objective to fill these gaps that the profession demands of the students, among them we have the Team-Based Learning (TBL) that is a collaborative learning methodology. Although there are tools to support the use of TBL as Computer-Mediated Communication (CMCs), no tools found to automate the implantation of TBL in the academic environment. The objective of this work is the implementation of a tool called PGTBL to automate the process of using the Active Learning Methodology Team Based Learning. The methodology used to achieve the objectives was an adaptation to SCRUM, XP and SAFe for a single developer and bibliographical research for the theoretical basis of the work. The purpose of the software is to make the TBL application easier, more constant and automated, making the process more enjoyable for both the student and the teacher. The idea of the system is to have an attractive design and will be responsible for the entire process of the TBL teaching methodology.

Key-words: TBL. active learning methodology. team based learning. tool. open-source. software. agile methodology. teaching. education. automated.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PGTBL	Plataforma Gerenciadora de Team Based Learning
TBL	Team Based Learning
PBL	Problem Based Learning
SAFe	Scaled Agile Framework
GQM	Goal Question Metric
EAP	Estrutura Analítica de Projetos
RAT	Readiness Assurance Test
iRAT	Individual Readiness Assurance Test
gRAT	Group Readiness Assurance Test
XP	Extreme Programming
UML	Unified Modeling Language
Devops	Desenvolvimento e Operações

Sumário

1 Introdução

Neste capítulo, é introduzido o trabalho de conclusão de curso. O capítulo está dividido em seções que estão dispostas da seguinte maneira: na seção 1.1 é contextualizado de forma resumida o que são as metodologias ativas de aprendizado até chegar no foco que é a metodologia *Team Based Learning*; na seção 1.2 é apresentado a problemática; na seção 1.3 é apresentado a justificativa; na seção 1.4 os objetivos; na seção 1.5 é apresentado a metodologia de pesquisa e desenvolvimento e por fim, na seção 1.6 é apresentado a organização do trabalho.

1.1 Contextualização

A educação está em um impasse diante das várias mudanças acontecendo na sociedade atual. As instituições que ensinam e avaliam a todos de forma igual e exigem resultados previsíveis, ignoram o fato que hoje vivemos em uma sociedade do conhecimento. Essa sociedade é baseada em competências cognitivas, pessoais e sociais, que não se adquirem da forma convencional e que exigem proatividade, visão, personalização, colaboração e empreendedorismo (MORAN, 2015).

Com a chegada da internet podemos aprender em qualquer lugar, a qualquer hora e com muitas pessoas diferentes. Essa mescla, entre sala de aula e ambientes virtuais é fundamental para abrir a escola para o mundo e trazer o mundo para dentro da escola. Na época em que o acesso à informação era difícil, fazia sentido os métodos tradicionais, que privilegiam a transmissão de informação pelos professores, porém na sociedade em que vivemos hoje esse tipo de metodologia não é tão efetiva, abrindo espaço para outras metodologias de ensino como as metodologias ativas de aprendizagem (MORAN, 2015).

Apesar de pouco conhecidas, as metodologias ativas de aprendizagem já vêm sendo aplicadas de forma indireta pelos professores por meio de projetos, resolução de problemas e outros meios de ensinar e aprender que podem ser considerados como um tipo de metodologia ativa (MORAN, 2015).

Não é de hoje que o mercado de engenharia exige dos seus profissionais algumas competências importante, como: capacidade de trabalho em equipe, análise de dados, resolução de problemas reais (DAVIS et al., 2013). As metodologias ativas de aprendizagem têm como objetivo preencher essas lacunas que o mercado exige dos alunos para que esses se sintam melhor preparados para o mercado.

Silberman modificou com suas palavras um provérbio chinês dito pelo filósofo Confúcio para facilitar o entendimento de métodos ativos de aprendizagem, que diz:

> "O que eu ouço, eu esqueço; o que eu ouço e vejo, eu me lembro; o que eu ouço, vejo e pergunto ou discuto, eu começo a compreender; o que eu ouço, vejo, debato e faço, eu aprendo desenvolvendo conhecimento e habilidade; o que eu ensino para alguém, eu domino com maestria."

Essa citação, com a modificação de Silberman, resume a teoria por trás das metodologias ativas de aprendizagem. Se o ensino englobar as atividades de ouvir, ver, perguntar, discutir, fazer e ensinar, estamos no caminho da aprendizagem ativa (BARBOSA; MOURA, 2013).

Existem hoje diversas metodologias ativas de aprendizagem, por exemplo, o *Team-Based Learning* ou TBL é uma metodologia de aprendizagem colaborativa, já o *Problem-Based Learning*, conhecida como PBL, é mais focada na resolução de problemas complexos com um toque ainda na abordagem pedagógica de ensino tradicional, como slides, provas e etc (CABRERA; VILLALON; CHAVEZ, 2017). O *Project-Based Learning*, tem como base a resolução de desafios por meio de projetos. Além disso pode-se contar com o apoio de um método ativo bastante atual chamado Sala de Aula Invertida que nada mais é do que substituir a maioria das aulas por conteúdos virtuais para que o tempo em sala seja otimizado (MORAN, 2015).

O principal foco deste trabalho é na metodologia ativa de aprendizado TBL, o qual implementa o construtivismo e dá ênfase no papel do aluno como mestre de suas próprias experiências educacionais. Eles saem de agente passivo do conhecimento para um agente ativo (GOMEZ; WU; PASSERINI, 2010).

1.2 Problemática

A maioria dos casos de uso do TBL utiliza-se de ferramentas de CMC (*Computer Mediated Communication*) como fóruns de discussão, chats entre outros meios de comunicação eletrônicos como Facebook (ALHOMOD; SHAFI, 2013), Blackboard, Moodle, Wiki ou Google Drive (AWATRAMINI; ROVER, 2015), entre outras tecnologias web (KAM; KATERATTANAKUL, 2014). Às vezes, utilizam até mais de uma dessas ferramentas juntas.

Mais especificamente, as ferramentas de comunicação fornecem um meio que permite que grupos de alunos troquem idéias e opiniões e compartilhem informações a qualquer hora e em qualquer lugar. Os principais benefícios do uso de CMC são a conveniência, a independência do lugar, do tempo e o potencial para que os usuários se tornem parte de uma comunidade virtual (BERGE; COLLINS, 1993). Porém, essas ferramentas são uma extensão do uso do TBL e não uma forma de automatizar o uso do mesmo.

Um excelente exemplo de uso de ferramenta de CMC é o *Moodle*, ela é uma boa

ferramenta para disponibilizar materiais e notas, além de ter um pequeno fórum para discussão. Porém, muitas vezes esses tipos de ferramentas deixam a desejar em vários aspectos, por exemplo, ela não calcula as notas dos alunos e dos grupos automaticamente por meio de apostas. Este cálculo é bem trabalhoso de se fazer já que o TBL é feito por pontuação e não por menção como é feito nas universidades e escolas, ele não gera relatórios para o professor saber qual é o tema que está gerando mais dúvidas entre os alunos, entre outros aspectos que são bem especificos da metodologia TBL.

Com isso temos a seguinte questão: Como automatizar a aplicação em sala de aula da Metodologia ativa de ensino conhecida por TBL - Team Based Learning para turmas de disciplinas de graduação do curso de Engenharia de Software da UnB-FGA - Faculdade Gama.

1.3 Justificativa

Procurando colaborar com essa problemática, o presente trabalho propôs a construção de uma plataforma para automatizar todo o processo da metodologia ativa de aprendizado: Team Based Learning, e com isso validar se a ferramenta estaria sendo efetiva no seu propósito.

1.4 Objetivo

O objetivo deste trabalho é a implementação da plataforma de gerenciamento da metodologia ativa de aprendizado: *Team Based-Learning*, chamada PGTBL, para automatizar o uso dessa metodologia no ambiente acadêmico da FGA, qual seja, o ensino de graduação de engenharia de software. Temos como objetivos específicos:

- Estudar o contexto do Team Based Learning;
- Especificar os requisitos do software para a sua utilização na FGA;
- Estabelecer o processo de desenvolvimento que será usado para elaborar a ferramenta pretendida;
- Desenvolver a ferramenta e aplicar em uma disciplina de graduação do curso de engenharia de software.
- Coletar feedback dos alunos comparando a utilização da metodologia antes e depois da ferramenta.

1.5 Metodologias

Como base de sustentação do trabalho, foi realizada uma pesquisa bibliográfica com o intuito de reunir as informações e dados que servirão de base para a construção da ferramenta, utilizando como banco de dados a Scopus e o Google Acadêmico, além de documentações ou sites oficiais de ferramentas ou metodologias.

Dentre as várias existentes, foram escolhidas metodologias pertencentes à corrente ágil de desenvolvimento, por serem uma tendência no ambiente da FGA. Foi criado um processo com adaptações para a utilização do SCRUM, XP e SAFe. Além disso, a ferramenta terá acesso livre para contribuições externas durante seu desenvolvimento, aplicando as boas práticas definidas na comunidade *Open Source*.

As metodologias, tanto do TCC, quanto do processo de desenvolvimento da ferramenta são mais detalhadas no capítulo 3.

1.6 Organização do Trabalho

O trabalho está organizado da seguinte forma: No Capítulo 2 é apresentado um pequeno referêncial teórico sobre o Team Based Learning. No Capítulo 3 é apresentado a metodologia de desenvolvimento, ou seja, será apresentado passo a passo o processo na qual o desenvolvimento foi realizado e a metodologia do trabalho de conclusão de curso (TCC). No Capítulo 4 é apresentado os resultados obtidos, entre eles os requisitos elicitados, arquitetura do software, métricas coletadas e algumas fotos do software. No Capítulo 5 foi documentado o caso de uso do software em disciplinas do curso de engenharia de software da UnB campus de engenharias Gama. E por último, temos as Considerações Finais na qual terá um resumo geral do que foi falado, e o que se pode esperar do projeto no futuro.

2 Team Based Learning

Neste capítulo, será descrito os conceitos-chave relacionado à metodologia ativa de aprendizagem TBL e está dividido em: O que é o TBL?, Detalhando cada etapa do TBL, Preparando um módulo em TBL e os pontos positivos e negativos encontrados na aplicação do TBL em algumas universidades.

2.1 O que é o TBL?

De acordo com (BURGESS; MCGREGOR; MELLIS, 2014) o TBL é uma estratégia educacional para grandes classes que, a partir da coordenação do professor, possibilita a interação e colaboração em pequenos grupos centrados no aluno para melhorar o processo de aprendizagem e capacitar os alunos para o mercado.

O TBL foi desenvolvido para cursos de administração nos anos de 1970, por Larry Michaelsen (MICHAELSEN; SWEET, 2008) e posteriormente foi usado também em medicina e implementado em cursos de engenharia (MATALONGA; MOUSQUES; BIA, 2017)

A abordagem TBL centra-se em promover o pensamento crítico de múltiplas formas, a fim de alcançar uma aprendizagem de nível superior (GOMEZ; WU; PASSERINI, 2010). Outra importante característica é a aprendizagem baseada no diálogo e na interação entre os alunos, o que contempla as habilidades de comunicação e trabalho colaborativo em grupos, que será necessária ao futuro profissional (BOLLELA et al., 2014).

Além disso a aprendizagem colaborativa não só promove o engajamento e a aquisição de conhecimento pelos alunos, como também ajuda o desenvolvimento de habilidades de trabalho em grupo, como a comunicação e a colaboração, ambos importantes para a profissão de engenharia. A colaboração apresenta importantes resultados educacionais, como pensamento crítico, raciocínio moral, eficácia intercultural e bem-estar pessoal (CABRERA; VILLALON; CHAVEZ, 2017).

O TBL pode complementar ou até mesmo substituir um curso focado em aulas expositivas (MICHAELSEN; SWEET; PARMELEE, 2008). Não requer múltiplas salas e nem vários docentes atuando, os alunos são induzidos a se prepararem para as atividades do TBL. Os alunos não precisam ter instruções específicas para trabalho em grupo, já que aprendem a medida que as sessões acontecem. Além de ter sua fundamentação teórica baseada no construtivismo que é uma das características das atividades colaborativas, em que o professor se torna facilitador para a aprendizagem. Os alunos tornam-se mestres de suas próprias experiências educacionais passando de um agente passivo para um agente ativo de sua aprendizagem (BOLLELA et al., 2014).

Essa metodologia envolve atividades individuais e em grupos. As atividades individuais visam preparar o aluno para as atividades que se segue. Já nas atividades em grupo, estes trabalham juntos promovendo o aprendizado ativo e efetivo ao longo do semestre, resolvendo problemas, discutindo e aplicar conceitos inicialmente aprendidos através das leituras individuais. Com isso produz maiores conquistas e relacionamentos mais saudáveis e mais positivos entre alunos, do que relacionamentos competitivos ou experiências individuais (GOMEZ; WU; PASSERINI, 2010).

De acordo com (MICHAELSEN; SWEET, 2008) o TBL é regida por quatro princípios:

1. **Os grupos devem ser devidamente formados e gerenciados**: Todo o processo de aprendizagem é realizado em torno da formação e interação de indivíduos em pequenos grupos permanentes e diversificadas.
2. **Os alunos devem ser responsáveis pelo qualidade do seu trabalho**: O TBL deve promover a noção de que os alunos devem ser responsáveis pelo seu aprendizado e pela qualidade de seu trabalho.
3. **As atividades do grupo na classe devem promover o aprendizado e o desenvolvimento do grupo**: Para isso é necessário promover a discussão e a tomada de decisões.
4. **Os alunos devem receber comentários frequentes e imediatos**: O feedback é essencial para o aprendizado e a retenção do conhecimento.

Ao aderir aos quatro elementos essenciais acima, os professores criam um contexto que promove a quantidade e a qualidade da interação dos grupos, tornando-os mais ativos. Além disso, ao longo do tempo, a confiança dos alunos em seu grupo cresce até o ponto em que eles estão dispostos e capazes de enfrentar tarefas difíceis com pouca ou nenhuma ajuda externa (MICHAELSEN; SWEET, 2008).

A organização de uma atividade utilizando o TBL é definida nas seguintes etapas: Primeiro deve ser formado os grupos. Estes devem ser compostos de cinco a sete integrantes e deve ser formado de forma diversificada, oferecendo os recursos necessários.

São fatores dificultadores à coesão do grupo: vínculos afetivos entre os membros, expertise diferenciada, entre outros. A tarefa de formação dos grupos devem ser realizadas somente pelo professor. Após isso deve ser realizada as etapas propostas pelo TBL (BOLLELA et al., 2014).

Cada módulo, unidade de trabalho ou tema coerente dentro do curso, segue um processo de aprendizagem iterativo que repete uma sequência de atividades propostas pelo TBL especificado na figura 1:

1. **Preparação individual** através de leitura fora da classe dos materiais disponibilizados e exercícios realizados.

2. **Avaliação de prontidão ou garantia de preparo** (RAT – *Readiness Assurance Test*) através de testes individuais (iRAT) e em grupo (gRAT). Nesta etapa, as atividades desenvolvidas buscam checar e garantir que o aluno está preparado e pronto para resolver testes individualmente, para contribuir com o seu grupo e aplicar os conhecimentos na etapa seguinte do TBL. Nesta etapa também é aplicado a apelação caso algum aluno se oponha ao resultado da avaliação.

3. **Aplicação dos conhecimentos** adquiridos pelo grupo por meio da resolução de problemas reais, essa atividade deve ocupar a maior parte do tempo.

4. **Avaliação em pares** para avaliar o desempenho de cada membro do grupo.

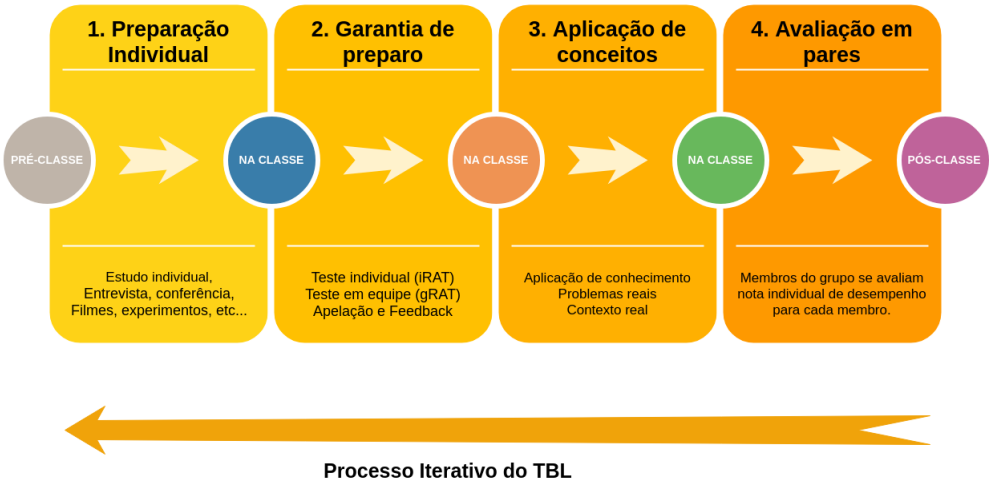


Figura 1 – Processo iterativo TBL. Fonte: Autor, baseado em (BOLLELA et al., 2014)

Para (BOLLELA et al., 2014) um aspecto crítico para implantar o TBL é a colaboração dos alunos dentro dos grupos. É importante que os membros do grupo busquem uma boa interação para que não haja conflitos. A intervenção do professor deve ser adiada o máximo possível, permitindo que o próprio grupo busque a solução de seus problemas principalmente na etapa de aplicação de conceitos.

Ele deve desenvolver questões ou testes que exijam dos grupos uma resposta (um produto) que possa ser facilmente observada e comparada entre os outros grupos e com possibilidade de incluir a perspectiva do especialista. Por isso a aplicação de apenas algumas etapas do TBL é alvo de críticas de seus criadores (MICHAELSEN; SWEET, 2008). Embora customizações sejam inevitáveis, é necessário ter a clareza de que, com isto, nem todos os benéficos atributos da metodologia serão alcançados.

A etapa de aplicação do conhecimento deve ser estruturada seguindo alguns preceitos. Os quatro princípios básicos para elaborar esta fase são conhecidos em inglês como

os 4 S's: Problema significativo e real (*Significant*), O mesmo problema para todas os grupos (*Same*), Respostas curtas e específicas (*Specific*) e Relatos simultâneos (*Simultaneous report*), ou seja, as respostas devem ser mostradas simultaneamente, de modo a inibir que alguns grupos manifestem sua resposta a partir da argumentação de outros grupos (MICHAELSEN; SWEET, 2008).

2.2 Detalhando cada etapa do TBL

2.2.1 Etapa 1: Preparação individual

Os alunos devem ser responsáveis por se prepararem individualmente para as etapas seguintes do TBL. Se os alunos não se prepararem eles não serão capazes de contribuir para o desempenho do seu grupo. A falta desta preparação dificulta o desenvolvimento de coesão do grupo e resulta em ressentimento por parte do mesmo que se preparou (MICHAELSEN; SWEET, 2008).

2.2.2 Etapa 2: Garantía de preparo

O RAT: *Readiness assurance test* (Avaliação de garantia de preparo) que deve ser realizado de maneira individual iRAT e depois em grupo gRAT. É o mecanismo básico que garante a responsabilidade individual pela preparação (MICHAELSEN; SWEET, 2008). É dividido em 4 atividades:

1. **Avaliação iRAT**: Respondido individualmente sem consulta a qualquer material, consiste de algumas questões de múltipla escolha contemplando os conceitos mais relevantes das leituras ou das atividades indicadas previamente. cada questão irá ter quatro alternativas e o aluno poderá distribuir quatro pontos entre as alternativas da forma que desejar.

2. **Avaliação gRAT**: Respondido em grupo sem consulta, os alunos devem discutir as mesmas questões e cada membro deve defender e argumentar as razões para sua escolha até o grupo decidir qual é a melhor resposta, exercitando suas habilidades de comunicação, argumentação e convencimento. Consiste das mesmas questões do iRAT, terá feedbacks imediatos da resposta certa.

3. **Apelação**: Após as avaliações os grupos podem recorrer a apelação no caso de não concordar com a resposta indicada como correta, todo apelo deve ser feito acompanhado de argumentação, e com consulta a fontes bibliográficas pertinentes.

4. **Feedback pelo professor**: Assim que as atividades acima forem concluídas o professor, buscando clarear conceitos fundamentais, oferece feedback a todos simultaneamente, de acordo com as questões que mais gerou dificuldade nos alunos. Ao final

desta etapa, os alunos devem estar confiantes a respeito dos conceitos fundamentais e poderão aplicá-los para resolver problemas mais complexos que serão oferecidos na etapa de aplicação do conhecimento, que se segue no processo do TBL.

2.2.3 Etapa 3: Aplicação de conceitos

Essa etapa é responsável pela aplicação do conhecimento adquiridos por meio da resolução de situações, problemas ou cenários relevantes e presentes na prática profissional diária do aluno, ou seja, problemas reais que ele pode enfrentar, preparando-os para o que o mercado cobra, deve ocupar a maior parte da carga horária. O fundamental é que todos os grupos estejam preparados para argumentar sobre a escolha que fizeram (MICHAELSEN; SWEET, 2008).

2.2.4 Etapa 4: Avaliação em pares

De acordo com (BOLLELA et al., 2014) os alunos são avaliados pelo seu desempenho individual e também pelo resultado do trabalho em grupo, além de se submeterem à avaliação entre os pares, o que incrementa a responsabilização. A avaliação pelos pares é essencial, pois, os componentes do grupo são, normalmente, os únicos que têm informações suficientes para avaliar com precisão a contribuição do outro.

A avaliação em pares, cada integrante do grupo avaliará os outros integrantes, será disponibilizado uma pontuação máxima para cada membro distribuir entre os outros membros e um campo para dizer o porquê da pontuação. Essa avaliação não precisa se identificar, ela é anônima e o cálculo das pontuações fará parte da nota individual de cada membro.

Conclui-se, assim, um módulo ou unidade educacional TBL.

2.3 Preparo de um módulo em TBL

De acordo com (BOLLELA et al., 2014), três mudanças são necessárias quando se modifica a estratégia pedagógica de uma aula expositiva, centrada no professor, para uma atividade do tipo TBL, centrada no aluno:

- Os objetivos primários devem ser trabalhados apenas com conceitos chaves para objetivos que incluem a compreensão de como estes conceitos devem ser aplicados em situações reais.
- O professor passa de alguém que oferece informações e conceitos, para alguém que facilita o aprendizado, ou seja, ele entrega a responsabilidade de gerar conhecimento para o aluno.

- É necessária uma mudança no papel e função dos alunos, que saem da posição de receptores passivos da informação para responsáveis pela aquisição do conhecimento e membros integrantes de um grupo que trabalha de forma colaborativa para compreender como aplicar o conteúdo na solução de problemas realísticos e contextualizados.

Segundo (BOLLELA et al., 2014) uma atividade inicial de treinamento usando o TBL com os alunos deve ser preparada para a primeira aproximação dos estudantes com a metodologia.

2.4 Pontos positivos e negativos da metodologia

Foram encontrados na literatura alguns pontos negativos sobre a implantação do TBL, entre eles temos: resistência dos alunos, pois esperavam aulas formais (DAVIS et al., 2013) e (MATALONGA; MOUSQUES; BIA, 2017), Insatisfação dos alunos devido a limitação de ferramentas online como fóruns de discussão (AWATRAMINI; ROVER, 2015), queixas da demanda de leitura com alto consumo de tempo extraclasse, além de alguns alunos reclamaram da falta de comprometimento de alguns colegas do grupo (RAMOS et al., 2018).

E vários pontos positivos foram encontrados:

- Em uma experiência do uso do TBL, no curso de Engenharia de Software da Faculdade do Gama, iniciada no ano de 2017 com duração de 3 semestres, evolvendo 5 disciplinas e uma média de 500 alunos. Estes relataram uma preferência pelo uso do TBL quando comparado ao uso de aulas expositivas convencionais, vários motivos se deram em relação a isso, entre eles: maior interação e troca de conhecimento, discussão de temas abordados de forma a promover maior diversidade de pontos de vista, maior fixação do conteúdo e participação mais ativa em sala de aula, feedbacks imediatos, maior interesse na aquisição do conhecimento devido a parte prática da metodologia, entre outros. E também teve vários pontos positivos especificados pelos professores (RAMOS et al., 2018);

- Em um curso de programação de primeiro semestre houve um aumento de 50% para 75% da taxa de aprovação dos alunos do curso e diminuição das taxas de abandono (MATALONGA; MOUSQUES; BIA, 2017);

- Em outro curso de programação o desempenho dos alunos melhoraram com uma abordagem mista, mistura de TBL com palestras formais, além da pesquisa de amostragem positiva dos alunos (ELNAGAR; ALI, 2012);

- Na universidade ORT no Uruguai no curso de engenharia de software os alunos tiveram uma percepção positiva de sua aprendizagem ao comparar o curso com TBL com os cursos tradicionais. Eles não acharam o estudo de todas as unidades temáticas estressantes e acreditam que o TBL atende ao seu estilo de aprendizagem individual (MATALONGA; MOUSQUES; BIA, 2017);

- Em um curso introdutório de programação, os resultados mostraram que o TBL ajudou os alunos a alcançar um nível mais alto de compreensão em um curto período de tempo (CABRERA; VILLALON; CHAVEZ, 2017);

- No curso de Engenharia de informática sobre o design de sistemas embarcados a pontuação alcançada usando o TBL foi relativamente boa comparada às aulas tradicionais e em laboratório (AWATRAMINI; ROVER, 2015);

- Em um curso de desenvolvimento profissional da universidade de Oklahoma os comentários finais dos alunos foram satisfatórios, por exemplo, muitos conseguiram sair da zona de conforto, aprenderam habilidades valiosas como: trabalho em equipe, escrever documentos técnicos, apresentar os resultados da sua pesquisa e ideias e falar e argumentar em público (DAVIS et al., 2013).

Com a análise da implantação do TBL em diversos cursos de universidades diferentes observamos que houve em praticamente todos os casos uma aceitação positiva por parte dos envolvidos, tornando a metodologia bastante aceita por parte dos alunos.

De acordo com (RAMOS et al., 2018) o TBL prepara os alunos para o desenvolvimento de capacidades previstas no Projeto Pedagógico do curso de engenharia de software. Por exemplo, em termos de capacidade técnica temos estudos teóricos e resolução de problemas na fase de aplicação de conceitos. Capacidade escrita temos a redação de apelações quando um aluno não concorda com algo em relação à avaliação e quanto a capacidade oral os alunos apresentam argumentações ao responder as avaliações em grupo, há uma discussão para chegar-se a um consenso da resposta correta.

3 Metodologia

Neste capítulo, será mostrado a metodologia de desenvolvimento e de TCC proposto neste trabalho. O capítulo está dividido em seções que estão disposta da seguinte maneira: na seção 3.1 será mostrado a métodologia do TCC1 e TCC2 e como ela foi aplicada e na seção 3.2 será mostrado todo o processo de desenvolvimento do software de uma forma generica.

3.1 Metodologia do TCC

Para o desenvolvimento do trabalho de conclusão de curso, foi definido o seguinte processo da figura 2

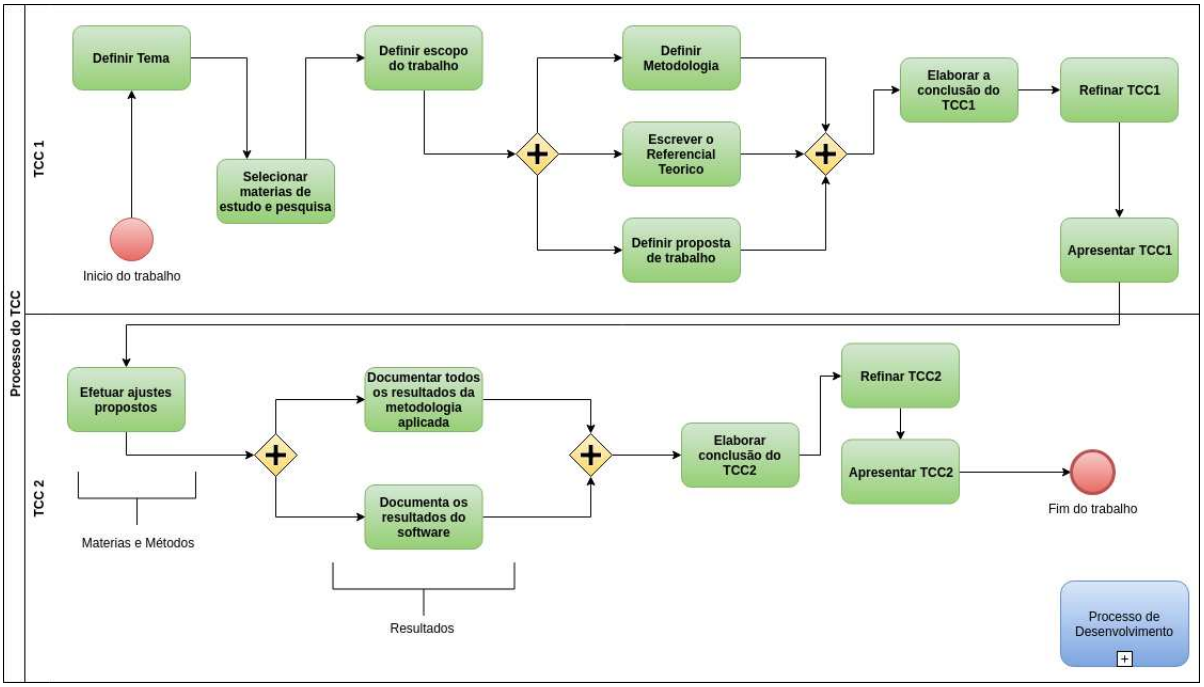


Figura 2 – Processo do TCC. Fonte: Autor

O processo foi dividido em dois grandes marcos: O primeiro é o planejamento da ferramenta e o segundo é a execução e resultados da aplicação de tudo que foi planejado, ou seja, é a entrega do produto, porém de acordo com a metodologia ágil utilizada o segundo marco se iniciará junto com o primeiro.

3.1.1 Marco 1: Trabalho de Conclusão de Curso 1

O TCC se iniciou com a definição do tema do trabalho, ao definir o tema foi selecionado materiais de estudo e pesquisa como artigos, livros, sites e outras referências

importantes para elaborar o embasamento teórico do projeto. Assim que se teve uma boa base de conteúdo foi definido o escopo do trabalho, restringindo o que será abordado e o que será descartado.

Com o escopo bem definido se iniciou a escrita do trabalho. Em paralelo foi definido a metodologia do TCC e a metodologia de desenvolvimento, além de documentar todos os processos que foram realizados dentro dessas metodologias; o referencial teórico que serviu de base para o andamento do projeto; a proposta de trabalho que deu uma visão geral do que é o software. O processo de desenvolvimento foi realizado em paralelo com toda a metodologia do TCC.

Em seguida foi elaborado a conclusão do trabalho (TCC1) e uma verificação geral no documento, ou seja, melhorar alguns conteúdos, ortografia, gramática e outros erros encontrados. Com isso foi elaborado os slides da apresentação do TCC 1 finalizando o marco 1 do projeto.

Em concordância com o processo de condução do TCC1 foi elaborado o cronograma da figura 3:

Cronograma para o TCC 1 (2018)				
Atividade	**Agosto**	**Setembro**	**Outubro**	**Novembro**
Definir Tema	X			
Selecionar materias de estudo e pesquisa	X			
Definir escopo do trabalho	X			
Definir Metodologia		X		
Escrever o Referencial Teórico		X		
Definir Proposta de Trabalho		X		
Elaborar conclusão do TCC 1			X	
Refinar TCC 1			X	X
Apresentar TCC 1				X

Figura 3 – Cronograma do TCC 1. Fonte: Autor

3.1.2 Marco 2: Trabalho de Conclusão de Curso 2

Dando entrada no marco 2 do projeto foi feito os ajustes necessários do TCC 1 para o trabalho de conclusão de curso 2 ou TCC 2. Com isso foi documentado de forma sucinta

os resultados da metodologia aplicada no processo de desenvolvimento. E em paralelo foi documentado os resultados do software e sua aplicação em um contexto real.

Com os resultados documentados foi elaborado a conclusão do TCC 2 e feito uma verificação geral no documento para melhorá-lo. Com o trabalho pronto, foi criado os slides da apresentação do TCC 2 e assim foi finalizado o marco 2 do projeto.

Em concordância com o processo de condução do TCC2 foi elaborado o cronograma da figura 4:

Cronograma para o TCC 2 (2019)				
Atividade	**Março**	**Abril**	**Maio**	**Junho**
Efetuar Ajustes Propostos	X			
Documentar todos os resultados da metodologia proposta		X	X	
Documentar os resultados do software		X	X	
Elaborar a conclusão do TCC 2			X	
Refinar TCC 2			X	
Apresentar TCC 2				X

Figura 4 – Cronograma do TCC 2. Fonte: Autor

3.2 Metodologia de Desenvolvimento

Para se construir um software é necessário seguir uma série de passos previsíveis, esses passos estão definidos no processo de software. De acordo com (PRESSMAN, 1997) um processo de software pode ser visto como um conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software.

Para a construção do processo há uma adaptação em 3 principais níveis de acordo com o (SAFE. . . , 2018) (portfólio, programa e time):

- O nível de portfólio foi utilizado para gerar boa parte da documentação do projeto aplicando as 3 atividades principais da engenharia de requisitos, que são elicitação, modelagem e análise;
- O nível de programa tem como artefatos o documento de arquitetura, as ferramentas de UX e o product backlog com sua rastreabilidade;

- O nível de time que é onde codificar a solução, está dividido em desenvolvimento que executa a sprint e gestão na qual aplica as práticas do scrum, além da infraestrutura e dos testes.

A construção do software foi feito seguindo uma adaptação das metodologias ágeis SCRUM, XP e SAFe para aumentar a produtividade e eficiência do mesmo, já que o projeto todo foi realizado por uma única pessoa, ou seja, teve toda uma adaptação para isso.

Foi decidido o SCRUM devido a grande familiaridade que o desenvolvedor tem com a metodologia, ela é bastante utilizada no mercado e muito útil para organização, controle e gerenciamento do projeto. O XP também já é bem utilizado e conhecido por ser uma metodologia de desenvolvimento de software que ajuda a criar sistemas de melhor qualidade e o SAFe, este é bastante útil para padronizar o processo de forma organizada.

Para a execução, foi realizado algumas adaptações:

- **Papeis**: A metodologia apresenta alguns papéis que devem ser focados pela equipe, porém como o projeto foi todo realizado por um único membro os papéis são todos realizados pelo mesmo.
- **SCRUM**: O SCRUM tem alguns rituais como o daily que para um único membro não se torna necessária. Rituais como retrospectiva e revisão não foram feitas de acordo com a metodologia, pois os interessados no projeto não tem muito tempo para realizar com frequência esses rituais, porém foi feito em forma de texto e documentado na wiki do projeto.
- **XP**: Uma das práticas citadas do XP que não foi utilizada é a programação pareada.
- **SAFe**: Do SAFe foi aproveitado somente os padrões de organização do processo em níveis e a rastreabilidade dos requisitos.
- **Verificação e validação**: No contexto de verificação e validação foi aplicado apenas as técnicas dinâmicas, ou seja, os testes. Os testes unitários e de integração foram automatizados e foi baseados em técnicas caixa preta, particionamento de equivalência e análise de valor limite para a construção dos códigos de teste, já os testes de sistema foi realizado manualmente, entre eles temos: documentação, treinamento, teste de aceitação e teste de instalação.
- **Medição e Análise - GQM**: Para o GQM foi coletado métricas de código, para verificar a qualidade do mesmo, tendo como objetivo a manutenabilidade e uma métrica de equipe para verificar a evolução do conhecimento do desenvolvedor durante o projeto.

- **DevOps**: O DevOps tem como foco a automação de algumas atividades necessárias para agilizar e entregar código de qualidade. Entre as atividades as que foram automatizadas são: testes, ambientes e infraestrutura de instalação, coleta de algumas métricas, integração contínua e deploy.
- **Software Livre**: A licença escolhida para o software é a General Public License (GPL) versão 3, pois ela é a que melhor se encaixa no contexto Open Source em relação as liberdades especificadas.

O gerenciamento da sprint foi feito por quadros Kanban, na qual teremos nove quadros principais. Esses quadros se encontram nos Boards do github, utilizando como base o plugin chamado zenhub. Para instalar é só baixar o plugin e instalar no navegador, a partir daí já conseguirá ver toda organização das issues do projeto a partir da ferramenta. Os quadros são:

- **Épicos**: Épicos do software.
- **Features e Enables**: *Features* e *Enables* do software.
- **Novas Issues**: Novas histórias de usuário, bug reports ou qualquer tipo de situações de trabalho relacionadas ao desenvolvimento da aplicação que ainda não foram mapeadas, pontuadas ou priorizadas e consequentemente alocadas nos demais quadros.
- **Product Backlog**: *Board* para histórias/tarefas ou correções já mapeadas e priorizadas. É importante esclarecer que nesse quadro, a prioridade é definida pela posição da *issue* na *board*, sendo que as posições superiores determinam maior relevância para o projeto.
- **Sprint Backlog**: *Issues* alocadas para os desenvolvedores na *sprint* corrente ou *bug fixes* de alta prioridade.
- **Congeladas**: Issues congeladas por dependência de outras, por alguma necessidade de reavaliação da necessidade ou do esforço necessário para concluí-la, por dependência de funcionalidades externas que não são disponibilizadas pelos mantenedores, por temporária incapacidade da equipe de solucionar determinado problema, etc.
- **Revisão**: Tarefas concluídas que necessitam de revisão para entrarem na aplicação.
- **Feitas**: Código já revisado que pode ser anexado a aplicação na próxima release.
- **Closed**: *Issues* fechadas já anexado a aplicação.

Na figura 5 está definido o processo de desenvolvimento que está no nível de programa e time.

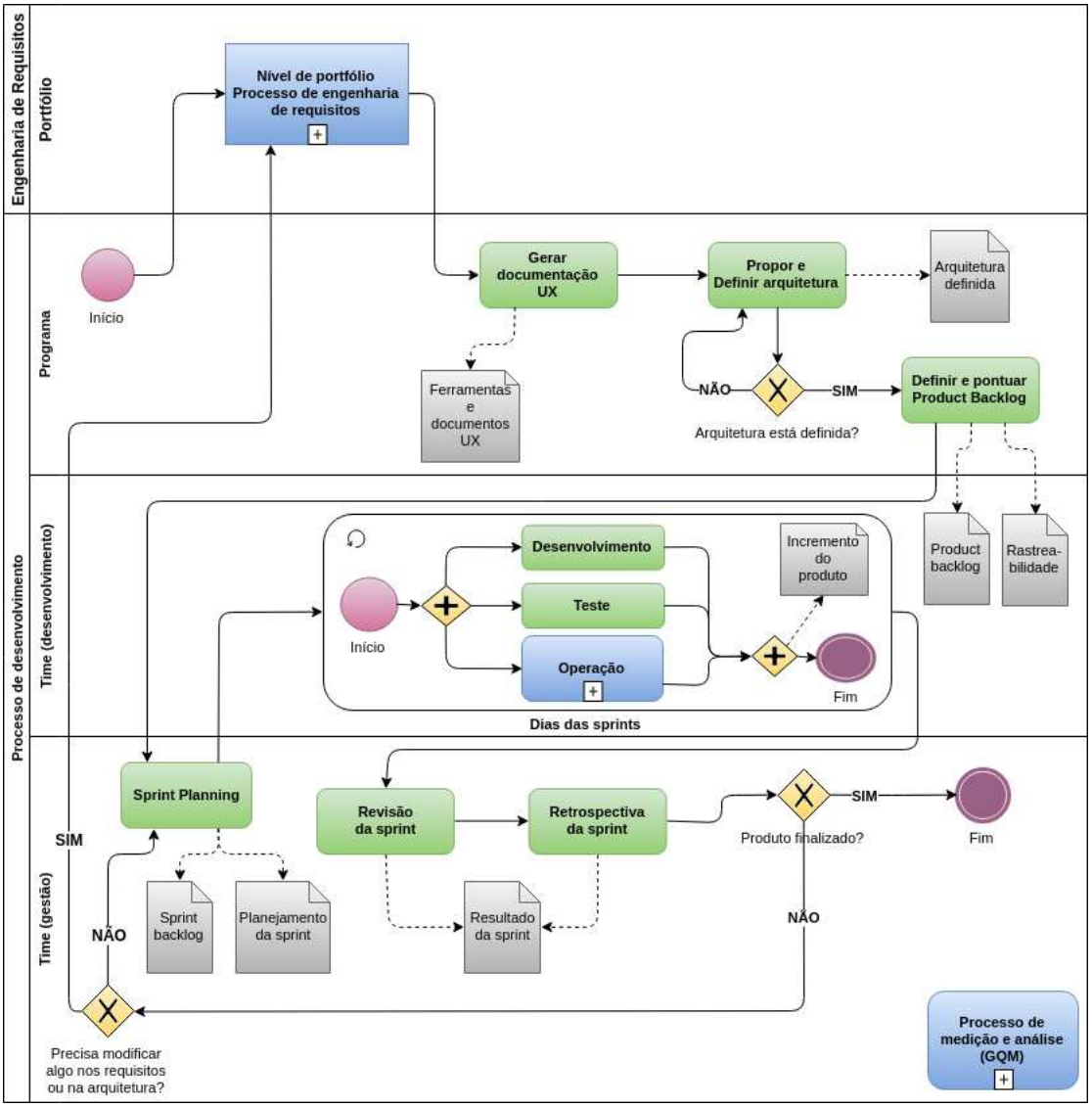


Figura 5 – Processo de desenvolvimento. Fonte: Autor

3.2.1 Nível de Portfólio

Dentro da Engenharia de Software vários modelos definem as etapas necessárias para se construir um software de acordo com o processo definido, mas todos têm algo em comum: uma etapa dedicada a compreensão dos problemas a serem solucionados e a definição do quê será feito. Esta etapa inicial recebe o nome de engenharia de requisitos e está localizado no nível de portfólio do SAFe (PRESSMAN, 1997).

O nível de portfólio definido para a etapa de engenharia de requisitos do software PGTBL segue três fases principais:

3.2.1.1 Elicitação

Em paralelo ao processo do TCC o processo de desenvolvimento se inicia com o subprocesso de engenharia de requisitos na parte de elicitação.

Essa atividade preocupa-se com o levantamento dos principais aspectos, sejam eles funcionais e/ou não funcionais. Podemos dizer que passamos de um alto nível de abstração para algo mais próximo do mundo real utilizando várias técnicas de elicitação. Ela se iniciou com a identificação do problema e o contexto na qual o software foi aplicado, além definir o tema de investimento. É nessa fase que é identificado os épicos (requisitos funcionais) e enables (requisitos não funcionais) através de técnicas de elicitação como brainstorms e prototipação. Com isso os épicos (EP) são separados em pedaços menores chamados Features (EN).

3.2.1.2 Modelagem

Na fase de modelagem objetiva-se criar um modelo organizacional em contexto com o uso do software, diagramas que auxilia na construção das funcionalidades por meio da modelagem UML, por exemplo, diagrama de classe, além de criar a modelagem do banco de dados relacional utilizando a modelo entidade relacionamento (MER). Na parte não funcional foi criado diagramas que auxilia a identificação e rastreabilidade de requisitos não funcionais como metas a serem alcançadas através do framework NFR.

3.2.1.3 Análise

Na fase de análise objetiva-se criar todos os documentos relacionados ao gerenciamento do projeto, por exemplo, documento de abertura de projeto, plano de gerenciamento de recursos humanos, comunicação, risco, medição e análise, configuração de software, aquisição e custos. Além disso nessa fase foi elaborado o documento de visão do projeto.

3.2.2 Nível de Programa

Nível responsável pela auto-organização de times ágeis, entrega contínua de valor, criação de Features por meio dos épicos encontrados, realização de toda a documentação relacionada ao *User Experience* ou UX, e definição do documento de arquitetura do software.

No nível de programa foi definido a arquitetura que melhor se adapte ao projeto, utilizaremos de algumas técnicas de UX e UI para melhorar a usabilidade do software e foi elaborado e pontuado as histórias de usuário do product backlog, essas foram derivadas das features definidas no nível de portfólio.

3.2.3 Nível de Time

Nível responsável pelo auto-gerenciamento da equipe ágil, incremento do software totalmente testado, práticas SCRUM e XP, descrição do valor por meio de histórias de

usuário e tarefas.

O nível de time se inicia com a *Sprint Planning*, na qual será priorizada as histórias de usuário que serão implementadas na *sprint* que se segue. Com isso inicia-se os ciclos de desenvolvimento das sprints. Dentro dessa atividade temos em paralelo o desenvolvimento, teste e operações (DevOps) tendo como resultado o incremento do produto.

Ao finalizar o ciclo de desenvolvimento será realizado a revisão da sprint, em que será apresentado as funcionalidades implementadas e verificar se há alguma dívida técnica para a próxima iteração que se segue. Em seguida será realizada a retrospectiva da sprint que será coletado pontos positivos, negativos e melhorias para as próximas iterações.

Todo o processo foi executado de forma iterativa e incremental, ou seja, a cada iteração chamada de sprint, foi retomado e avaliado se houve necessidade de alguma modificação ou incremento nesses diagramas e documentos. Além de que na finalização de cada fase teve uma atividade de revisão dos documentos e em paralelo com todo o processo teve o processo de medição e análise responsável pela coleta de métricas de qualidade.

4 Resultados

Neste capítulo, será mostrado todos os resultados do software proposto neste trabalho. O capítulo está dividido em seções que estão disposta da seguinte maneira: na seção 4.1 será mostrado os requisitos funcionais e não funcionais do software lincando com uma ilustração desse requisito no sistema; na seção 4.2 será mostrado a arquitetura do software, ou seja, sua representação arquitetural, pacotes e diagrama de classe e na seção 4.3 será mostrado as métricas de qualidade coletadas durante o desenvolvimento do software.

4.1 Requisitos do Software

4.1.1 Recursos do produto (Requisitos funcionais ou Épicos)

Tabela 1 – Requisitos Funcionais ou Épicos do produto

Recurso	**Descrição**
Administrar Disciplinas e alunos	O professor pode adicionar, remover, criar e editar disciplinas e turmas e disponibilizar a senha de acesso a elas para os alunos entrarem, além de poder remover ou adicionar estudantes na turma e gerenciar suas notas.
Criar conta	O professor e os alunos pode criar sua contas no sistema, apenas passando seus dados pessoais como nome, email, senha, e usuário
Gerenciar dados pessoais	O usuário poderá editar sua senha e dados pessoais do usuário conforme necessário.
Funcionalidades do TBL	Funcionalidades relacionadas às fases de preparação, garantia de preparo ou RAT, iRAT, gRAT e apelações, Aplicação de conceitos e avaliação em pares
Relatório	O professor terá um dashboard com relatórios do desempenho dos alunos em cada questão da avaliação, tendo um feedback para o que ele deve focar mais nas aulas.
Gamificação	Terá também um rank de grupos, na qual o primeiro colocado ficará exposto no Hall da Fama que será visto por novos alunos dos próximos semestres, não há rank individual porque o objetivo não é a competição e sim a colaboração.

No apendice B será exemplificado cada requisito funcional dentro do software, porém não irá mostrar todas as funcionalidades do mesmo.

4.1.2 Requisitos de qualidade (Não funcionais ou Enables)

Além do requisitos funcionais, foram identificados como importantes os seguintes requisitos de qualidade:

- **Sistema**: O sistema deve seguir a arquitetura MVT definida no documento de arquitetura e as ferramentas de desenvolvimento será o Python (versão 3.5) e o framework Django (versão 2.0) tendo atualização constante.
- **Suportabilidade**: O sistema poderá ser acessado em computadores pessoais - notebook, desktop – utilizando-se de um serviço de internet. Sendo uma aplicação Web compatível com os principais sistemas operacionais (Linux, Mac, Windows), acessada através do navegador Google Chrome e/ou Firefox de um dispositivo móvel ou fixo.
- **Qualidade**: O sistema deve seguir uma folha de estilo a ponto de o código ser legível e de fácil manutenção, tendo como base boas práticas de programação, o sistema deve ter baixo acoplamento e alta coesão além de ser modularizado focando na flexibilidade e manutenção do mesmo.
- **Usabilidade**: O sistema deve ser responsivo, adaptando-se à plataforma que o usuário estiver utilizando e o design deve ser fácil de usar e aprender.
- **Desempenho**: Por ser um sistema web o software necessita de uma conexão estável com a internet para seu funcionamento. A velocidade da internet tem impacto direto no desempenho da aplicação, sendo necessário uma velocidade suficiente para processar as informações e executar as funcionalidades do sistema.
- **Confiabilidade**: O sistema deve apresentar uma boa percentagem de cobertura de testes automatizados, mínimo de 70% e bastante encapsulamento de código, além de um sistema de log eficiente e ter segurança.
- **Segurança**: O sistema deve se comprometer em apresentar informações confiáveis para o usuário do sistema, entretanto dependerá diretamente dos demais usuários, uma vez que o conteúdo apresentado será de autoria destes e deve apresentar um sistema de autenticação, autorização e recuperação seguro e eficiente.

4.1.3 Ferramentas e Tecnologias

Abaixo será listado todas as ferramentas e tecnologias que foram utilizadas para a realização do projeto.

- **Draw.io**: Criação de diagramas.

- **Google Drive**: Armazenamento e edição colaborativa dos artefatos e documentos do projeto.
- **Git e Github**: Controle de versão do código para um bom gerenciamento do mesmo.
- **Zenhub**: Permitir o gerenciamento constante das tarefas que serão realizadas durante o projeto.
- **Linux ubuntu**: Ambiente de desenvolvimento, homologação, produção e teste.
- **Whatsapp e Email**: Permite a comunicação entre os membros da equipe e os stakeholders, além da comunicação presencial.
- **Travis CI**: Ferramenta responsável por realizar a integração contínua das funcionalidades realizadas
- **Codacy**: Ferramenta para cobertura de testes e análise estática de código, usado para coletar métricas e melhorar a qualidade do código.
- **Docker**: Ferramentas responsáveis por criar um ambientes de desenvolvimento, teste, homologação e produção
- **DockerHub**: Repositório para armazenar as imagens de cada ambiente.
- **Python 3.5**: Linguagem de programação para a criação do software.
- **Django 2.0**: Framework utilizado para a criação do software.
- **Makefile**: Usado para facilitar a execução de scripts.
- **mkdocs**: Usado para criar a documentação do software.

4.1.4 EAP - Estrutura Analítica do Produto

A Estrutura Analítica do Produto, foi utilizada como entrada do processo fornecendo os pacotes de trabalho previamente discutidos. Esses pacotes foram então decompostos de maneira a gerar suas respectivas atividades.

A EAP se encontra no apêndice A

4.1.5 Marcos do projeto

O projeto tem como base dois principais marcos definidos na tabela 2, que representam entregas do produto, são eles: *Release 01* e *Release 02*.

Tabela 2 – Marcos do projeto

Marco	Data	Atividade
Início do projeto	03/10/2017	Começo do projeto
Release 01	01/08/2018	Entrega da primeira versão funcional do sistema com algumas funcionalidades implementadas e testadas, além de sua documentação
Release 02	01/03/2019	Entrega do versão final do projeto com as funcionalidades restantes do escopo previamente definido com a utilização da abordagem ágil.

4.2 Métricas GQM

Todas os indicadores e métricas aqui documentados foram tirados da ferramenta de análise estática Codacy utilizado para coletar e análisar as métricas do software de forma automatizada. Apenas as métricas relacionadas ao planejamento e build que foi coletada manualmente pela ferramenta github.

4.2.1 Planejamento e Build

Figura 6 – Pontos Planejados x Pontos Concluídos. Fonte: Autor

Figura 7 – Build. Fonte: Autor

Resultado: Como podemos ver, a build está funcionando perfeitamente e foi executado todos os pontos definidos no planejamento do projeto.

4.2.2 Objetivo 01: Qualidade de código

4.2.2.1 Questão 01: A qualidade do código está aceitável?

4.2.2.1.1 Métrica: Quantidade de issues geradas pela ferramenta de análise estática

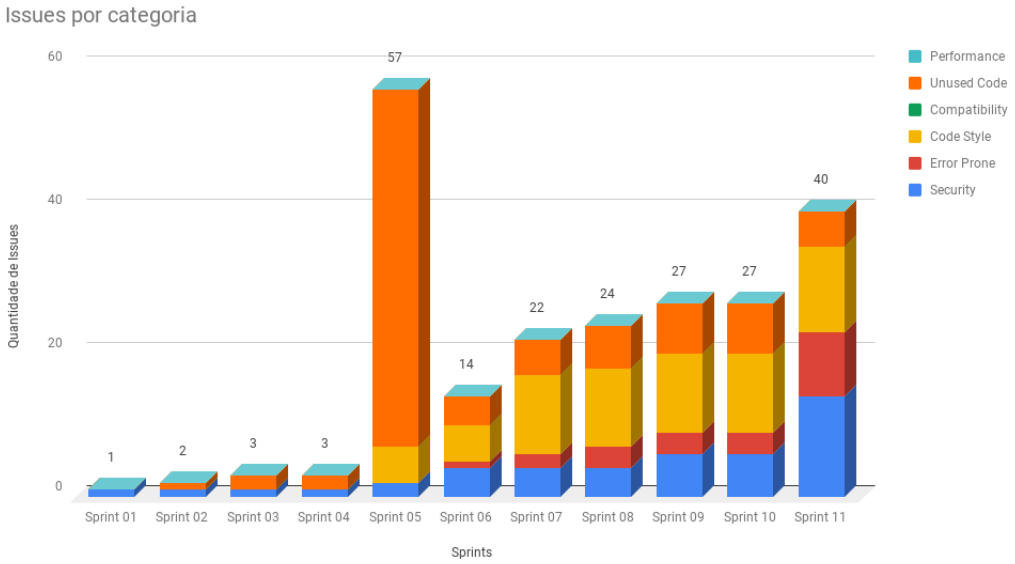


Figura 8 – Issues por Categoria. Fonte: Autor

- **Descrição**: O resultado desta métrica será dado pela quantidade de issues que a ferramenta de análise estática de código encontrou, a ferramenta utiliza folhas de estilos normalmente adotadas para padronizar as linguagens, por exemplo a pep8 que é uma folha de estilo para a linguagem python padronizando seu código. Além disso cada issue é separado em categorias, entre elas as mais importantes são Style, Error Prone, Unused code, security.
- **Procedimentos**: Verificar a quantidade de issues por categoria geradas pela ferramenta de análise estática Codacy e o nível de gravidade de cada uma.
- **Análise**:
  - **Valor Aceitável**: Menos de 50 issues, Refatorar sempre que possível o código até atingir o valor ótimo.
  - **Valor Ótimo**: Nenhuma issue, Manter a qualidade do código estável.
  - **Valor Preocupante**: Mais que 50 issues, Priorizar a refatoração do código com urgência utilizando de preferência folhas de estilo, e focar todos os esforços possíveis a fim de diminuir a quantidade de issues.
- **Resultados**: Tivemos ao final do projeto apenas 40 Issues divididas em suas categorias, o que de acordo com a métrica é um valor aceitável.

4.2.2.1.2 Métrica: Complexidade ciclomática

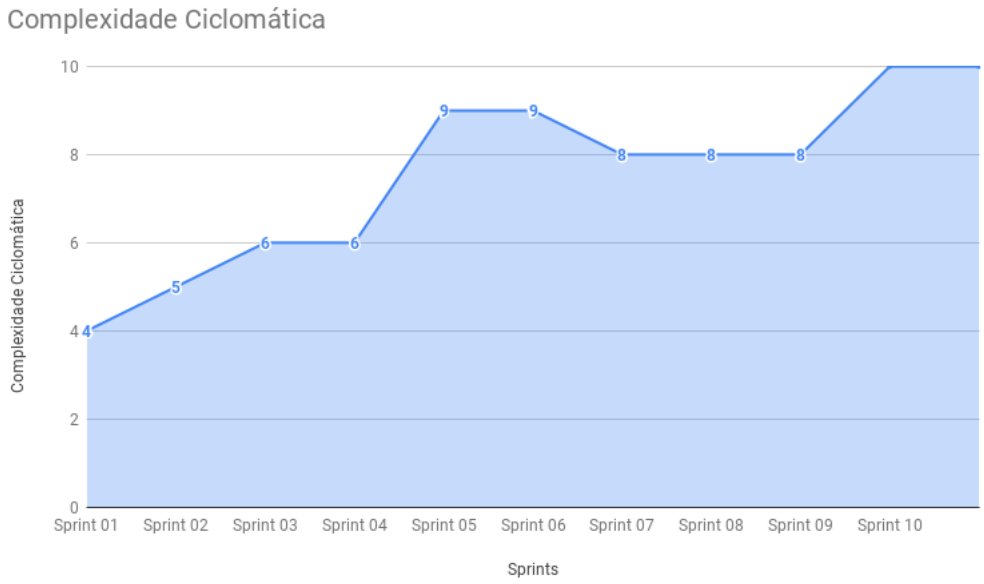


Figura 9 – Complexidade Ciclomática. Fonte: Autor

- **Descrição**: O resultado desta métrica é realizado por meio de porcentagens, complexidade entre 0% e 20%, é um valor de ótimo para aceitável, complexidades acima de 20%, gera complexidade computacional custosa podendo refletir em tempo de espera em consultas.
- **Procedimentos**: Na nova versão do Codacy a complexidade ciclomática do projeto vem em porcentagem.
- **Análise**:
  - **Valor Aceitável**: Complexidade entre 5% a 20%, refatorar sempre que possível o código até atingir o valor ótimo.
  - **Valor Ótimo**: Complexidade abaixo de 5%, manter a qualidade do código estável.
  - **Valor Preocupante**: Complexidade acima de 20%, priorizar a refatoração do código com urgência, e focar todos os esforços possíveis a fim de aumentá-lo para o valor ótimo.
- **Resultados**: Como podemos observar a complexidade ficou em torno de 10%, ou seja, está com um valor de aceitável para ótimo.

4.2.2.1.3 Métrica: Índice de qualidade (Certification)

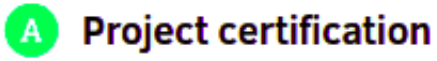


Figura 10 – Certificação de Projeto. Fonte: Autor

- **Descrição**: O resultado desta métrica será dado em um índice entre F e A, onde F significa que o código está horrível, D significa que o código está ruim, C significa nota que o código está na média, B o código está bom e A o código está com uma qualidade excelente.
- **Procedimentos**: Verificar o índice de qualidade disponibilizado pela ferramenta codacy que indica a qualidade de código total de um projeto.
- **Análise**:
  - **Valor Aceitável**: B e C, refatorar sempre que possível o código até atingir o valor ótimo.
  - **Valor Ótimo**: A, manter a qualidade do código estável.
  - **Valor Preocupante**: D e F, priorizar a refatoração do código com urgência, e focar todos os esforços possíveis a fim de aumentar esse índice.
- **Resultados**: A qualidade do código sempre se manteve estável em nível ótimo.

4.2.2.2 Questão 02: O software está com uma cobertura de código aceitável?

4.2.2.2.1 Métrica: Cobertura de Testes

- **Descrição**: O resultado desta métrica será dado em um valor de porcentagem que variará entre 0% e 100%, onde 0% significa que nenhum dos arquivos analisados pela ferramenta foi testado e 100% indica que todas as linhas de códigos forám testadas, mas não quer dizer que há uma cobertura completa de testes do software, ou seja, que testa todos os possíveis casos.
- **Procedimentos**: Foi verificado utilizando uma ferramenta de cobertura de teste chamada Codacy que rodará dentro do software, ela gera um tracking dos arquivos que necessitam de teste. Esse tracking demonstra uma porcentagem para cada arquivo, que indica a cobertura de testes deste e além disso indica a porcentagem total de cobertura do projeto.
- **Análise**:

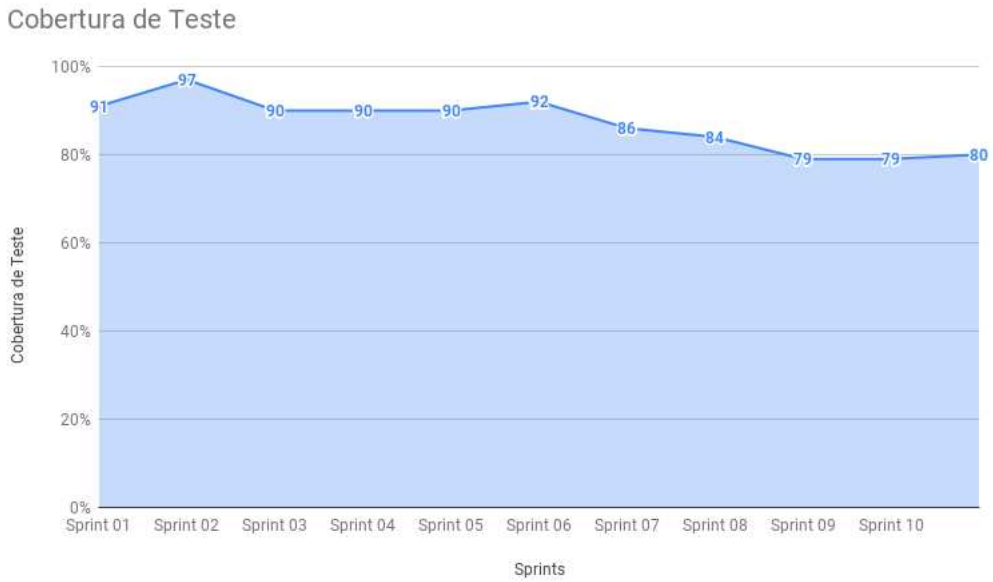


Figura 11 – Cobertura de Testes. Fonte: Autor

  - **Valor Aceitável**: Maior que 50% de cobertura de código, a medida a ser tomada é manter o nível de cobertura, e se possível aumentar o nível para que este alcance o valor ótimo.
  - **Valor Ótimo**: Maior que 90% de cobertura de código, a medida a ser tomada é manter o nível de cobertura e tentar alcançar o 100%.
  - **Valor Preocupante**: Menor que 50% de cobertura de código, a medida a ser tomada é priorizar cobertura de testes como um fator crítico na equipe, e focar todos os esforços possíveis a fim de aumentar o nível de cobertura

- **Resultados**: Cobertura de código final é de 80% ou seja um valor aceitável. Com isso podemos concluir que a qualidade do código está boa.

4.2.3 Objetivo 02: Conhecimento da equipe

4.2.3.1 Questão 01: A equipe tem conhecimento suficiente para o desenvolvimento?

4.2.3.1.1 Métrica: Quadro de conhecimento

- **Descrição**: O resultado desta métrica será dado no quadro de conhecimento, na qual o eixo x será as tecnologias empregadas no projeto e o eixo y será o nome dos integrantes da equipe, terá seis rostos que dirá se o integrante tem domínio da tecnologia, está aprendendo e consegue se virar sozinho, está tendo dificuldades na tecnologia ou se desistiu da tecnologia.

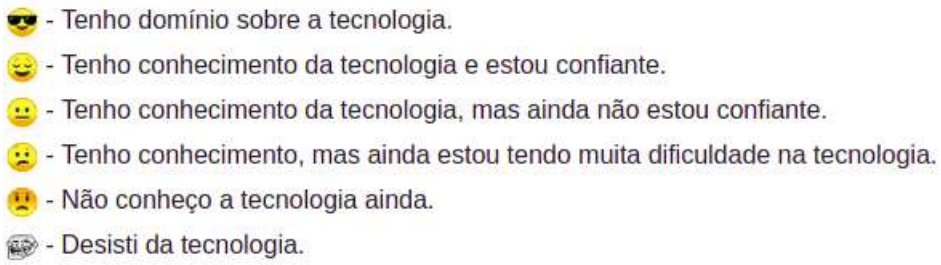


Figura 12 – Conhecimentos. Fonte: Autor

Python	Django	Pytest	Vagrant	docker	TravisCI	Bootstrap	Javascript
😎	😌	😐	😐	😐	😐	😎	😕

Figura 13 – Conhecimentos Inicio. Fonte: Autor

Python	Django	Pytest	Vagrant	docker	TravisCI	Bootstrap	Javascript
😎	😌	😌	😌	😌	😌	😎	😌

Figura 14 – Conhecimentos Final. Fonte: Autor

- **Procedimentos**: Verificar como está o quadro de conhecimento a cada sprint e identificar as tecnologias que os desenvolvedores estão tendo mais dificuldade de aprender.
- **Análise**:
  - **Valor Aceitável**: "Tenho conhecimento da tecnologia, mas ainda não estou confiante". Estudar a tecnologia sempre que houver tempo.
  - **Valor Ótimo**: "Tenho conhecimento da tecnologia e estou confiante ou tenho domínio sobre a tecnologia". Manter o domínio sobre a tecnologia.
  - **Valor Preocupante**: "Tenho dificuldade ou não tenho conhecimento sobre a tecnologia". Criar dojos e estudar a tecnologia mais a fundo com a equipe.
  - **Desistência**: Encontrar outra tecnologia para substituí-la na função que ela ia desempenhar.
- **Resultados**: Ao final do projeto o desenvolvedor teve um valor ótimo de domínio das tecnologias utilizadas, ocasionando na finalização do software.

4.3 Arquitetura do Software

O objetivo dessa seção é apresentar e detalhar a arquitetura que foi utilizada na plataforma PGTBL. O desenvolvedor que ler essa seção obterá uma visão geral de como esse software está estruturado.

4.3.1 Representação Arquitetural

O projeto será implementado utilizando o framework Django na versão 2.0. O django utiliza-se do MVT (Model-View-Template) como uma adaptação do MVC (Model-View-Controller), nas views utilizaremos as *Class Based Views* para padronização do código. Esse framework fará comunicação com o banco de dados Postgresql e o servidor nginx. A figura 15 exemplifica essa representação arquitetural.

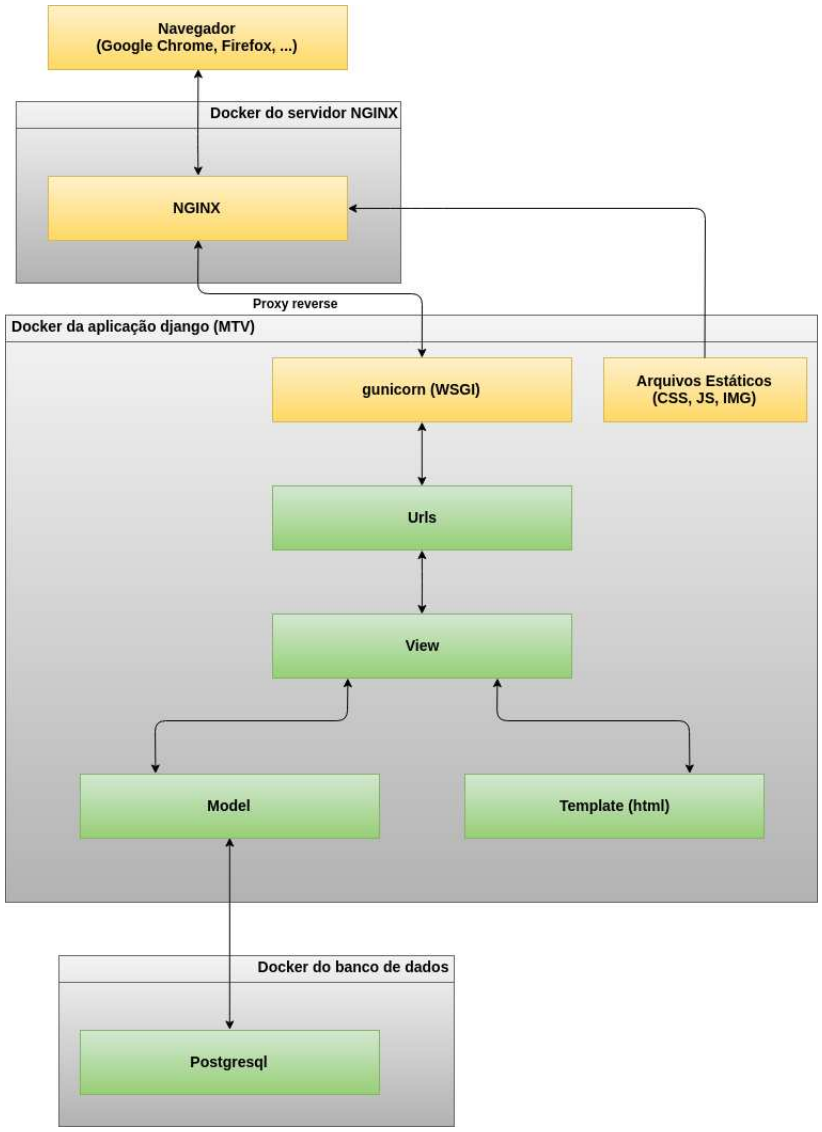


Figura 15 – Representação Arquitetural. Fonte: Autor

1. O web client (navegador) manda uma requisição para o web server (Nginx) com o protocolo HTTP.

2. Os arquivos estáticos armazenados no sistema de arquivos, como CSS, JavaScript, Imagens e documentos PDF, podem ser processados diretamente pelo web server (Nginx).

3. A parte dinâmica é delegada ao servidor de aplicativos WSGI (Web Server Gateway Interface) do django, no caso o gunicorn que é um servidor WSGI para Unix feito em python puro, ele irá converter solicitações HTTP recebidas do servidor em chamadas python em colaboração com o framework django que irá ter um arquivo chamado urls.py que diz ao nginx qual código deverá ser executado de acordo com o path e código HTTP recebido, através de proxy reverso será feito o redirecionamento inicial do Nginx com o servidor da aplicação, ou seja, o proxy reverso irá funcionar como uma ponte de ligação entre o nginx e o django através do gunicorn.

4. Dentro do django a requisição recebida pelo web server é mapeado para uma view especifica através das urls, a view pede dados a modelo, a model pega os dados do banco de dados postgresql e retorna a view, esta seleciona o template e fornece os dados, com isso o template é preenchido e devolvido a view que devolve o template como resposta ao web server.

5. O web server (nginx) retorna a resposta para o web client (navegador)

4.3.2 Pacotes Significativos do Ponto de Vista da Arquitetura

A figura 16 lista todos os pacotes do ponto de vista da arquitetura do software.

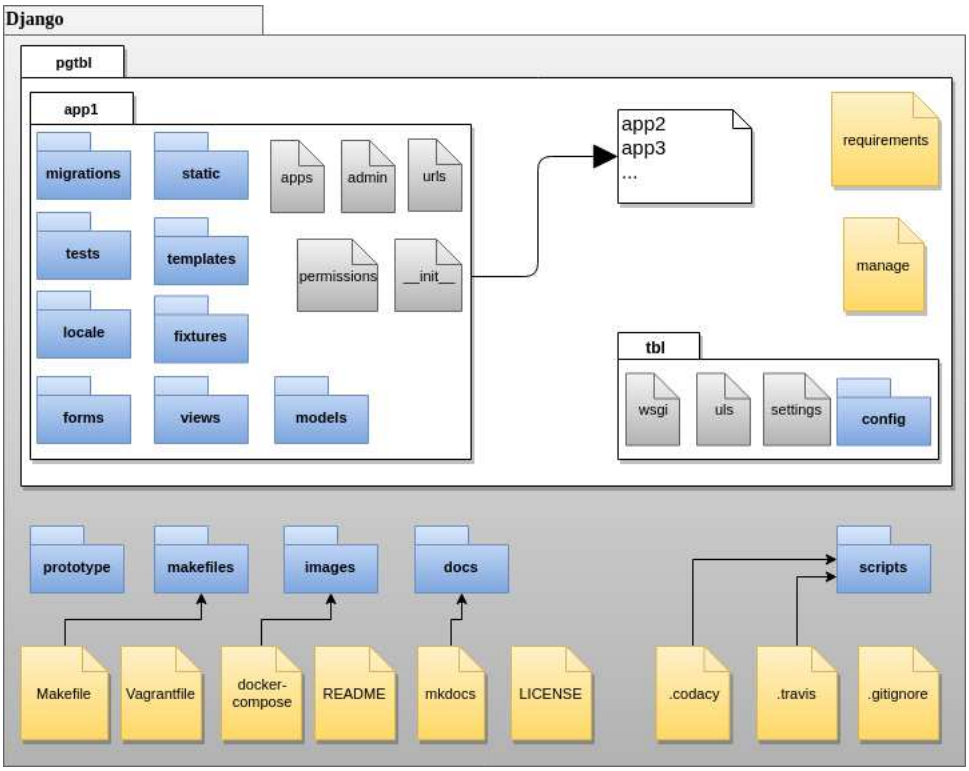


Figura 16 – Pacotes do projeto. Fonte: Autor

- Os pacotes de cada aplicação:

  - **locale**: Pasta que irá ter toda a tradução do software para pt-BR.
  - **migrations**: pasta com todas as migrações das modelos para o banco, são os SQLs.
  - **static**: É onde fica os arquivos estáticos da aplicação (CSS, JS e IMG)
  - **templates**: É onde fica os templates da aplicação (HTML)
  - **tests**: contém os testes automatizados feitos no sistema.
  - **___init___**: É o arquivo que define que sua pasta é um pacote python.
  - **admin**: contém a instância da modelo que fará parte do sistema de administração do django.
  - **app**: arquivo que contém informações da aplicação do django.
  - **forms**: pasta que contém os campos que será inserido no formularios
  - **models**: pasta de arquivos que faz interface com o banco de dados, é responsável por leitura, validação e escrita de dados no banco de dados.
  - **permissions**: Arquivo de implementação de permissões do aplicativo.
  - **urls**: São as rotas para ser acessada pelo navegador
  - **views**: pasta que contém a camada lógica do sistema e a comunicação com o navegador por meio de rotas (Classe Based Views).
  - **fixtures**: pasta que contém arquivos json para pré-popular o banco de dados para testes manuais

- Os pacotes de configuração:
  - **config**: É uma pasca que contém as configurações do software separada em arquivos.
  - **settings**: São as configurações gerais do software importadas da pasta config.
  - **urls**: Arquivo que terá o mapeamento de rotas de todo o projeto com todas as aplicações.
  - **wsgi**: Arquivo usado para deploy do projeto.

- Os pacotes do projeto Django:
  - **manage**: Arquivo de configuração geral do django.
  - **requirements**: Arquivos para instalar dependências da aplicação através do seguinte comando: pip3 install -r requirements.txt

- Os pacotes gerais do projeto PGTBL:

- **Vagrantfile**: Arquivo que gerencia a máquina virtual de desenvolvimento, criado para desenvolvedores que queiram desenvolver em sistemas operacionais diferentes do Linux, como Windows ou Mac, precisa ter o Vagrant instalado.
- **Makefile**: Arquivo de atalhos para comandos muito usados pelos desenvolvedores.
- **docker-compose**: Arquivos que gerencia todos os containers da aplicação (deploy, homolog, production, test e desenvolvimento).
- **.travis**: Arquivo que gerencia a entrega continua da aplicação através da ferramenta Travis CI no github.
- **.codacy**: Arquivo de configuração da ferramenta de análise estática de código Codacy.
- **.gitignore**: Arquivo que faz o git ignorar alguns arquivos do projeto.
- **README**: Arquivo com um conteúdo markdown inicial do projeto.
- **LICENSE**: Licença do software.
- **mkdocs**: Arquivo que contém a configuração da documentação do software.
- **prototype**: Pasta com o protótipo do software.
- **docs**: Pasta com toda a documentação do software.
- **makefiles**: Pasta que contém de forma organizada comandos do Makefile.
- **images**: Pasta que armazena as imagens de deploy da aplicação tbl.
- **scripts**: Pasta com alguns scripts de integração e deploy continuo

O repositório com o código fonte do projeto se encontra no seguinte link: <https://github.com/VictorDeon/PGTBL>

4.3.3 Diagrama de Classe

O diagrama de classe é muito grande para ser colocado no documento, logo ele se encontra na documentação no tópico diagrama de classe. Ao entrar clique com o botão direito do mouse e clique no link abrir imagem em um novo guia, com isso você consegue dar zoom na imagem.

5 Avaliação

Neste capítulo, será mostrado os resultados da aplicação do software em um contexto real, ou seja, o software foi utilizado na disciplina de Teste de Software do curso de engenharia de software na universidade de brasília campus de engenharias gama. O capítulo está dividido em seções que estão disposta da seguinte maneira: na seção 5.1 será mostrado uma pequena introdução sobre os questionários de avaliação, na seção 5.2 será listado os cenários de avaliação, na seção 5.3 será mostrado o questionário de avaliação e na seção 5.4 o resultado da avaliação.

5.1 Questionários de Usabilidade

Uma parte importante da engenharia de produtos é a medida de usabilidade. Os métodos de medição para avaliar a usabilidade não são óbvios e são um preocupação permanente dos engenheiros. A maioria das avaliações de usabilidade reúne dados quantitativos subjetivos e objetos no contexto de cenários realistas de uso. Os dados subjetivos são medidas das opiniões ou atitudes dos participantes em relação a sua percepção de usabilidade, já os objetivos são medidas do desempenho dos participantes como tempo de conclusão, taxa de sucesso e etc (LEWIS, 1993).

Como o objetivo do software é automatizar e aumentar a satisfação do uso de metodologias ativas de aprendizado, as medidas subjetivas terão maior consideração na avaliação do mesmo. As medidas subjetivas são geralmente respostas a itens do questionário do tipo "*like*"que avaliam atitudes do usuário em relação a atributos como facilidade de uso do sistema e a similaridade de interfaces em determinados cenários. A maioria dos cenários de avaliação de usabilidade são conjuntos de tarefas nas quais os usuários resolvem problemas, por exemplo, como resolver a lista de exercício, se preparando para as avaliações, automatizar o cálculo das notas entre outras em relação aos requisitos não funcionais (LEWIS, 1993).

Para adquirir essas métricas será utilizado um questionário da IBM especificamente para uso no contexto de testes de usabilidade baseados em cenários e medidas subjetivas. Foi analisado alguns tipos de questionários como o *After-Scenario Questionnaire* (ASQ), *Printer Scenario Questionnaire* (PSQ), *Post-Study System Usability Questionnaire* (PSSUQ) e *Computer System Usability Questionnaire* (CSUQ). Como as metodologias ativas de aprendizado tem alguns cenários que precisam ser avaliados separadamente, foi definido para a avaliação de satisfação dos usuários o questionário ASQ, pois ele é um questionário curto e consegue adquirir métricas para cada cenário do software. Apesar de o ASQ e o PSQ terem os mesmo itens mudando apenas a escala de avaliação, o ASQ tem uma

confiabilidade melhor de acordo com (LEWIS, 1993). Já o PSSUQ e o CSUQ são ambos questionários gerais de satisfação, ou seja, são questionários mais extensos que englobam todo o sistema, que não é o ideal para a avaliação, já que não conseguiria coletar feedbacks de cada cenário para possíveis melhorias.

5.2 Cenários de avaliação

Abaixo será listado os cenários de avaliação na qual será aplicado o questionário ASQ em comparação ao mesmo cenário sem utilizar o softwate PGTBL.

- avaliação iRAT
- avaliação gRAT

Os 2 itens são questionários para o estudante responder depois de cada cenário.

5.3 Questionário ASQ

O questionário Pós-cenário (ASQ) é um questionário de três ou mais itens que os avaliadores de usabilidade da IBM utilizam para avaliar a satisfação do participante após a conclusão de cada cenário proposto (LEWIS, 1993). Os itens do questionário abordam afirmações relacionadas a 4 dos 5 requisitos não funcionais aplicados a ferramenta para avaliar se os requisitos foram satisfeitos. Os requisitos são: suportabilidade, desempenho, usabilidade e segurança ou confiabilidade. O quinto requisito não funcional, o requisito de qualidade, é respondido pelas métricas coletadas na ferramenta codacy.

O questionário será constituido de itens com escalas gráficas de 7 pontos, na qual temos 7 para o caso do usuário concordar totalmente e 1 caso discordar totalmente e um ponto não aplicavel (N/A) fora da escala para o caso de o usuário não querer responder. O resultado será a média aritmética dos escores dos quatro itens para obter a pontuação do ASQ para a satisfação de um participante com o sistema para um determinado cenário. Quanto mais alto o resultado melhor. Se um participante não responder a um item ou marcar N/A, o cálculo da média será a média dos itens respondidos.

$$P_i = \frac{x_1 + x_2 + x_3 + x_4}{4}$$

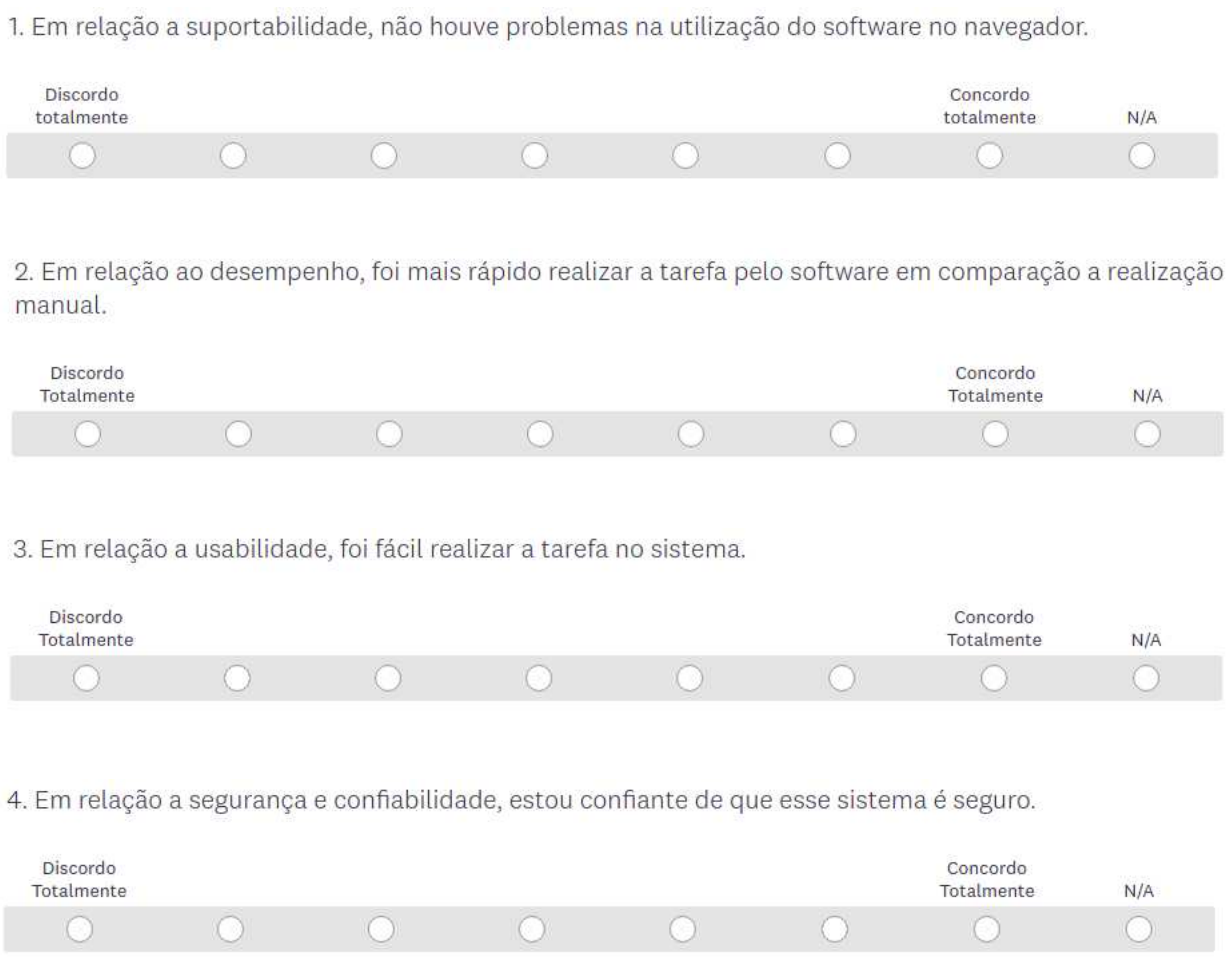
$$R_{req} = \sum_{i=1}^{n} \frac{y_i}{n}$$

$$R_{final} = \sum_{i=1}^{n} \frac{P_i}{n}$$

P_i é a pontuação final de cada aluno no questionário, x é a pontuação de cada item do questionário, y é a nota do aluno relacionada ao requisito não funcional especificado, R_{req} é a média dos pontos para um determinado requisito não funcional, R_{final} é o resultado final da avaliação de usabilidade, i é o indice de cada aluno e n é a quantidade total de alunos. A média padrão do questionário é 4.0, se o resultado da média relacionada ao requisito não funcional tiver abaixo dessa média padrão, o requisito não foi satisfeito, e se o resultado final de um determinado cenário tiver abaixo da média padrão, quer dizer que o cénario não foi bem aceito.

Questionário para cada cenário:

Em relação ao software PGTBL para cada uma das afirmações abaixo, preencha a classificação de sua escolha.



5.4 Resultado

Nesta seção será apresentado os resultados do estudo de cado para os dois cenários especificados: iRAT e gRAT. Os outros cenários como Avalição prática e avaliação em pares não foi possível de ser feito por limitações no servidor de produção.

5.4.1 Objetivo

Normalmente os alunos tem um pouco de dificuldade em aceitar o novo e sair do comodismo, que são as avaliações no papel, como foi explicado no referência teórico do trabalho, isso pode dificultar um pouco a aceitação da ferramenta, já que, por ser algo novo e está sendo aplicado a uma disciplina em que a menção é importante para a continuação do curso por parte do aluno. A ferramenta pode trazer um pouco de desconforto

O objetivo dessa avaliação é verificar o indice de usabilidade para cada um dos requisitos não funcionais dentro do contexto dos cenários propostos.

5.4.2 Roteiro

No dia 03/06/2019 será realizado a avaliação iRAT e gRAT do TBL2 da disciplina de teste de software na universidade de brasília utilizando a ferramenta.

O questionários será realizado da seguinte maneira:

1. Antes da avaliação os alunos serão cadastrados na ferramenta e inseridos na disciplina.
2. Com os alunos cadastrados, o professor irá montar os grupos na ferramenta e disponibiliza-los para os alunos.
3. No dia 03/06/2019 às 14:30 a avaliação iRAT será disponibilizada e terá duração de 30 minutos.
4. No mesmo dia de 15:00 às 15:10 será o tempo para os alunos responderem o questionário voltado ao iRAT
5. No mesmo dia de 15:20 às 15:40 será liberado a avaliação gRAT em que apenas um aluno junto com seu grupo irá submeter.
6. No mesmo dia de 15:40 às 16:00 será o tempo para os alunos responderem o questionário voltado ao gRAT e visualizar sua nota na ferramenta.
7. Às 16:00 se encerra a avaliação e é recolhido os questionários e gerado o relatório na ferramenta para o professor.

5.4.3 Resultados Esperados

Na questão sobre portabilidade, pode-se esperar que o resultado seja positivo, entre 5 e 7, já que a ferramenta é voltada para o navegado que é suportado em qualquer sistema operacional, em qualquer computador. Na segunda questão voltada ao desempenho para a realização da tarefa, aqui terá convergencias e divergencias já que o maior foco da

ferramenta no quesito de agilidade é para o professor no cálculo das notas, e na economia de recursos.

Na terceira questão voltado a usabilidade da ferramente, espera-se um valor alto entre 5 e 7, já que a ferramenta foi desenhada para ser fácil de usar, padronizado e com um design atraente. Na última questão voltado a segurança e confiabilidade da ferramenta, não se espera um resultado muito posítivo já que os alunos normalmente tem um pouco de dificuldade de aceitar o novo, principalmente quando se trata de avaliação.

5.4.4 Resultados Obtidos iRAT

Figura 17 – iRAT Suportabilidade. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 6.54, como citado anteriormente já era esperado esse tipo de resultado.

Figura 18 – iRAT Desempenho. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.79, como citado anteriormente o resultado do requisito de desempenho iria oscilar bastante no gráfico, e o resultado ficou um pouco acima da média padrão.

Figura 19 – iRAT Usabilidade. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.95, erá esperado um resultado maior para esse requisito não funcional, porém ainda está acima da média padrão.

Figura 20 – iRAT Segurança e Confiabilidade. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.72, erá esperado um resultado menor, porém o resultado foi acima de média padrão.

Figura 21 – iRAT Resultado Final. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 5.25, ou seja, o iRAT teve uma avaliação positiva em relação os requisitos não funcionais especificados.

5.4.5 Resultados Obtidos gRAT

Figura 22 – gRAT Suportabilidade. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 6.33, assim como o iRAT já era esperado esse tipo de resultado.

Figura 23 – gRAT Desempenho. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.49, assim como o iRAT o resultado do requisito de desempenho iria oscilar bastante no gráfico, e o resultado ficou um pouco acima da média padrão.

Figura 24 – gRAT Usabilidade. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.38, assim como o iRAT também erá esperado um resultado maior para esse requisito não funcional, porém ainda está acima da média padrão.

Figura 25 – gRAT Segurança e Confiabilidade. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.64, assim como no iRAT erá esperado um resultado menor, porém o resultado foi acima da média padrão.

Figura 26 – gRAT Resultado Final. Fonte: Autor

Análise: A média obtida desse requisito não funcional foi de 4.96, ou seja, o gRAT teve uma avaliação positiva em relação os requisitos não funcionais especificados, porém um pouco menor do que a iRAT.

5.4.6 Ponto de vista dos usuários

O professor Ricardo Ajax, que utilizou a ferramenta em sua disciplina de teste de software, citou algumas considerações sobre a experiência que ele teve usando a ferramenta:

Esta é uma parte finalzinha, após o relato de termos usado o PGTBL em sala de aula para executar o TBL 2. Vamos a ele.

O sistema foi usado para administrar toda uma seção completa de TBL e os resultado obtidos serviram como validação do uso do sistema junto aos seus usuários, além de coletar oportunidades de melhoria.

Inicialmente foi feita uma seção onde todos os estudantes da turma se cadastraram no sistema. Para isso já tinha me cadastrado previamente, assim como criado a disciplina e a sessão de TBL a ser usada nos testes de validação do sistema. Os alunos foram acompanhados pelo desenvolvedor durante os seus cadastramentos, a fim de sanar alguma dúvida que ocorresse no cadastramento dos alunos, ou mesmo algum erro de sistema. Esta atividade ocupou certa de 30 a 40 minutos de uma das aulas da disciplina, cujo total de tempo é de 100 minutos. Ou seja, foram ocupados cerca de 30 a 40% de uma das aulas da disciplina, considerado um tempo bastante adequado para uma turma de 50 alunos. Posteriormente agrupamos os estudantes, preparando o PGTBL para ser usado na sessão onde o questionário é respondido em grupo.

Dando continuidade aos testes de validação do sistema, inseri as questões da sessão de TBL a ser executado em sala de aula. Na inserção das questões, é necessário informar qual a alternativa correta de cada uma para que o sistema proceda com as devidas correções e atribuições de notas das várias fases do TBL. Esta atividade ocorreu sem maiores problemas. Algumas dúvidas pontuais sobre o uso das funcionalidades do sistema envolvidas com essa tarefa foram tiradas prontamente pelo desenvolvedor. Com as questões prontas o tempo necessário para o cadastro delas no PGTBL foi considerado adequado (cerca de 1,5 a 2 minutos para cada questão) e não exitiram intercorrências na atividade . Ate esse momento todas as informações necessárias foram cadastradas no PGTBL, deixando-o pronto para a realização do TBL junto aos alunos.

A sessão de TBL é realizada conforme cronograma de aulas da disciplina. Os estudantes devem estar em sala de aula com antecedência mínima de 15 minutos para que todas as máquinas sejam ligadas, o SISTEMA carregado e cada estudante esteja com o seu logon efetuado. Nos testes de validação esse tempo foi usado e a sessão de TBL só teve início quando todos os estudantes assinalaram estar logados ao sistema. Neste momento foi iniciado a sessão onde os alunos responderão individualmente o questionário (iRAT). A sessão tem tempo marcado que, após transcorrido o sistema encerra automaticamente a fase de respostas individuais.

Após a sessão individual inicia-se a sessão onde os alunos respondem em grupo o questionário (gRAT). A mesma dinâmica de início e prazos marcados é usada, mas os alunos podem se dispor fisicamente conforme for melhor para eles no ambiente da sala de aula, mas é necessário apenas um dos componentes do grupo marcar as respostas pelo grupo, não permitindo outros membros do grupo responder a mesma questão, uma vez respondida por um membro já é computado a resposta. E encerrada a sessão, o PGTBL não permite mais marcar respostas ou atualizar as já marcadas.

A fase prática é realizada em uma próxima aula, devido ao tempo que ela leva

> para ser executada. O problema prático é publicado e os estudantes passam a realiza-lo. Não dando tempo para completar a realização eles têm um prazo para entrega. A entrega do relatório da fase prática foi feito pelo Moodle, devido a restrições do ambiente onde o PGTBL foi instalado sobre o upload de arquivos.
>
> A grande vantagem do PGTBL está em finalizada cada sessão do TBL já ser possível emitir relatórios sobre as notas de cada estudante e de seus respectivos grupos para iRAT e gRAT. Essa atividade era realizada manualmente com a digitação de todos os gabaritos de todos os alunos em uma planilha Excel que calculava a nota de cada um. Esse tempo de digitação foi economizado, pois representava uma jornada de trabalho de pelo menos 2 horas para uma turma de 50 alunos. Além disso, outras vantagens do uso do PGTBL são: a funcionalidade que exporta as notas em uma planilha no formato CSV que pode ser importada pelo professor em outras ferramentas, o traçado de gráficos com as questões e seus níveis de acertos. Tais gráfico podem ser usados pelo professor para identificar problemas de compreensão do conteúdo teórico, apoiando-o na elaboração da aula de fechamento do TBL onde esses conteúdos são reforçados.
>
> Com esses testes o sistema PGTBL foi considerado validado em seu uso, conforme as regras determinadas pela teoria de TBLs e por uma aplicação prática na turma. O questionário de usabilidade aplicado, detalha melhor a opinião dos alunos sobre o uso do PGTBL.
>
> O uso do TBL resultou em algumas oportunidades de melhoria a título de trabalhos futuros, como: Criar uma funcionalidade que importe os dados dos alunos da turma da disciplina (ex.: matricula, nome, e grupo a que pertence). Isso evitaria a sessão de cadastramento dos alunos e a necessidade de agrupa-los em seus respectivos grupos de trabalho; Construir uma área onde questões possam ser armazenadas para uso posterior em outros TBLs. As questões cadastradas além de poderem ser usadas em outras sessões de TBL poderiam ser somente atualizadas, sem a necessidade de serem novamente incluídas.
>
> Uma oportunidade de melhoria já implementada foi modificar o relatório de resultados das sessões de iRAT e gRAT para ser emitido não com o nome de login do aluno, mas sim pelo seu nome, o que simplifica depois que as notas sejam importadas para outras ferramentas e usadas como parte do cálculo da menção final da disciplina. (KOSLOSKI, 2019)

Alguns alunos também citaram alguns pontos que devem ser melhorados na ferramenta para trabalhos futuros, são eles:

- Calcular as notas na hora que o aluno finalizar a avaliação, independente de a avaliação estiver rodando ou não.
- Aumentar o tamanho da raspadinha.
- Mostrar a quantidade total de pontos que a equipe colocou em cada alternativa do gRAT. Armazenar os pontos que os alunos do grupo fizeram no iRAT para usar

como base no gRAT.

- Ao inves dos botões de próximo e anterior, colocar uma paginação para navegar nas questões. E nessa paginação mostrar quais foram e não foram respondidas ainda.
- No momento que responder uma questão, ser direcionado para a próxima questão.
- Poder visualizar e alterar as respostas até o final da avaliação.
- Calcular e aplicar a questão sómente ao final da prova, ao concluir a prova, não em cada questão respondida.
- Deixar mais explícito a navegabilidade da aplicação, deixando mais visível os itens mais importantes.

6 Considerações finais

Este trabalho abordou uma metodologia que vem sendo bastante empregada em várias instituições de ensino para preparar melhor os alunos para o mercado de trabalho chamada *Team Based Learning*. Entretanto, as ferramentas encontradas para apoio à sua aplicação não cobriam determinadas áreas críticas necessárias à aplicação do TBL como: apostas em avaliaçôes, o calculo das notas derivadas dessas apostas, feedbacks em tempo real e etc. Elas apenas ajudam a implantação da mesma.

Com o intuito de fechar essa lacuna em relação a ferramenta foi proposto um software chamado PGTBL que irá automatizar todo o processo da metodologia para que o professor possa apenas focar no conteúdo e facilitar o gerenciamento da turma, não tendo que gastar recursos para poder implantar a metodologia em sala de aula.

O *software* foi pensado sempre com foco na qualidade do mesmo, e em aspectos de engenharia de software como engenharia de requisitos, UX, devops, métodos ágeis, medição e análise, testes automatizados, integração e entrega contínua, boas práticas de programação entre outros.

Em relação aos objetivos do TCC, todos foram concluidos com sucesso, desde o estudo do contexto do TBL, elicitação dos requisitos do software, elaboração do processo de desenvolvimento, execução e aplicação do software em uma disciplina da UnB aplicando o questionário de satisfação em uso para coletar feedbacks sobre o software.

Os alunos gostaram bastante da ferramenta, apesar da dificuldade de aceitação no início, já que é algo relacionado a avaliação e menção deles na disciplina, os resultados do questionário de usabilidade foram bem satisfatórios e tornaram possível a avaliação dos requisitos não funcionais da ferramenta, como: desempenho, suportabilidade, usabilidade, segurança e confiabilidade. O professor também gostou bastante do resultado, principalmente na facilidade do cálculo das notas e feedbacks de questões que geraram mais dificuldades nos alunos.

Foi proposto um escopo bem fechado em relação a ferramenta nessas duas entregas planejadas, porém, para trabalhos futuros, aplicação poderá ter um sistema de *machine learning*, a fase de preparação terá ebooks dinâmicos, e um sistema de gamificação além de ser extensível a qualquer disciplina ou curso tanto superior como ensino médio. Também será modificado a arquitetura da aplicação para que ela possa ser extensivel para aplicativos mobile e desktop, tendo como foco avaliações offline, sem a dependência de internet.

Referências

ALHOMOD, S. M.; SHAFI, M. M. Facebook as a tool to enhance team based learning. *International Journal of Computer Science*, 2013. Citado na página 15.

AWATRAMINI, M.; ROVER, D. Team-based learning course design and assessment in computer engineering. In: *In Frontiers in Education Conference (FIE) - IEEE*. [S.l.: s.n.], 2015. Citado 3 vezes nas páginas 15, 23 e 24.

BARBOSA, E. F.; MOURA, D. G. de. *Metodologias ativas de aprendizagem na educação profissional e tecnológica*. [S.l.], 2013. 48–67 p. Boletim Técnico do Senac 39. Citado na página 15.

BERGE, Z.; COLLINS, M. Computer conferencing and online education. In: . [S.l.: s.n.], 1993. Citado na página 15.

BOLLELA, V. R. et al. Aprendizagem baseada em equipes: da teoria à prática. Ribeirão Preto, 2014. Medicina. Citado 5 vezes nas páginas 18, 19, 20, 22 e 23.

BURGESS, A. W.; MCGREGOR, D. M.; MELLIS, C. M. Applying established guidelines to team-based learning programs in medical schools: A systematic review. *Academic Medicine*, 2014. Disponível em: <https://doi.org/10.1097/ACM.0000000000000162>. Citado na página 18.

CABRERA, I.; VILLALON, J.; CHAVEZ, J. Blending communities and team-based learning in a programming course. *IEEE Transactions on Education*, 2017. Citado 3 vezes nas páginas 15, 18 e 24.

DAVIS, C. E. et al. Innovative practices for engineering professional development courses. In: *In Frontiers in Education Conference, IEEE*. [s.n.], 2013. Disponível em: <http://ieeexplore.ieee.org/abstract/document/6684898/>. Citado 3 vezes nas páginas 14, 23 e 24.

ELNAGAR, A.; ALI, M. S. A modified team-based learning methodology for effective delivery of an introductory programming course. In: *In Proceedings of the 13th annual conference on Information technology education*. [S.l.: s.n.], 2012. Citado na página 23.

GOMEZ, E. A.; WU, D.; PASSERINI, K. Computer-supported team-based learning: The impact of motivation, enjoyment and team contributions on learning outcomes. *Computers and Education*, 2010. Citado 3 vezes nas páginas 15, 18 e 19.

KAM, H.-J.; KATERATTANAKUL, P. Structural model of team-based learning using web 2.0 collaborative software. *Computers and Education*, 2014. Citado na página 15.

KOSLOSKI, R. A. D. Considerações sobre o uso do pgtbl. In: . [S.l.: s.n.], 2019. Citado na página 56.

LEWIS, J. R. *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instruction for User*. Boca Raton, Florida: [s.n.], 1993. Citado 2 vezes nas páginas 46 e 47.

MATALONGA, S.; MOUSQUES, G.; BIA, A. Deploying team-based learning at undergraduate software engineering courses. *IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM)*, 2017. Citado 3 vezes nas páginas 18, 23 e 24.

MICHAELSEN, L. K.; SWEET, M. The essential elements of team-based learning. *New Directions for Teaching and Learning 2008*, 2008. Citado 5 vezes nas páginas 18, 19, 20, 21 e 22.

MICHAELSEN, L. K.; SWEET, M.; PARMELEE, D. Team-based learning: Small group learning's next big step. *New Directions for Teaching and Learning*, San Francisco, CA, 2008. Citado na página 18.

MORAN, J. Mudando a educação com metodologias ativas. In: . [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 14 e 15.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. [S.l.: s.n.], 1997. Citado 2 vezes nas páginas 27 e 30.

RAMOS, C. S. et al. Tbl as an active learning-teaching methodology for software engineering courses. *In XXXII Brazilian Symposium on software engineering (SBES 2018)*, São Carlos, 2018. Disponível em: <https://dl.acm.org/citation.cfm?id=3266253>. Citado 2 vezes nas páginas 23 e 24.

SAFE: Scaled Agile Framework. 2018. Acessado em: 28 de Abril de 2018. Disponível em: <https://www.scaledagileframework.com/>. Citado na página 27.

Apêndices

APÊNDICE A – EAP

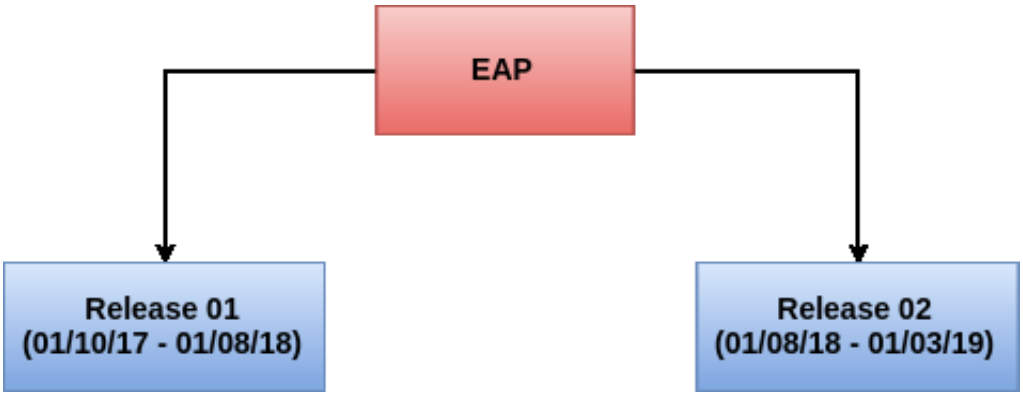


Figura 27 – EAP do produto. Fonte: Autor

A.1 Release 01

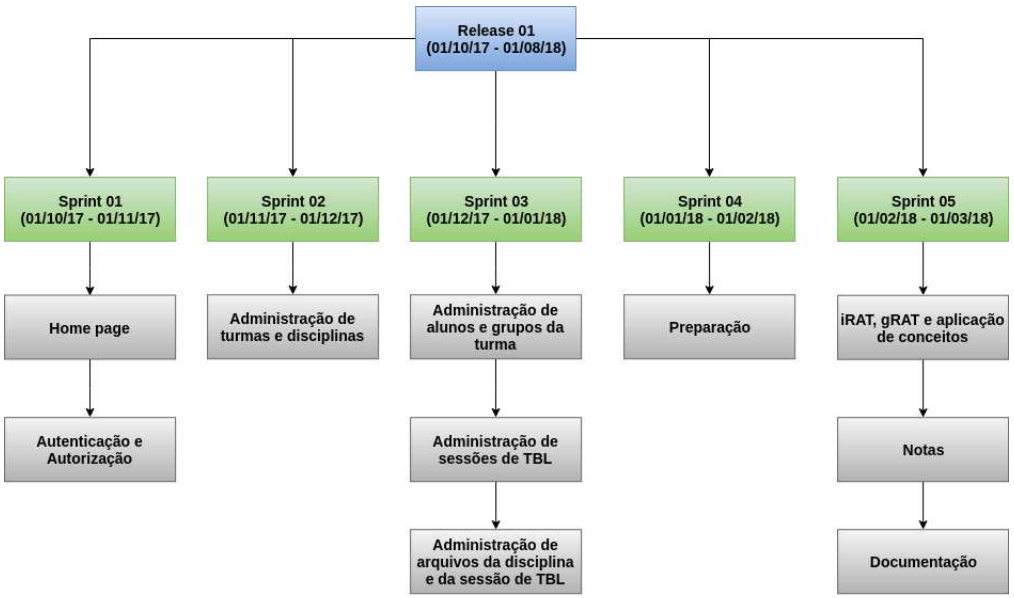


Figura 28 – EAP do produto, release 01. Fonte: Autor

A.2 Release 02

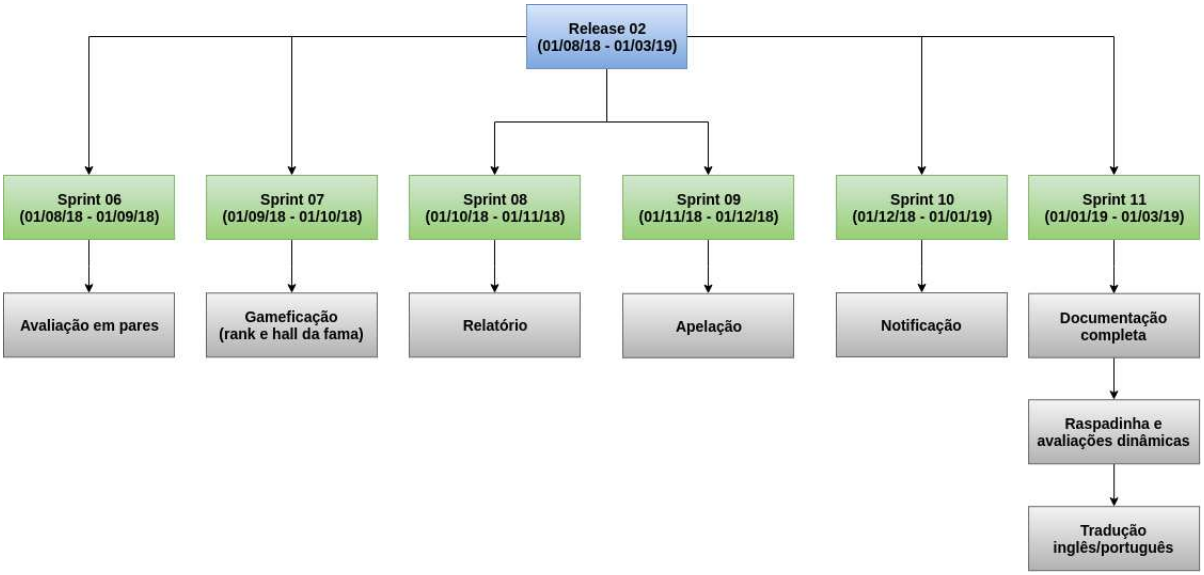


Figura 29 – EAP do produto, release 02 Fonte: Autor

APÊNDICE B – Fotos PGTBL

B.1 Perfil do usuário

Figura 30 – Perfil do usuário. Fonte: Autor

B.2 Criar ou atualizar disciplina (Markdown)

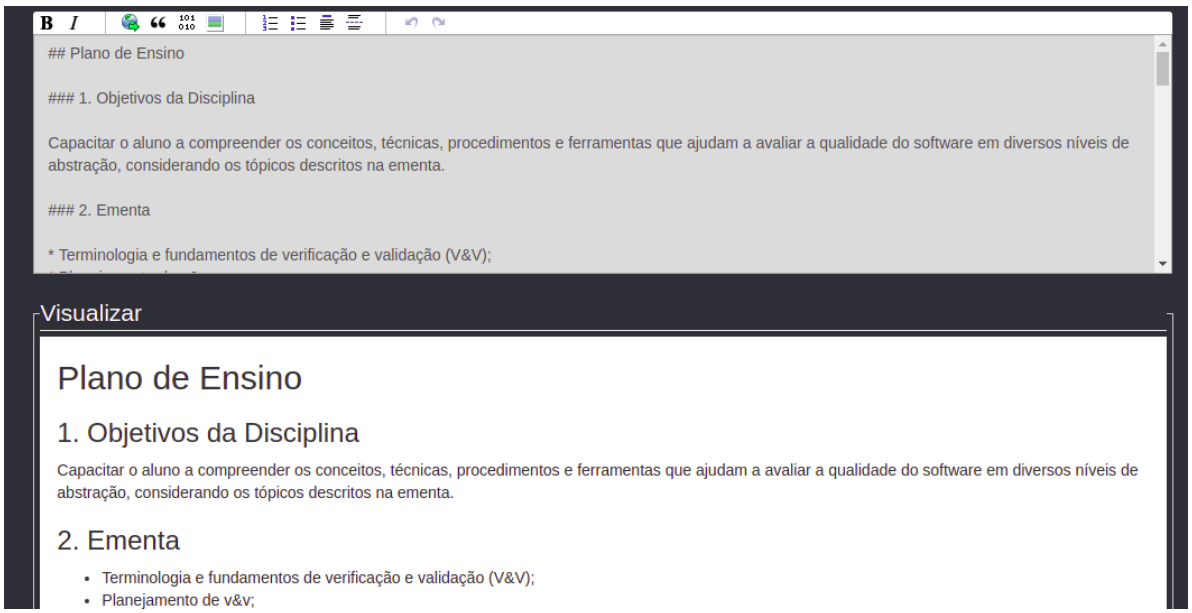


Figura 31 – Criar ou atualizar disciplina. Fonte: Autor

B.3 Rank de grupos

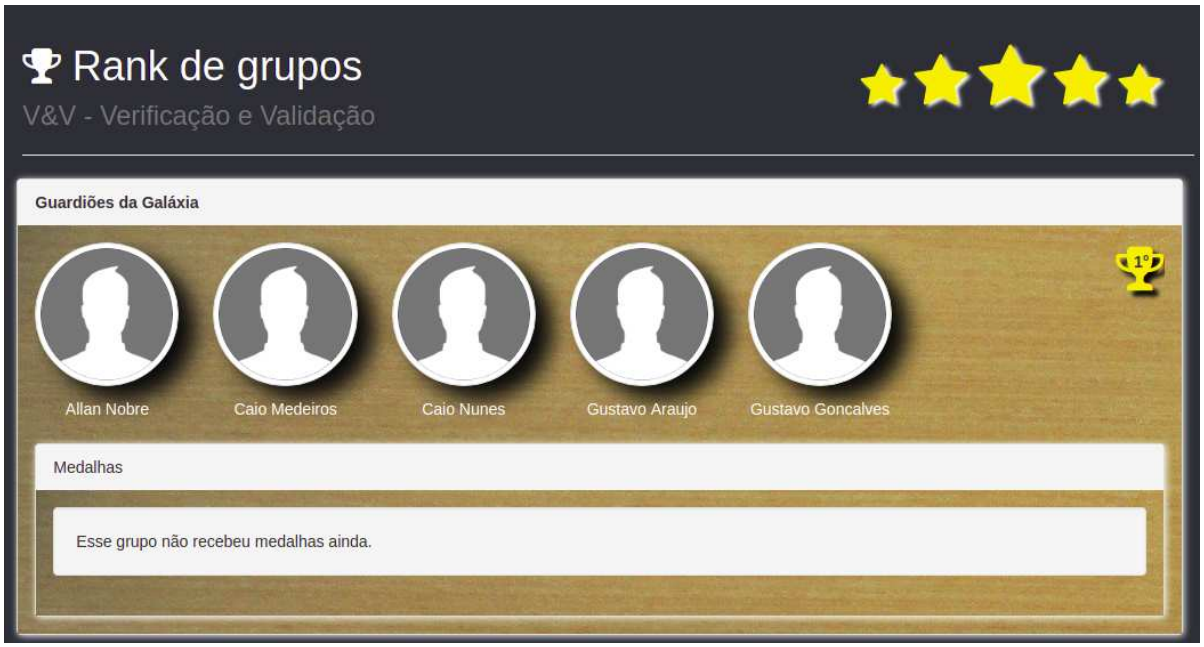


Figura 32 – Rank de grupos. Fonte: Autor

B.4 Dashboard

Figura 33 – Dashboard. Fonte: Autor

B.5 Avaliação iRAT

Figura 34 – Avaliação iRAT. Fonte: Autor

B.6 Avaliação gRAT

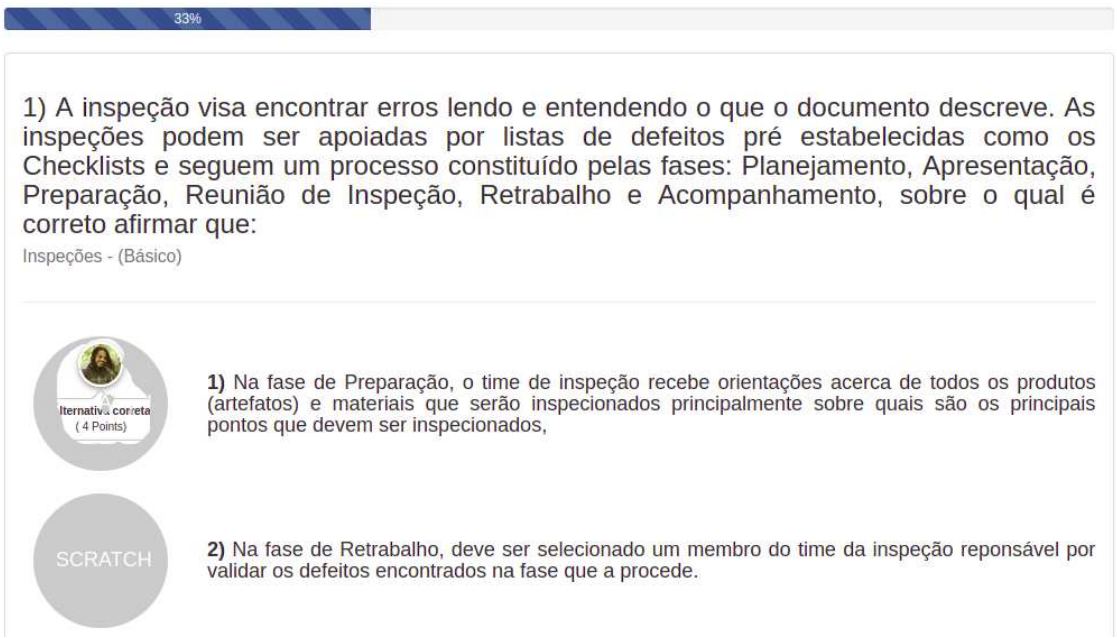


Figura 35 – Avaliação gRAT. Fonte: Autor

B.7 Avaliação prática

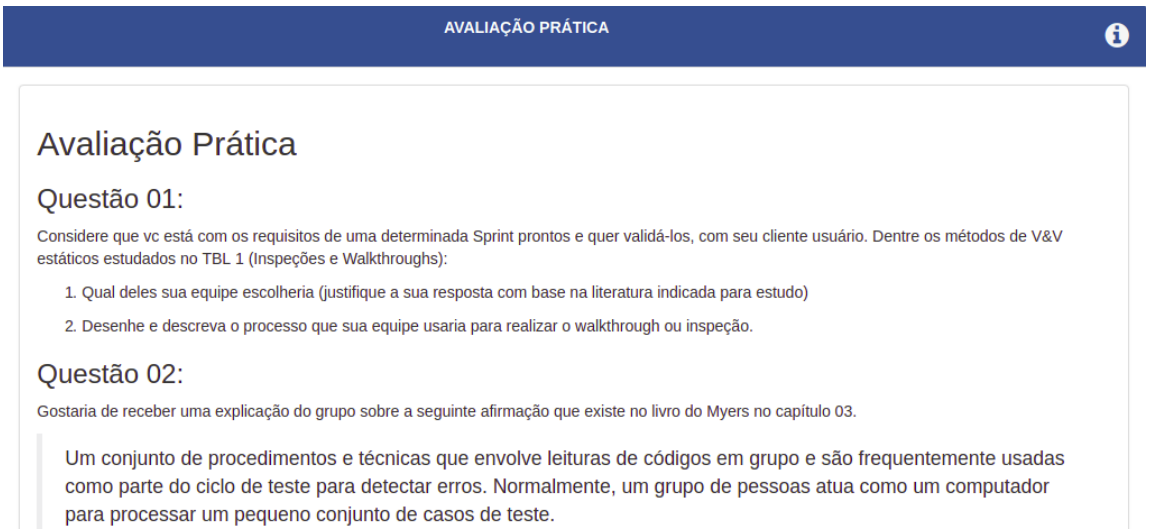


Figura 36 – Avaliação prática. Fonte: Autor

B.8 Avaliação em pares

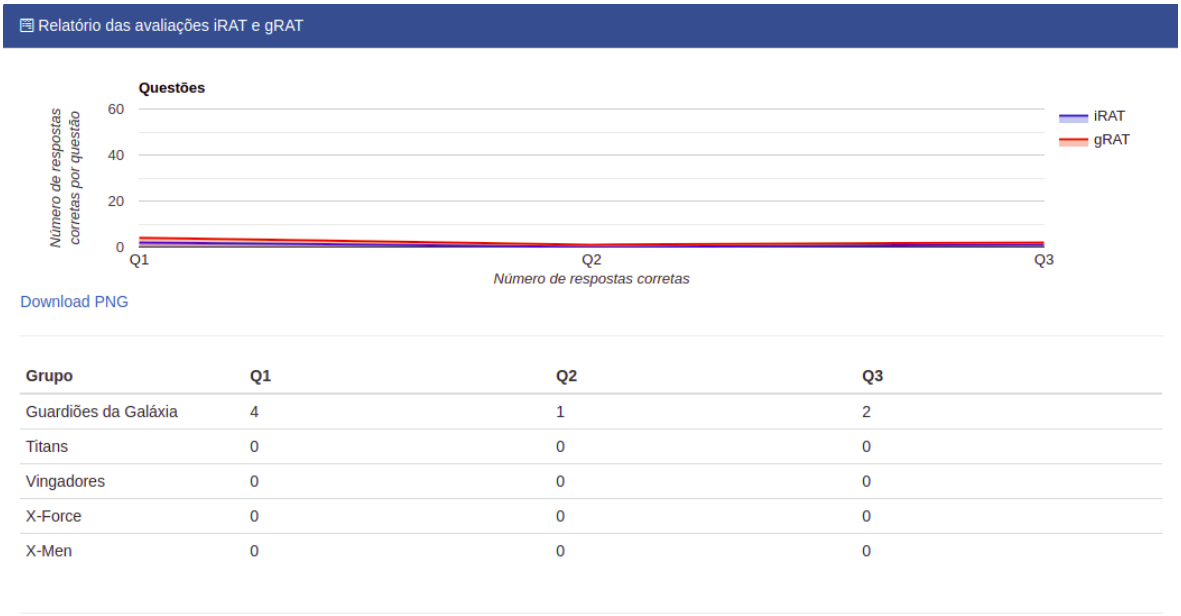
Figura 37 – Avaliação em pares. Fonte: Autor

B.9 Notas

Estudantes		Sessões				
Grupo	Usuário	iRAT	gRAT	Avaliação prática	Avaliação em pares	Nota TBL
Guardiões da Galáxia	15/0029624	2,50	5,83	0,00	0,00	1,92

Figura 38 – Notas da sessão de TBL. Fonte: Autor

B.10 Relatórios



Grupo	Q1	Q2	Q3
Guardiões da Galáxia	4	1	2
Titans	0	0	0
Vingadores	0	0	0
X-Force	0	0	0
X-Men	0	0	0

Figura 39 – Relatórios do professor. Fonte: Autor

B.11 Apelação

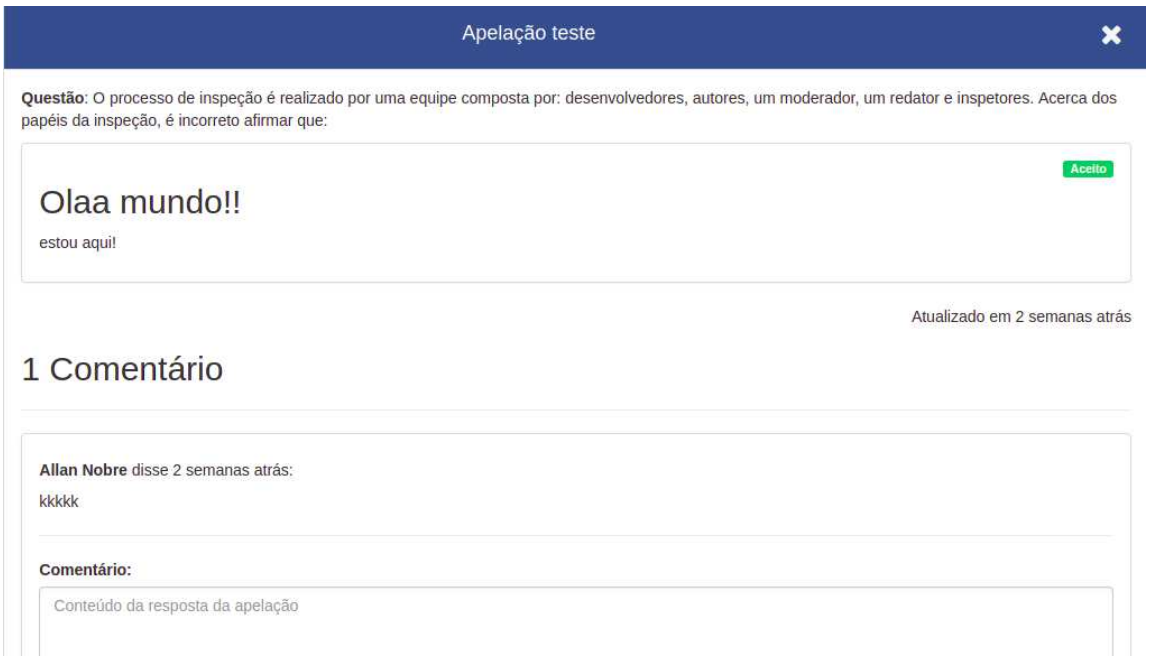


Figura 40 – Apelações. Fonte: Autor