UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LEANDRO C. LEITE
RENAN H. BARBIERI

Motiva: uma ferramenta para incentivar o aprendizado fora de aula

RIO DE JANEIRO
2019

LEANDRO C. LEITE
RENAN H. BARBIERI

Motiva: uma ferramenta para incentivar o aprendizado fora de aula

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Vinícius Gusmão Pereira de Sá

RIO DE JANEIRO
2019

Leite, Leandro Carneiro
L533m Motiva: uma ferramenta para incentivar o aprendizado fora de aula / Leandro Carneiro Leite, Renan Hozumi Barbieri. – 2019.

50 f.

Orientador: Vinícius Gusmão Pereira de Sá.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Sistema. 2. Colaboração. 3. Comunicação. 4. Tempo real. 5. Estudo. 6. Motivação. I. Barbieri, Renan Hozumi. II. Sá, Vinícius Gusmão Pereira (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Matemática. IV. Título.

Motiva: uma ferramenta para incentivar o aprendizado fora de aula

Leandro Carneiro Leite

Renan Hozumi Barbieri

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Leandro Carneiro Leite

Renan Hozumi Barbieri

Aprovado por:

Prof. Vinícius Gusmão Pereira de Sá, D.Sc.

Profª. Maria Luiza Machado Campos, D.Sc.

Profª. Valeria Menezes Bastos, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

Setembro de 2019

RESUMO

Este trabalho é um sistema – uma rede social – onde um usuário pode escrever, editar, gerenciar e publicar conteúdo para que os demais usuários possam interagir lendo e comentando. Criado sob a investigação de pontos de melhoria no curso de Bacharelado em Ciências da Computação da Universidade Federal do Rio de Janeiro e de acordo com técnicas de gerência de desenvolvimento e de validação de ideias, seu objetivo principal é motivar o compartilhamento de conhecimento dentro do mesmo com a finalidade de ampliar a capacidade de todos, utilizando ferramentas motivacionais como gamificação e facilitadores como a comunicação em tempo real. Com esta publicação, espera-se que os interessados possam compreender melhor a ideia e implementar a proposta como uma nova forma de comunicação e evolução para discentes e também para docentes.

Palavras-chave: sistema. colaboração. comunicação. tempo real. estudo. motivação.

ABSTRACT

Motiva is a system – as a social network – where a user can write, edit, manage, and publish content so the other users can interact reading and commenting on those publications. Created with a study about improvement points we could apply on the Computer Science Bachelor's Degree of the University of Rio de Janeiro and accordingly to management techniques of software development and hypotesis validation, its main goal is to use motivational tools like gamification and facilitators like real time communication to motivate the knowledgement sharing so everyone who's on the system can improve their skills. It's expected that with this publishing the interested ones can understand better the idea of the system and be able to implement the solution as a new communication and evolution way, both for students and professors.

Keywords: system. colaboration. communication. realtime. study. motivation.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Durante o curso de uma disciplina optativa chamada Tópicos Especiais em Programação, várias discussões foram geradas em sala de aula, diversos tópicos interessantes foram abordados, inúmeras particularidades de algoritmos foram apresentadas. Aquele tipo de conteúdo e aula poderia motivar qualquer aluno que não estivesse no auge da sua graduação. O inconveniente, não apenas nessa disciplina, é que todo aquele ambiente era restrito à sala de aula. Nenhuma discussão era gerada durante qualquer tempo extra classe, pouco material de estudo era exposto por conta do estilo da aula. Seria muito interessante, não só para os alunos inscritos, também para os possíveis futuros ingressantes nas disciplinas, ter contato com o maior volume possível de conteúdo gerado nas salas de aula.

Muitas vezes o próprio professor não tem disponibilidade para gerar conteúdo e prefere, com razão, se dedicar à excelência na hora de ensinar um determinado assunto. A sala de aula é um ambiente colaborativo, e esse comportamento deveria ser estendido aos momentos pós classe, afinal os encontros presenciais são apenas uma pequena parte do aprendizado. Nesse caso, os alunos também são responsáveis por entender e gerar materiais de estudo, e os que o fazem, sempre são lembrados e viram uma referência em sala de aula.

Mesmo que a motivação seja um fator pessoal, uma comunidade de estudantes pode influenciar positivamente nesse aspecto. É muito comum ver o aumento de desempenho de um aluno, ou até mesmo sua recuperação durante um período letivo, quando se junta a outros para estudar. Essa prática é constantemente influenciada pelos professores, inclusive utilizando canais oficiais de comunicação das disciplinas, mas raramente é executada de forma eficiente e duradoura, principalmente por conta da escolha da ferramenta de gerenciamento de conteúdo e comunicação.

Dentro do cenário apresentado e com uma certa semelhança às salas de aulas e seus momentos cooperativos, a proposta é apresentar o Motiva, um sistema colaborativo e organizado que facilita a troca de conhecimento e gerenciamento de conteúdo, podendo ser utilizado por professores e alunos independentemente de disciplinas, períodos, temas, experiência, horários disponíveis, entre outros. Os objetivos principais desse modelo proposto são motivar os alunos e melhorar não só a interação no ambiente acadêmico, mas também o seu nível por conta do desempenho dos discentes e da colaboração extra classe dos docentes.

1.1 CONTRIBUIÇÃO

Este projeto visa contribuir com a solução dos seguintes problemas:

- Falta de discussões de estudos extra classe

O tempo em sala de aula é limitado e muitas das vezes alguns bons debates são interrompidos para que o fluxo de ensino siga em frente. As discussões que ocorrem – quando ocorrem – após as aulas são, muitas das vezes, em grupos fechados e restritos. Seria mais benéfico que também pudessem participar todos os alunos interessados, podendo até os mais experientes no tema discutido contribuir.

- Materiais de estudo desatualizados

  Muitos materiais de estudo e conteúdo de disciplina estão desatualizados, sem respostas ou possuem pequenos erros que não são ajustados período após período. Com o material em um sistema colaborativo, todos podem apontar, discutir, encontrar soluções de problemas e corrigir eventuais erros encontrados nos materiais.

- Desinteresse e desmotivação

  Existe um modelo clássico de aula, onde o professor introduz um conceito teórico após outro para, ao final, o aluno formar um raciocínio e começar a dominar um determinado tema. No entanto, o excesso de foco na teoria pode desmotivar uma grande parte de estudantes que possui a curiosidade natural de não apenas saber em que determinado momento e como implementar uma definição, como também colocá-la em prática em forma de pequenos projetos. Esse cenário pode mudar com comunidades ativas em vários assuntos, gerando discussões e motivando a união de alunos para colocar em prática diversos conceitos que foram aprendidos durante as aulas.

- Gerenciamento de conteúdo e comunicação nas ferramentas utilizadas atualmente

  A maioria das disciplinas utiliza grupos em rede sociais e mensageiros instantâneos como canais de comunicação. Além de gerar distrações a todo instante, o conteúdo postado não é fácil de ser gerenciado, categorizado e encontrado. Quando vários canais de comunicação são criados para uma mesma disciplina de acordo com período ou professor, os problemas encontrados se agravam.

1.2 MODELOS ATUAIS

São utilizadas diversas maneiras para estabelecer a comunicação entre o professor de uma disciplina e os estudantes inscritos em um determinado período. Algumas soluções são mais formais que outras, como por exemplo a utilização do Moodle oficial do DCC quando comparamos com a criação de um grupo na rede social Facebook.

Independente do sistema adotado, é possível estabelecer o mínimo de comunicação necessária para administrar uma turma durante um período, ou seja, aplicar trabalhos, marcar provas, enviar comunicados e afins. Porém quanto menor a comunicação, mais

unilateral ela é, e isso faz com que os alunos apenas enviem suas respostas aos comunicados dos professores.

O problema não está apenas na escolha e na usabilidade de um determinado sistema, mas também na restrição e controle de acesso dos modelos. Muitas vezes essas dificuldades diminuem a interação entre a classe e contribuem para a falta de discussões sobre temas pertinentes.

1.2.1 Moodle

Alguns docentes optam pela utilização do Ambiente Virtual de Aprendizagem da UFRJ, que é um sistema cujo objetivo é organizar o material de classe, enviar e receber tarefas. O mesmo funciona bem para tais propósitos, embora não tenha uma interface de usuário amigável.

Neste modelo, predomina a comunicação unilateral do professor aos alunos, fator intensificado pela falta de uma interface para comunicação entre os alunos. Além disso, os conteúdos são disponibilizados de acordo com o período letivo e o professor da disciplina, o que significa que diversos grupos serão abertos para uma mesma disciplina, fator que contribui para duplicação e fragmentação desnecessária de conteúdo.

Um outro fator problemático é a presença de senhas para ingressar nos cursos. Um aluno não inscrito na disciplina não consegue acessar seu material para consulta, seja antes ou após o curso. Não existe motivo para essa barreira de segurança, uma vez que discentes não possuem permissão para fazer qualquer alteração no material.

1.2.2 Grupo de Redes Sociais

A utilização de redes sociais, seja por parte de alunos ou professores, está presente no cotidiano, inserida no mesmo como certeza de acesso por pelo menos algumas vezes. Por esse motivo, a praticidade, pode parecer uma boa ideia gerenciar um grupo de disciplina em sites como o Facebook.

O acesso a esse tipo de sistema está, geralmente, ligado a lazer e vida pessoal. Logo quando ingressamos, estamos sujeitos às mais diversas formas de distração. Postagens de amigos, notificações, mensagens privadas, notícias, todos esses fatores influenciam no tempo que os estudantes vão se dedicar, e também conseguir o foco necessário, para acessar o conteúdo da sua disciplina acadêmica nesse ambiente.

Utilizando o Facebook – rede social atualmente mais utilizada – como exemplo, é possível notar que seu gerenciamento de grupos não funciona de maneira satisfatória, principalmente quando um usuário está ingressado em diversos deles. Por esse motivo, a maioria dos usuários acaba saindo dos que estão menos ativos, geralmente grupo de disciplinas que já cursaram, uma vez que os mesmos interferem na experiência do usuário no momento de utilização dos seus grupos pessoais.

O gerenciamento de grupos também carece das funcionalidades necessárias para gerir conteúdo acadêmico. A ferramenta de busca do grupo não é efetiva para esses casos e torna bastante difícil a navegação se um usuário necessita buscar uma informação passada. Outro fator que prejudica bastante a interação entre os membros é a falta de possibilidade de enviar arquivos de documentos para uma postagem, sendo possível apenas enviar fotos.

1.2.3 Dontpad

O Dontpad ganha usuários devido à sua praticidade, que torna muito convidativo o fato de apenas digitar um complemento em uma URL para obter o seu próprio espaço. No entanto a sua falta de segurança e ausência de gerenciamento de acesso – ponto que não recebeu atenção pela proposta básica do sistema – tornam inviáveis sua utilização em sala de aula.

Qualquer usuário pode editar, até mesmo apagar, qualquer conteúdo existente caso acesse seu endereço eletrônico. Isso ocasiona uma perda de confiabilidade na qualidade material e até mesmo no seu acesso, uma vez que o mesmo pode deixar de existir subitamente.

Outro ponto importante é que o site não possui nenhuma maneira de interação entre seus usuários, ou seja, não é possível trocar mensagens, enviar arquivos ou fotos, criar um grupo, entre outras funcionalidades que os usuários estão acostumados a utilizar no cotidiano.

1.2.4 Grupo de Mensageiro Instantâneo

Este meio de comunicação pode parecer mais prático devido à comunicação direta e menor tempo de resposta entre os usuários, porém existem outros fatores que contribuem para a falta de sucesso na utilização desse tipo de ferramenta como gerenciador de grupos de estudos e conteúdos.

Mesmo que seja possível enviar documentos e fotos de maneira prática e rápida utilizando telefones celulares e aplicativos como WhatsApp e Telegram, não existe nenhuma maneira de categorizar ou indexar essas mídias. Consequentemente, a busca posterior por esse tipo de material é muito prejudicada, podendo gerar até materiais duplicados e um aumento significativo nas mensagens do grupo.

O volume de mensagens também é um transtorno. Nem todos os alunos possuem a mesma rotina e seus momentos dedicados aos estudos variam, por esse motivo não é incomum estudantes encontrarem um número enorme de mensagens não lidas devido a uma discussão que não puderam participar. Nesse caso, é mais difícil endereçar uma dúvida e até mesmo conseguir rever o que ocorreu previamente no grupo.

É possível destacar, do mesmo modo, a questão da privacidade. Em muitos mensageiros instantâneos, é necessário que o usuário informe o seu número de telefone e, por mais

que essa prática seja cada vez mais comum, esse dado continua sendo um canal direto de comunicação com uma pessoa. Transtornos podem ocorrer caso o contato de um membro seja utilizado de forma inapropriada.

1.3 PROPOSTA

O Motiva é um sistema colaborativo organizado para facilitar a troca de conhecimento e gerenciamento de conteúdo em tempo real. A ideia é que qualquer pessoa inserida no contexto acadêmico, seja aluno ou professor, possa gerar uma postagem sobre um determinado tópico e os demais usuários possam interagir de acordo com seu interesse.

No sistema, é possível que um usuário escreva uma postagem, receba notificações sobre novas publicações relacionadas a algum tema relevante, favorite postagens, faça comentários e ainda sugira temas que não possui conhecimento, porém gostaria que um outro participante conhecedor do assunto escreva sobre.

O principal objetivo do Motiva é fazer com que todos os usuários possam compartilhar conhecimento e gerar discussões em tempo real de maneira organizada, eliminando todos os problemas dos outros meios de comunicação citados anteriormente.

2 O SISTEMA

2.1 CADASTRO

Na versão inicial do Motiva, é necessário que o usuário seja um estudante com DRE e email, ou um Grupo de Extensão reconhecido, ambos do curso de Bacharelado em Ciência da Computação na UFRJ, ou um docente com email no domínio ufrj.com.br. É extremamente importante que não haja anonimato para o bom funcionamento do sistema.

O cadastro é feito manualmente e o usuário recebe uma senha inicial por email. Uma vez que o mesmo faça login no sistema, ele é direcionado para a ação de alterar a senha inicial e escolher uma nova para sua maior segurança. É importante ressaltar que apenas usuários cadastrados podem acessar o sistema e todo seu conteúdo.

2.2 ÁREA DO USUÁRIO

A área do usuário é a tela principal do sistema, nela está contida elementos como resumo do usuário, barra de busca, área de notificação e postagens mais recentes categorizadas.

Ao lado esquerdo está o resumo do usuário. Nele é possível que o usuário veja sua foto de perfil e seu nível atual, assim como quantos pontos faltam para que ele avance de nível e suas principais conquistas exibidas. Também existe a possibilidade de acessar o seu perfil detalhado, suas postagens favoritas, configurações do sistema e fazer logout.

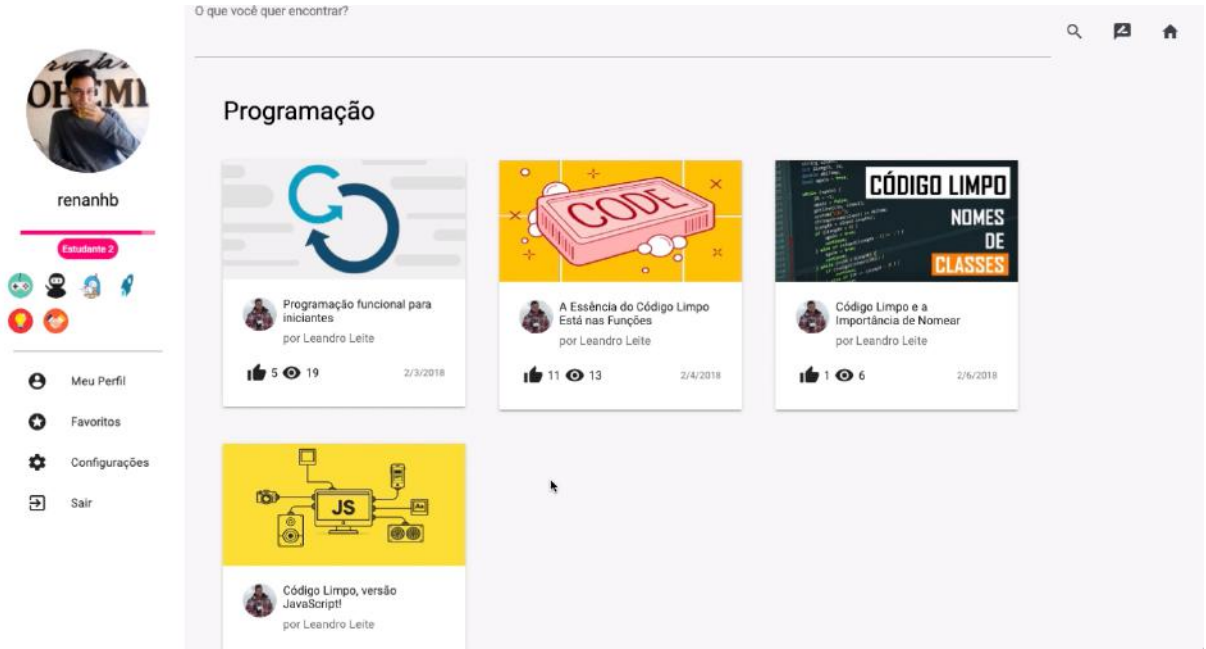

Figura 1 – Tela principal do Motiva

2.3 NÍVEIS E CONQUISTAS

Existem recompensas de acordo com a interação de quem usa o sistema. Conforme os usuários têm suas publicações lidas e curtidas ou leem e curtem as publicações de outros, são somados pontos de experiência que são acumulados e ditam o nível de um usuário, como por exemplo, os níveis Calouro e Profissional.

Para situações mais pontuais, existem as conquistas. Elas também possuem diferentes estágios – do mais básico ao avançado – e são dadas como recompensa quando alguém realiza uma determinada ação ou um conjunto de ações. Por exemplo, quando um usuário publica uma postagem pela primeira vez, ele recebe a conquista "Primeira postagem", assim como quando um usuário mais avançado realiza sua quinquagésima postagem, ele recebe a conquista "Publicador imparável".

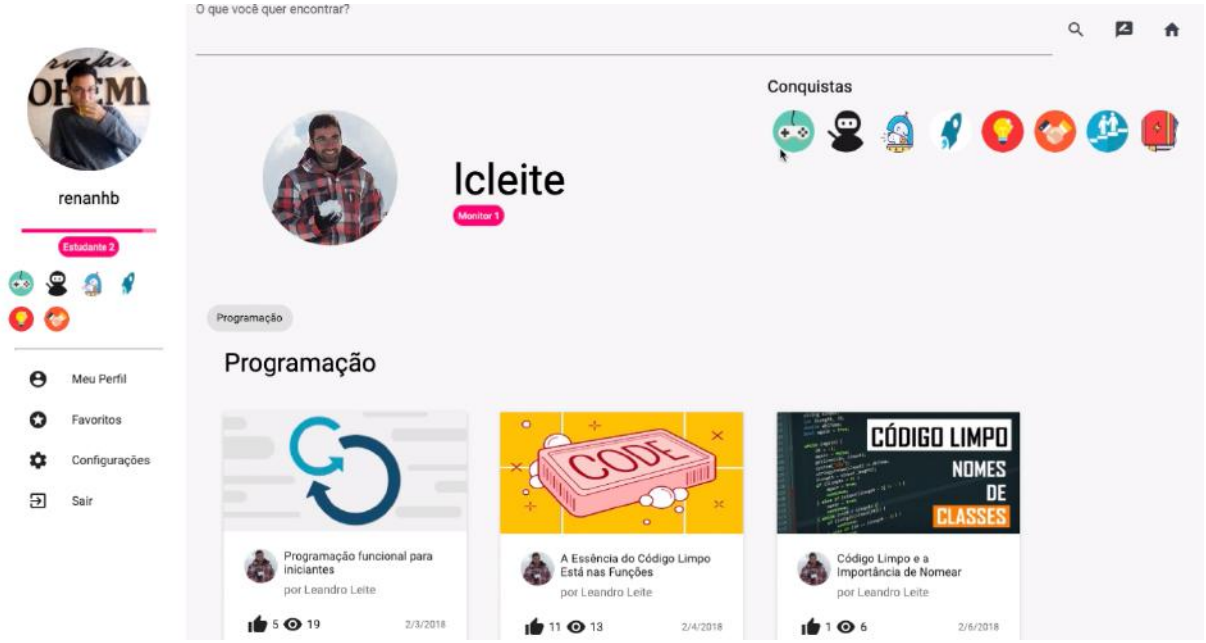

Figura 2 – Perfil de um usuário com suas conquistas e nível

2.4 REALIZANDO UMA BUSCA

É possível inserir um termo de busca na barra que fica no topo da tela principal. Quando o usuário confirma a busca, os resultados retornados são divididos e classificados entre usuários, tópicos e postagens. A procura por usuário é feita de acordo com seu nome e sobrenome, assim como as postagens são encontradas pelo seu título ou pelo conteúdo do seu texto. No caso dos tópicos, quanto uma postagem é escrita, o autor pode especificar os assuntos referentes à publicação por meio de tags, como por exemplo "Grafos", "Android", "Cálculo", etc. Então, a procura por tópicos retorna postagens que sejam classificadas de acordo com o tópico relacionado.

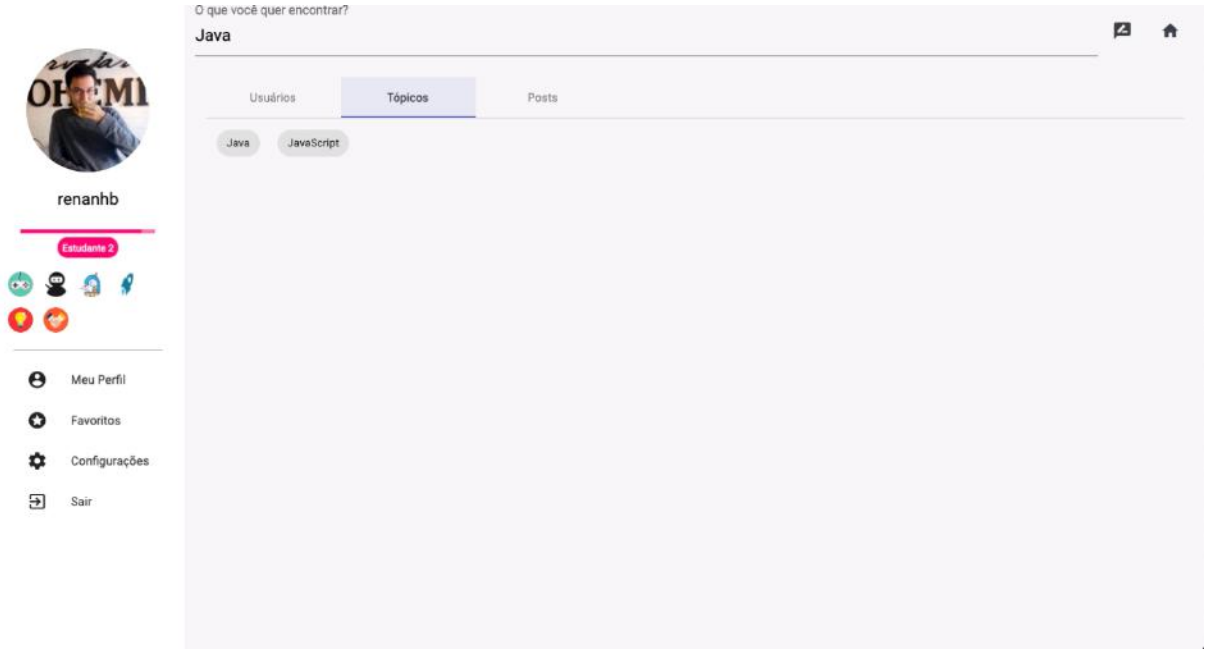

Figura 3 – Retorno de uma busca por tópicos

2.5 ESCREVENDO UMA POSTAGEM

Primeiramente, o autor tem a opção de escolher uma imagem para ilustrar a sua postagem. Não é uma etapa obrigatória, porém é altamente recomendada, pois a imagem escolhida será exibida tanto nas buscas quanto nas páginas iniciais de outros usuários.

Em seguida, o autor deve escolher um título para sua postagem e categorizá-la de acordo as tags já existentes no sistema ou criando novas quando necessário. Essa necessidade ficará clara, pois quando o usuário começa a escrever uma tag, o sistema identifica e dá sugestões para autocompletar a mesma e, se não aparecer nenhuma opção para o caso, é porque a tag ainda não existe e será criada no momento da postagem.

Finalmente, é possível que o autor utilize uma caixa de texto que facilita a formatação e a escrita do conteúdo da publicação. Com essa ferramenta, é possível formatar o texto com os principais estilos e, inclusive, escrever fórmulas matemáticas.

Quando todo o conteúdo estiver pronto, o botão "Postar" publicará a postagem e o autor será levado à sua postagem publicada. A tela de postagem exibe tudo o que foi previamente inserido pelo autor, com o adicional da média de tempo que uma pessoa levará para ler toda aquela publicação. Esse cálculo é baseado no total de palavras e utiliza a média normal de uma pessoa adulta, que é de 160 palavras por minuto.

2.6 INTERAGINDO COM UMA POSTAGEM

Ao encontrar uma publicação, existe a possibilidade de que o usuário interaja de algumas maneiras. A mais básica é a simples leitura do conteúdo da postagem, que

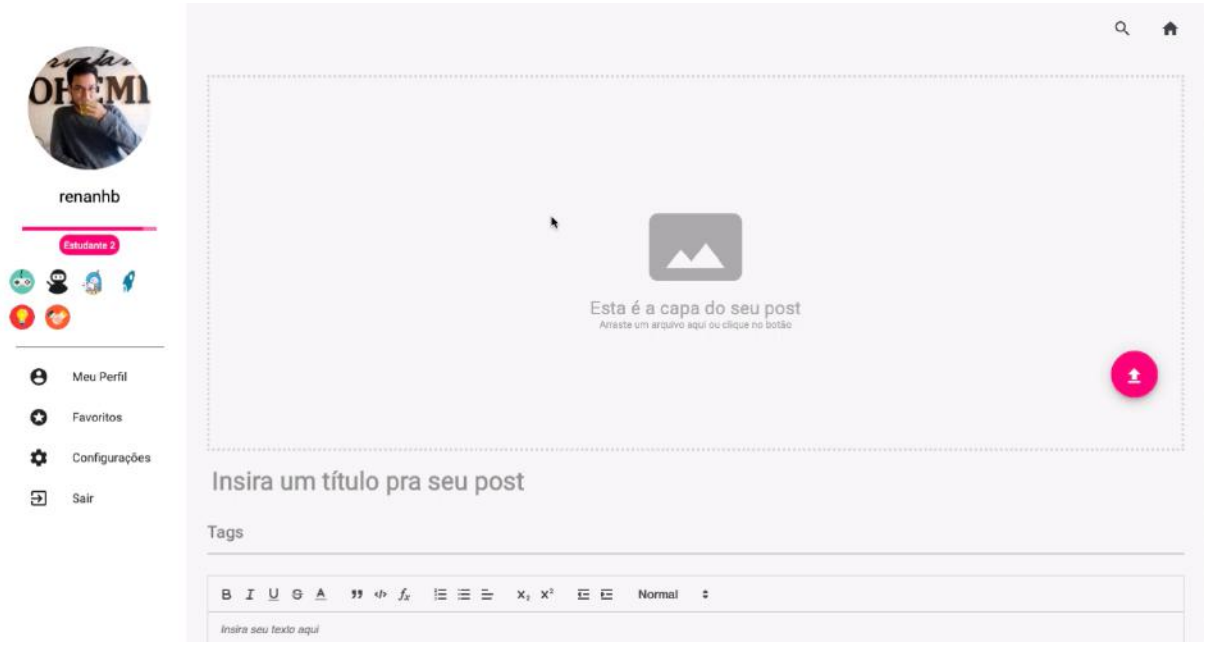

Figura 4 – Tela de criação de postagem

já garante pontos de experiência ao leitor e ao autor. Outras ações rápidas são as de curtir e favoritar uma publicação. A curtida incentiva e reconhece o trabalho do autor, e favoritar uma postagem ajuda a não perder de vista uma leitura e facilita sua releitura posteriormente.

É possível, e recomendado, que as publicações gerem discussões. Para que esse cenário aconteça, toda postagem possui uma seção de comentários onde todos os usuários do sistema podem interagir. Basta escrever no campo ao final do post e confirmar para que o comentário seja publicado.

Vale interessante ressaltar que das interações citadas, as ações de curtir e comentar geram notificações em tempo real para o autor e para quem está acompanhando a postagem. No caso da curtida, somente o autor é notificado, pois ele é o mais interessado nesse tipo de interação. Quando surge um novo comentário, não só o autor, mas todas as outras pessoas que curtiram ou comentaram na publicação também recebem uma notificação.

2.7 SUGERINDO UM TÓPICO DE INTERESSE

Uma vez que algum utilizador do sistema não encontre um tópico específico que gostaria de aprender mais sobre, é possível que ele sugira esse assunto a fim de motivar algum entendedor a fazer uma publicação. Para que não aconteça facilmente casos de sugestões de tópicos duplicados, o sistema oferece uma busca ao usuário antes da sua sugestão.

As propostas de assuntos são ordenadas de acordo com o interesse dos usuários do sistema. Caso alguém concorde em querer ver alguma publicação sobre determinado assunto sugerido, basta sinalizar com uma curtida. Também é possível alertar sugestões

duplicadas e sinalizar que uma sugestão já foi atendida indicando com qual postagem.

3 REFERENCIAL TEÓRICO

3.1 VALIDAÇÃO DA IDEIA

Encontrar um problema no dia-a-dia pode não ser uma tarefa tão difícil, mas encontrar uma solução para ele pode ser algo mais complicado. É necessário saber se o problema encontrado é realmente um problema sem solução e se há mais pessoas que enfrentam este problema. Mesmo que esses dois pontos se confirmem, ainda é necessário saber se a solução proposta realmente resolve o problema.

Neste momento precisa-se ter cautela. Há um enorme risco em se desenvolver a solução, que por vezes, possui custos de desenvolvimento altíssimos. Desta forma, desenvolver a solução por completo para depois validar se ela de fato resolve o problema se torna uma opção arriscada demais. Para contornar esta limitação, recorremos ao desenvolvimento de um menor produto com valor para o usuário, mais conhecido como MVP.

"Um produto mínimo viável (MVP) ajuda os empreendedores a começar o processo de aprendizagem o mais rápido possível"(RIES, 2011, p. I). Diferentemente do modelo mais "tradicional" de desenvolvimento, onde há um período de desenvolvimento até se chegar a uma "perfeição" de produto e daí disponibilizar para o usuário, o MVP se baseia na ideia de começar o processo de aprendizado do comportamento do usuário.

Desta forma, o MVP se diferencia de um protótipo ou teste de conceito. A ideia é que haja o início da construção de um produto o mais rápido possível, para que se aprenda e se molde ao comportamento e necessidades dos usuários. O MVP é apenas a "maneira mais rápida de percorrer o ciclo construir-medir-aprender de feedback com o menor esforço possível"(RIES, 2011, p. I)

O ciclo construir-medir-aprender é uma atividade fundamental quando se busca transformar uma ideia em um produto. O processo que envolve esse ciclo consiste em construir um MVP, medir como os usuários reagem a ele e, então, aprender se é o caso de mudar de ideia ou mantê-la.

3.1.1 Validation Board

Validar a ideia de acordo com uma reação do usuário é uma tarefa bastante subjetiva. Para ajudar nessa etapa, foi desenvolvido pela Lean Startup Machine (Lean Startup Machine, 2014) um quadro, demonstrado na figura 5, que auxilia nesse processo de validação. Ele é composto por alguns espaços que colaboram para a análise dos resultados do produto.

Um dos primeiros espaços ser preenchido é o "Customer Hypothesis" (Hipótese do Cliente, tradução nossa)(Lean Startup Machine, 2014). Trata-se do momento em que se "cria" o perfil do usuário alvo do projeto, sempre pensando em uma pessoa em específico.

Figura 5 – Quadro de validação (Lean Startup Machine, 2014)

Deve-se, como exemplo, preencher o campo com "aluno do segundo período de ciência da computação" ao invés de "alunos de computação".

Após identificado o perfil de usuário, deve-se preencher o campo de "Problem Hypothesis" (Hipótese do Problema, tradução nossa)(Lean Startup Machine, 2014). Neste espaço deve-se conjecturar um problema do usuário identificado que pode ser resolvido pela a ideia inicial. Este problema "deve ser específico para a pessoa"(Lean Startup Machine, 2014)(tradução nossa), e não um problema generalizado.

Com a definição dos principais elementos da hipótese de problema, deve-se pensar em como serão feitos os experimentos para validar a ideia. Esta etapa da validação da ideia requer um reflexão sobre o que deve ser aprendido. Desta forma, inicia-se o processo de preenchimento do espaço "Core Assumptions" (Suposições Centrais, tradução nossa)(Lean Startup Machine, 2014).

Esse espaço é preenchido com possíveis cenários que acredita-se que sejam fatos. Desta forma, lista-se todas as premissas necessárias para a realização da ideia, priorizando-as de acordo com o risco e grau de incerteza que tal suposição tem para que a ideia seja validada.

Com os cenários priorizados, seleciona-se o que possui mais risco para a ideia. Esta seleção é necessária para que a ideia seja invalidada já no primeiro cenário caso este seja invalidado. O cenário mais arriscado é então levado para o campo "Riskiest Assumption"

(suposição mais arriscada, tradução nossa)(Lean Startup Machine, 2014).

Neste momento da validação, têm-se o usuário-alvo, o problema que ele possui e um cenário que deve ser verificado se é real. Para que isso ocorra, precisa-se definir qual o método a ser utilizado para validar (ou invalidar) a ideia e qual será o critério de aceitação do mesmo.

Para a definição de qual método será adotado para validar um cenário, deve-se pensar em uma maneira rápida para a colheita dos resultados. Pode ser aplicado aqui um questionário para os usuários, assim como ser desenvolvido um protótipo da ideia ou então se simular o problema. Definido o método a ser utilizado, é preciso se estabelecer um critério mínimo de sucesso. Este critério irá definir de tal suposição é válida e se deve-se prosseguir com a ideia.

Com os resultados em mãos, é preciso tomar a decisão para o próximo passo. Caso os resultados tenham sido positivos, ou seja, a suposição seja confirmada, move-se o cenário para a área de resultados validados e uma nova suposição deve ser validada. Caso os resultados sejam negativos, move-se o cenário para a área de resultados invalidados e armazena-se de alguma forma os dados e aprendizados adquiridos na fase de validação.

No caso em que o resultado é negativo, parte-se para o "pivoteamento". Esta etapa consiste em realizar uma nova análise do usuário e seu problema, assim como gerar novas suposições de cenários. Ao fim desta reanálise, todo o processo de validação é iniciado.

3.2 SCRUM

O gerenciamento de um projeto é tão importante quanto o desenvolvimento do mesmo. Uma má gerência pode resultar em um projeto inacabado ou em um projeto terminado, mas que não resolve o problema inicial e, por vezes, adiciona mais problemas a serem solucionados.

Um software deve ser desenvolvido para as pessoas que irão utilizá-lo. Sendo assim, é necessário que se haja transparência quanto ao que está sendo desenvolvido. Neste cenário, uma metodologia de gerenciamento que deixe a evolução do desenvolvimento e suas dificuldades às claras se torna um aliado. E o Scrum é uma metodologia com esses princípios.

"Na essência, o Scrum se baseia em uma ideia simples: quando começamos um projeto, por que não verificar a intervalos regulares se ele está indo no caminho certo e se aquilo é realmente o que as pessoas querem? E por que não se perguntar se é possível aprimorar a forma como você está trabalhando para obter resultados melhores e mais rápidos, e o que poderia estar impedindo você de fazer isso? O nome disso é ciclo de 'inspeção e adaptação'."(SUTHERLAND, 2016, p. I)

3.2.1 Product Backlog e Product Owner

Na metodologia Scrum, durante a análise de requisitos de um software, as funcionalidades mapeadas são tratadas como histórias. Nelas, personagens são "criados para que fique mais fácil identificar o que o software precisa satisfazer. Todas essas histórias são "guardadas" no Product Backlog.

O Product Backlog possui todas as histórias que um software inicialmente possui. Essa lista de histórias serve como um alicerce para saber o que ainda falta a ser feito para o software ficar pronto. Esta lista é alterada ao longo do desenvolvimento, sendo histórias removidas conforme são feitas ou quando se mostram não mais necesárias para o sistema, assim como novas histórias podem ser adicionadas caso haja uma necessidade ainda não mapeada para o sistema. Com todos as histórias mapeadas, é necessário identificar qual o caminho que o desenvolvimento vai seguir. Neste momento, se faz necessária a figura do Product Owner.

O Product Owner (P.O.) é a pessoa que definirá o que a equipe fará, produzirá ou realizará. "Ela leva em consideração os riscos e as recompensas, o que é possível, o que pode ser feito e aquilo pelo que tem paixão"(SUTHERLAND, 2016, p. I). O P.O. deve priorizar as tarefas levando em consideração alguns aspectos de cada tarefa. Como mostrado na figura 6, a história a ser desenvolvida deve ser possível de implementar, vender e amar. Para se chegar nessa valoração, é necessário priorizar e estimar cada história listada no backlog. Com todas as histórias priorizadas, pode-se iniciar a sprint.

3.2.2 Sprint

Na metodologia Scrum, os objetivos são sequenciais e devem ser atingidos em um intervalo definido. Ao final de cada intervalo, deve-se haver uma parte do produto entregue que seja funcional. Desta forma, é possível mostrar o andamento do projeto a quem interessa e, se necessário, adaptar as novas funcionalidades às necessidades encontradas. Estes intervalos de inspeção do produto e adaptação são chamados de sprints.

No início de cada sprint acontece uma reunião, chamada de "Sprint Planning Meeting", para o planejamento da mesma. "A equipe determina a quantidade de trabalho que acredita ser capaz de realizar nas duas semanas seguintes. As tarefas são selecionadas a partir daquela lista de prioridades e anotadas em post-its, que são colados na parede. A equipe resolve quantas tarefas será capaz de executar durante o sprint"(SUTHERLAND, 2016, p. I). Esta lista de tarefas a serem realizadas em uma sprint é denominada "Sprint Backlog".

Ao final de cada sprint ocorre a "Sprint Review Meeting", uma reunião onde os integrantes do grupo de desenvolvedores mostram o que conseguiram realizar dento da sprint e analisam quantas tarefas foram concluídas. Nesta etapa é estabelecido o ritmo de trabalho da equipe. Ao final de algumas sprints já se torna possível identificar a velocidade

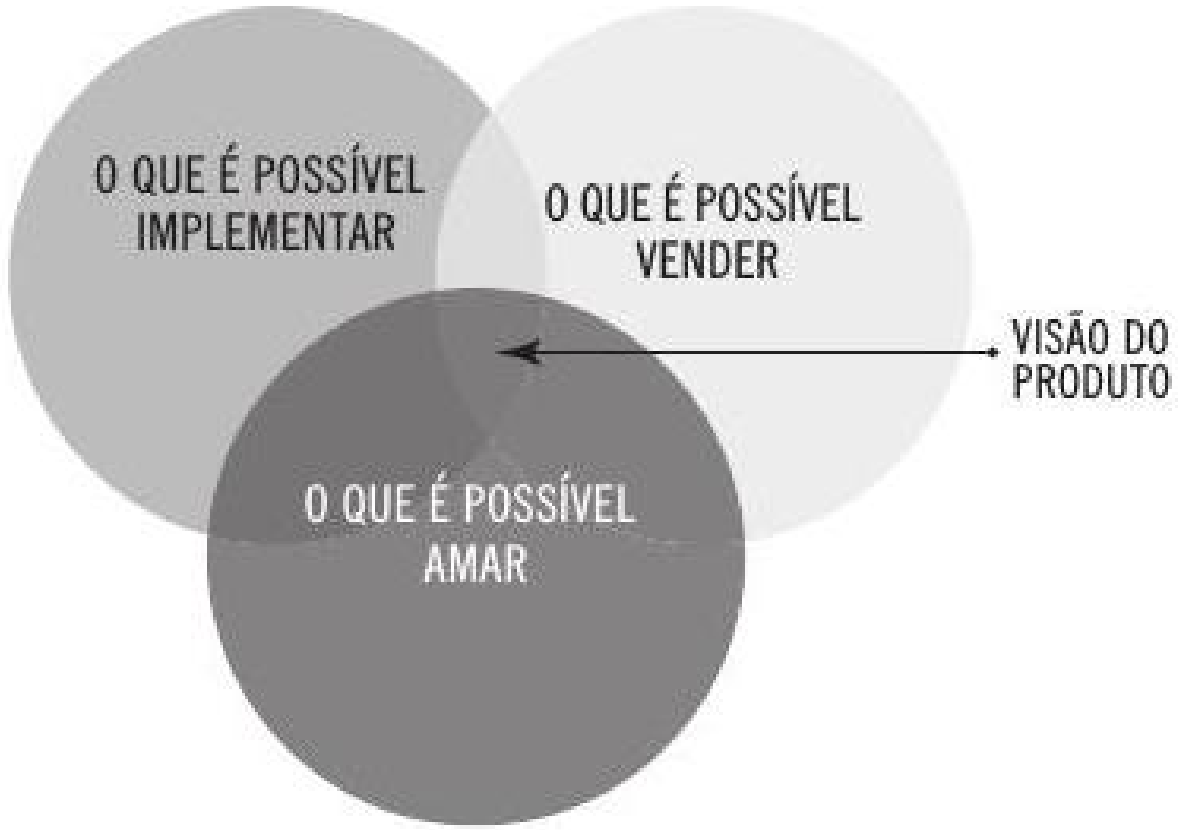

Figura 6 – Diagrama de visão do produto (SUTHERLAND, 2016, p. I)

média da equipe e, portanto, quantas tarefas a equipe consegue realizar em cada sprint.

Após a apresentação e análise das tarefas realizadas e não realizadas, a equipe discute como cada coisa foi feita na sprint. O intuito desta etapa do processo, chamada de "Sprint Retrospective", é identificar melhores formas de se trabalhar para aumentar a velocidade do time. É neste momento que se identifica o que atrapalhou o andamento da sprint que passou e o que pode ser feito para remover os obstáculos que atrapalham o andamento do projeto (SUTHERLAND, 2016, p. I).

A figura responsável por remover obstáculos encontrados nas reuniões diárias, as "Daily Scrum", e nas Sprint Retrospective é o "Scrum Master". Além desta responsabilidade, a pessoa que assumir este papel deve assegurar que a equipe não se comprometa a entregar muito mais do que é capaz de realizar. Normalmente a figura de Scrum Master é realizada por um gerente de projeto ou então um líder técnico, mas não há objeções em ser qualquer membro da equipe.

3.3 SISTEMA DE CONTROLE DE VERSÃO

Ao longo do desenvolvimento de um software é essencial que se possa restrear todas as modificações que foram realizadas, podendo assim, identificar de modo mais rápido as

causas de um possível problema. Desta forma é necessário possuir um sistema de controle de versão (VCS) para que essa rastreabilidade possa ser possível. Os sistemas de controle de versão disponíveis nos dias atuais, como Git, SVN e Perforce, permitem que um arquivo possa voltar para um estado anterior e marcar um estado do projeto com uma versão, possibilitando assim que o projeto possa ser completamente revertido para outra versão. Eles permitem também a comparação entre os arquivos de diferentes versões.(CHACON, 2010, p. 2)

A utilização de um VCS permite também que o o projeto possa ser desenvolvido por mais de um colaborador, de modo que a integração entre as funcionalidades desenvolvidas por diferentes pessoas sejam realizadas sem perda de dados e evitando falhas humanas.

Para que o projeto seja compartilhado por mais de uma pessoa de forma assíncrona, sem a necessidade de que todos estejam com o código atualizado ao mesmo tempo, se faz necessário possuir um repositório conectado à internet, de modo que a qualquer momento seja possível atualizar a versão do código. Partindo dessa necessidade, há dois formatos de sistema aplicados por VCS para suprí-la.

3.3.1 Sistemas centralizados

Em um VCS com sistema centralizado, há um servidor único para armazenamento de todo o código versionado e o número de clientes que conferem os arquivos desse servidor (CHACON, 2010, p. 3). Essa abordagem, ilustrada na Figura 7, permite que todos os que estão participando do projeto tenham, até certo ponto, conhecimento do andamento do projeto. Além disso, administradores conseguem saber exatamente o que foi feito e por quem foi feita cada modificação do sistema.

Contudo, a adoção desse tipo de sistema cria uma forte dependência com o servidor onde o código está armazenado. Caso ocorra algum problema com esse servidor, desde ataques de hackers a problemas de hardware, e nenhum backup tenha sido realizado, todo o histórico do código desenvolvido corre o risco de ser perdido, sendo mantido apenas alguns estados do código nos computadores dos desenvolvedores - o que não garante que o último estado esteja salvo.

3.3.2 Sistemas distribuídos

Em sistemas distribuídos, essa forte dependência do servidor não existe. Ao invés de pegar apenas o último estado do código, os clientes fazem uma cópia de todo o repositório, incluindo todo seu histórico. Desta forma, caso o servidor tenha algum problema, qualquer cliente possui um backup de todo o repositório, podendo assim recuperar o estado do projeto (CHACON, 2010, p. 4). Podemos representar o funcionamento deste sistema como na Figura 8

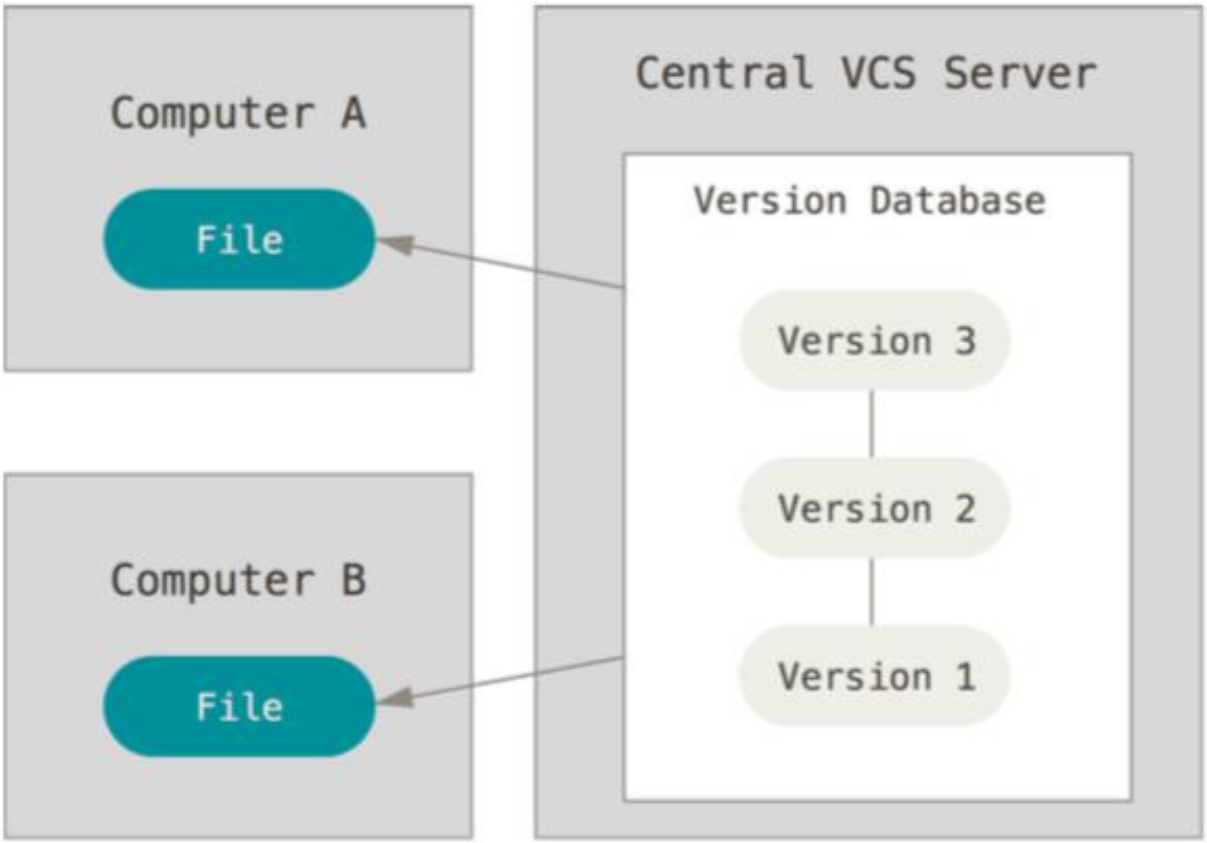

Figura 7 – Representação de como funciona um VCS centralizado (CHACON, 2010, p. I)

3.4 CLEAN ARCHITECTURE

No processo de construção de um software, é comum encontrarmos algumas mudanças de escopo enquanto o software está sendo construído. Desta forma, se faz necessário ser capaz de construir um sistema em que se possa realizar as modificações sem estas tenham um grande impacto sob o que já foi construído. Uma forma de se atingir esse resultado é pensar em uma arquitetura de software que permita essas modificações.

Segundo Robert Cecil Martin, idealizador do Clean Architecture, "Quando um software é feito corretamente, ele necessita de uma fração dos recursos humanos para criar e manter. Mudanças são simples e rápidas. Defeitos são poucos e ocorrem com pouca frequência. O esforço é minimizado, enquanto que a funcionalidade e flexibilidade são maximizadas."(MARTIN, 2017, p. I)(tradução nossa). Partindo desta afirmação, tem-se como objetivo encontrar uma arquitetura que seja capaz de fornecer todas estas particularidades.

Existem hoje vários modelos de arquitetura que são amplamente usados pelos desenvolvedores para a construção de um software, tais como MVC (Model View Controller), MVVM (Model View ViewModel), EDA (Event-driven Architecture), entre outros. Todos eles são recomendados para resolver alguns problemas específicos, mas nem sempre estão aptos a resolver outros problemas enfrentados na construção de um software robusto

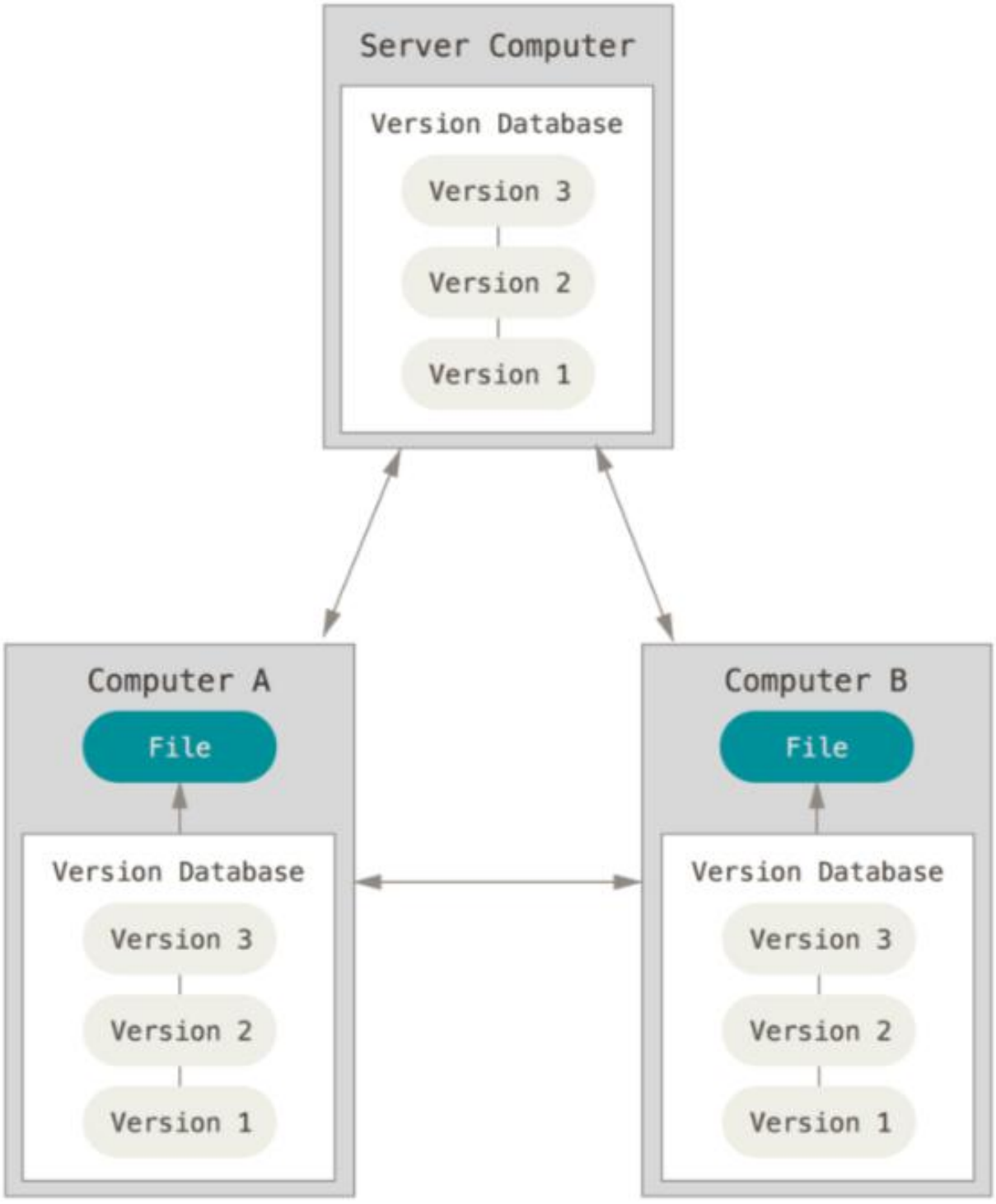

Figura 8 – Representação de como funciona um VCS distribuído (CHACON, 2010, p. I)

como o Motiva.

A proposta do Clean Architecture é fornecer uma solução em arquitetura que sirva para, se não todos, quase todos os problemas hoje existentes. Ela tem como objetivo a separação de responsabilidades entre as camadas e objetos dentro do projeto.

A separação de responsabilidades que permite modularizar e e tornar cada camada

independente entre si. Essas duas características são bastante importantes quando se quer criar um sistema que seja escalável e que possua uma manutenção de baixo custo.

Havendo camadas independentes entre si, pode-se verificar que o sistema é: independente de framework, de interface de usuário, de banco de dados e de agentes externos. Todas estas características permitem que haja uma troca - de banco de dados, por exemplo - sem que afete o funcionamento principal do software.

Desta forma, Robert Cecil Martin identificou as seguintes camadas de um software: Entidades, Casos de Uso, Adaptadores de Interface e Frameworks e Drivers (MARTIN, 2017, p. I), conforme a figura 9. Todas estas independentes entre si, de modo que nenhuma de fato sabe como a outra está implementada.

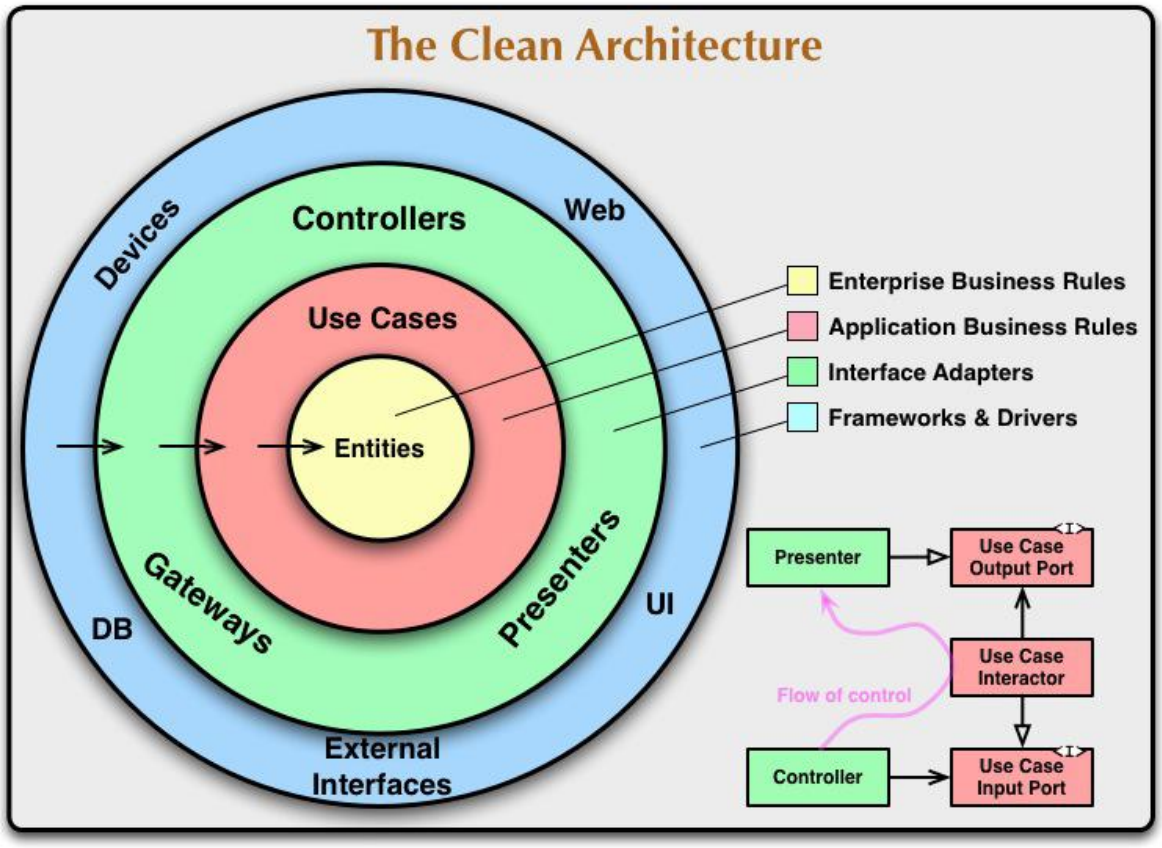

Figura 9 – Representação em camadas da arquitetura clean (MARTIN, 2017, p. I)

3.4.1 Entidades

Na camada de entidade estão inseridas as regras para negócio no qual o software está sendo desenvolvido. São todas as particularidades que fazem ou salvam o dinheiro envolvido no negócio da empresa. "Uma entidade pode ser um objeto com métodos ou pode ser um conjunto de estruturas de dados e funções. Não importa o tamanho,

desde que as entidades possam ser usadas por diferentes aplicações."(MARTIN, 2017, p. I)(tradução nossa)

Nesta camada, não é esperado que hajam modificações caso a forma de se navegar em um site, por exemplo, seja modificada. Se não houver nenhuma modificação operacional, as entidades não devem ser alteradas.

3.4.2 Casos de Uso

Nesta camada estão inseridas as regras de negócio específicas do software. Essas regras não devem ser utilizadas em um ambiente onde não haja a automação do processo. São regras que só fazem sentido no contexto do software.

"Um caso de uso é a descrição da forma como um sistema automatizado é usado. Ela especifica a entrada a ser fornecida pelo usuário e o passos do processo envolvido para produzir uma saída"(MARTIN, 2017, p. I)(tradução nossa)

Sendo assim, não é esperado que uma modificação nesta camada afete as entidades do software. Assim como não é esperado que esta camada seja afetada por mudanças no endereço da base de dados, assim como a tecnologia utilizada para armazenagem dos dados ou qualquer outra modificação na interface do software. A camada de Casos de Uso é isolada destes tipos de modificações.

3.4.3 Adaptadores de Interface

Nesta camada é realizada a conversão dos dados utilizados nas camadas de "casos de uso" e "entidade" para o formato mais adequado para agentes externos, como bancos de dados ou interfaces de usuário. Todos os componentes que estejam na camada de "entidade" ou "casos de uso" não devem saber nada sobre qual a tecnologia de banco de dados está sendo utilizada, ou então qual arquitetura de interface de usuário foi utilizada.

Também há necessidade nesta camada de haver um outro adaptador de dados, que converte os dados utilizados nos agentes externos para os dados utilizados internamente na aplicação (i.e. casos de uso e entidades).

3.4.4 Frameworks e Drivers

Esta camada é a camada mais externa de toda a arquitetura. É nela que são realizadas as conexões com os bancos de dados, são criadas as telas para visualização dos dados, etc. Normalmente, a maior parte do código escrito nesta camada tem o objetivo de se conectar com uma fonte de dados.

3.5 REST

Em um sistema onde a interação entre os usuários é a principal funcionalidade, se faz necessário estar bem atento à forma como que as informações são trocadas. Escolher uma solução que esteja embutida dentro do software atual pode não ser uma boa escolha. Deve-se escolher uma arquitetura onde a troca de informações independa da origem da solicitação dos dados.

Para que seja possível recuperar os dados independentemente da origem, é necessário utilizar o conceito de separação de responsabilidades. Como já abordado, a separação de responsabilidades permite modularizar as partes do software. Segundo Roy Thomas Fielding, "ao separar as preocupações da interface do usuário do preocupações de armazenamento de dados, melhoramos a portabilidade da interface do usuário através de múltiplas plataformas e a escalabilidade, simplificando os componentes de servidor"(FIELDING, 2000, p. 78)(tradução nossa).

Buscando a separação de responsabilidades, pode-se utilizar um serviço remoto de acesso aos dados do sistema, deixando o cliente responsável apenas pela exibição dos dados e o servidor pela criação, atualização, recuperação de remoção dos dados. Dentre as diversas arquiteturas existentes, que se permite estabelecer essa conexão entre cliente-servidor, há uma denominada "Representational State Transfer" (REST).

A REST é uma arquitetura baseada em rede derivada de várias outras arquiteturas do mesmo estilo, combinado com restrições adicionais que definem uma interface de conexão uniforme. Tais restrições permitem que a arquitetura do sistema seja simplificada e a visibilidade das iterações seja melhorada (FIELDING, 2000, p. 81).

A arquitetura REST é definida por seis restrições: arquitetura cliente-servidor, que satisfaz o conceito de separação de responsabilidades; sem estado, onde cada solicitação do cliente para o servidor deve possuir toda a informação necessária para se entender a solicitação (FIELDING, 2000, p. 78); possibilidade de persistência, onde os dados a serem respondidos devem possuir um indicativo para "armazenamento temporário" (cache) habilitado ou não; sistema em camadas, que permite a implementação de uma arquitetura hierárquica, onde cada camada não pode "enxergar" além da camada que ele está interagindo; código sob demanda, uma restrição opcional em que os servidores podem estender uma funcionalidade do cliente realizando a transferência e executando o código em forma de applets ou scripts; interface uniforme, em que a implementação de cada método é desacoplada do serviço que é fornecido.

3.5.1 REST aplicado ao HTTP

O HTTP é um protocolo da camada de aplicação da Web que define as propriedades que cada mensagem trocada entre cliente e servidor deve possuir (KUROSE; ROSS, 2013, p. 72). Mesmo havendo algumas incompatibilidades (FIELDING, 2000, p. 129) entre

REST e HTTP, há algumas características que satisfazem a arquitetura, como o fato de ser um protocolo sem estado, com possibilidade de reaproveitamento de resposta (cache) e uma interface uniforme. Desta forma, é possível utilizar o HTTP para implementar uma aplicação baseada em REST.

3.5.1.1 REST API

Uma aplicação API é um sistema desenvolvido para o servidor que auxilia outros sites ou clientes a realizar suas necessidades. "De modo geral, uma API (acrônimo para Application Programming Interfaces) expõe um conjunto de dados e funções para facilitar iterações entre programas e permite que eles troquem informações."(MASSé, 2012, p. 5)(tradução nossa). Uma aplicação REST API é, portanto, uma API que mantém a conformidade com a arquitetura REST.

Para manter a conformidade com o protocolo HTTP e modularizar as funções, uma REST API utiliza os métodos HTTP disponíveis para executar diferentes ações. De modo geral, para obter dados é utilizado o método GET, enquanto que para a criação de dados aplica-se o método POST. Caso seja desejado alterar um dado recorre-se ao método PUT e, para deletar um dado o método DELETE é usado.

3.6 WEBSOCKET

3.6.1 Contexto histórico

O conceito de WebSocket nasceu da necessidade de construir uma comunicação em tempo real e bilateral de maneira efetiva. Esse tipo de comunicação já era um desejo antigo no desenvolvimento web, no entanto, para obter tal resultado era necessário contornar a implementação de comunicações que não tinham exatamente esse propósito.

A web foi construída baseada no protocolo HTTP, que foi originalmente projetado para atender o modelo de requisição e resposta. No caso, simplesmente abrir uma conexão, descrever um conteúdo, receber uma resposta e fechar a conexão anteriormente aberta. Esse tipo de comunicação se mostrou efetivo durante muito tempo, pois na maior parte do tempo, os usuários necessitavam apenas carregar algumas páginas simples com texto e imagens e, em alguns casos, algum link direto para realizar o download de outro tipo de arquivo específico.

Com o avanço do ambiente online, as páginas foram se tornando cada vez mais próximas dos sistemas offlines, com diversas funcionalidades e uma maior necessidade de interação com o usuário. Durante essa mudança, alguns sites se destacavam por levar informação ao usuário sem que o mesmo tivesse que atualizar o seu navegador. Era o surgimento de uma comunicação onde o servidor poderia enviar respostas sem ter recebido uma requisição imediata.

3.6.2 Long polling

O long polling foi uma das técnicas mais utilizadas para tentar exercer uma comunicação bilateral. Seu princípio era criar uma conexão HTTP e deixá-la aberta até que não fosse mais necessária. Essa técnica se baseou na percepção de que não era necessário responder uma requisição de imediato. A maioria das requisições HTTP são respondidas pela mesma conexão que foi criada, no entanto, caso a conexão HTTP permaneça aberta após uma requisição, o servidor pode respondê-la momentos depois, ou até mesmo o cliente enviar novas requisições sem contexto algum com as anteriores.

Basicamente, o cliente fazia uma requisição HTTP inicial e sua conexão permanecia aberta, ou seja, essa requisição só seria respondida ao final de uma sessão com o servidor. Nesse período, com o canal de comunicação estabelecido, o servidor pode enviar respostas sem ter um conteúdo de dados padrão, o que significa que o mesmo pode atualizar diversas partes do site ou trazer informações diferentes para diferentes contexto mesmo sem ter uma requisição comandando esses retornos.

3.6.3 Surgimento do WebSocket

Por mais que o long polling pareça uma solução viável, além de ser uma adaptação de uma tecnologia com outra finalidade, sua implementação era bastante complicada e também era necessário lidar com complicações inesperadas da conexão via protocolo HTTP.

Em 2007, Michael Carter escreveu um artigo introduzindo a ideia de WebSockets à comunidade web e, em 2010, o Google Chrome foi o primeiro navegador a adotar e embarcar o suporte nativo à nova tecnologia que estava surgindo. Logo em seguida, em 2011, a RFC 6455 (FETTE; MELNIKOV, 2011) que abordava o protocolo WebSocket foi publicada. Hoje em dia, todos os navegadores modernos suportam o protocolo e ele faz parte do desenvolvimento web.

3.6.4 Como funciona

O WebSocket faz parte da pilha de protocolos TCP/IP, especificamente como parte da camada de transporte. Inicialmente, existe um handshake - processo automático de negociação e estabelecimento de conexão entre duas partes - entre o cliente e o servidor, feito por uma requisição HTTP do tipo Upgrade pelo cliente. O cabeçalho desse tipo de requisição muda o protocolo de comunicação, pois quando o servidor interpreta esse cabeçalho, percebe a troca do tipo de comunicação TCP e reage deixando de utilizar o HTTP para utilizar o protocolo WebSocket nas próximas interações. Em caso de utilização de conexão sem criptografia, a comunicação é feita através das porta 80 no esquema ws. Caso seja utilizada conexão criptografada, o esquema passa a ser o wss e a porta 443.

Estabelecida a conexão, a comunicação pode ser feita de maneira bilateral, ou seja, tanto o servidor quanto o cliente podem enviar mensagens independente do destinatário estar esperando por uma resposta. É intuitivo imaginar como esse tipo de ação ocorre no lado do servidor, uma vez que o modelo cliente-servidor é algo bem difundido e implementado, além do fato de que durante anos a internet funcionou apenas com os servidores reagindo às requisições dos usuários. Porém, é importante ressaltar que a comunicação do servidor para o cliente é feita por via de mensagens com conteúdo de dados sem nenhuma pré definição, ou seja, o cliente deve saber interpretá-los. O cliente reage às mensagens, enquanto a conexão estiver ativa, por meio de eventos que ocorrem quando uma mensagem é recebida.

O interessante é que como a conexão entre o servidor e o cliente para utilização de WebSockets é persistente, é possível enviar mensagens aos clientes destinatários de formas diferentes. Por exemplo, no ato de conexão e handshaking, o servidor pode identificar o cliente de maneira única e persistir essa informação de maneiras diferentes, como por exemplo, a criação de listas como canais de comunicação. Dessa maneira, é possível que o servidor não só envie mensagens para um destinatário, mas também para um grupo de destinatários de uma só vez, fazendo com que todos os que estão ativos e conectados no momento recebam a mensagem em tempo real.

3.7 GAMIFICAÇÃO

3.7.1 Conceito

O termo gamificação vem do inglês *gamification*, que nada mais é do que utilizar mecânicas e características de jogos para engajar, motivar comportamentos e facilitar o aprendizado de pessoas em situações reais que, normalmente, não são relacionadas a jogos. A prática de gamificar tem como diferencial despertar um maior engajamento do público e facilitar a obtenção de medidas dos resultados das ações de um determinado grupo de pessoas.

3.7.2 Motivações da Gamificação

Segundo o Framework Octalysis (CHOU, 2015), existem oito pontos principais que são tidos como alicerce da gamificação, que fazem com que o jogador seja envolvido pelo jogo. Esses pontos são chamados Propósito, Empoderamento, Influência Social, Imprevisibilidade, Evasão, Escassez, Sentimento de Posse e Realização.

3.7.2.1 Propósito

É nesse ponto que o jogador acredita que faz parte de um bem maior, que foi o escolhido para realizar algo. O sintoma imediato desse sentimento é se dedicar horas para

cumprir determinada tarefa.

3.7.2.2 Empoderamento

Além de precisar de maneiras para expressar sua criatividade, as pessoas também necessitam ver resultados imediatos das sua ações e constantemente responder a estímulos. É nesse ponto que o jogador é provocado e, quando bem sucedido, exibe seus resultados para os demais.

3.7.2.3 Influência Social

O elemento social direciona e afeta diretamente a motivação do jogador. Quando um jogador percebe o quão bom um amigo é, ou nota algo extraordinário que ele possui, esse jogador fica motivado e é direcionado a superar seu amigo, seja por um sentimento competitivo ou por inveja.

3.7.2.4 Imprevisibilidade

Quando não sabemos o que está por vir, o que vai acontecer em um determinado momento, nosso cérebro reage com engajamento e muitas vezes acabamos pensando demais sobre, despertando um sentimento de curiosidade constante. É por essa razão que muitos assistem filmes e leem livros.

3.7.2.5 Evasão

É o sentimento que se tem no caso de um jogador abandonar o jogo, onde todo o seu progresso é perdido. Perder um trabalho em andamento, ou simplesmente reconhecer o fato de que tudo o que foi feito até o momento será em vão caso o jogo termine, faz com que um jogador não queira abandonar um jogo.

3.7.2.6 Escassez

A escassez faz com que um jogador busque um objetivo que está diretamente ligado ao seu desejo. Por exemplo, é comum que em jogos existam níveis iniciais em que os personagens não podem obter determinados itens valiosos. Isso faz com que os jogadores desejem tanto algo, que passam a focar em jogar até que consigam atingir esses objetivos

3.7.2.7 Sentimento de Posse

Quando um jogador tem a sensação de que possui bens em um jogo, ele vai lutar para que sempre possa melhorar os seus bens e até acumular cada vez mais.

3.7.2.8 Realização

As sensações que mais laçam os jogadores são as de realizar progresso, desenvolver habilidades, crescer e superar cada vez mais novos desafios. Esse pilar da gamificação, geralmente, é aplicado junto com o pilar de Empoderamento, que ajuda no reconhecimento de uma realização.

3.7.3 Aplicação

Primeiramente, o cenário e o problema a ser resolvido devem ser mapeados. É necessário implementar a gamificação para solucionar um problema. É nessa etapa que mapeamos o comportamento do usuário com a finalidade de entender suas ações, necessidades, pensamentos e objetivos que visa alcançar.

Em seguida, de acordo com a solução do problema proposto, definimos a missão do jogo. A missão é a razão de existir do jogo, o objetivo principal da iniciativa de gamificação. Mesmo que o jogo não possua um fim, é necessário estabelecer uma missão para que o desenvolvimento do jogo seja guiado e vá de encontro ao objetivo do jogador.

Com o cenário montado, devemos definir quem são os jogadores. É preciso conhecer a motivação dos envolvidos com o jogo, tais como seus hábitos e suas características comportamentais, que vão definir a estratégia para o desenvolvimento do jogo.

Finalmente, desenvolva a mecânica do jogo. Ela é o núcleo do jogo. É nessa etapa que são definidas as variáveis e tudo que pode ser visto ou manipulado no jogo, como por exemplo, personagens, placares, regras e atividades que o jogador deve concluir. É importantíssimo definir nesse processo a forma de pontuação, as conquistas e recompensas, que são a base de uma gamificação bem sucedida.

3.7.4 Exemplos

O conceito de gamificação pode ser aplicado em qualquer situação. Citaremos, a seguir, alguns exemplos de diferentes setores, com e sem fins lucrativos.

A rede de supermercados Pão de Açúcar criou uma iniciativa em que os clientes precisam juntar selos para trocar por brindes. A cada R$ 30 em compras, os clientes recebem selos para colar em uma cartela e, a partir de um certo número de selos, a cartela pode ser trocada por diversos utensílios domésticos.

O site de *e-commerce* MercadoLivre passou a adotar elementos de gamificação nas suas compras. Um usuário recebe pontos, de acordo com o valor, ao efetivar uma compra e esses pontos são acumulados e, a medida que vão aumentando, geram diversas vantagens como descontos e fretes grátis para o usuário. Além de trabalhar com recompensas, o site também trabalha com reconhecimento por meio de medalhas para cada nível que um usuário vai avançando.

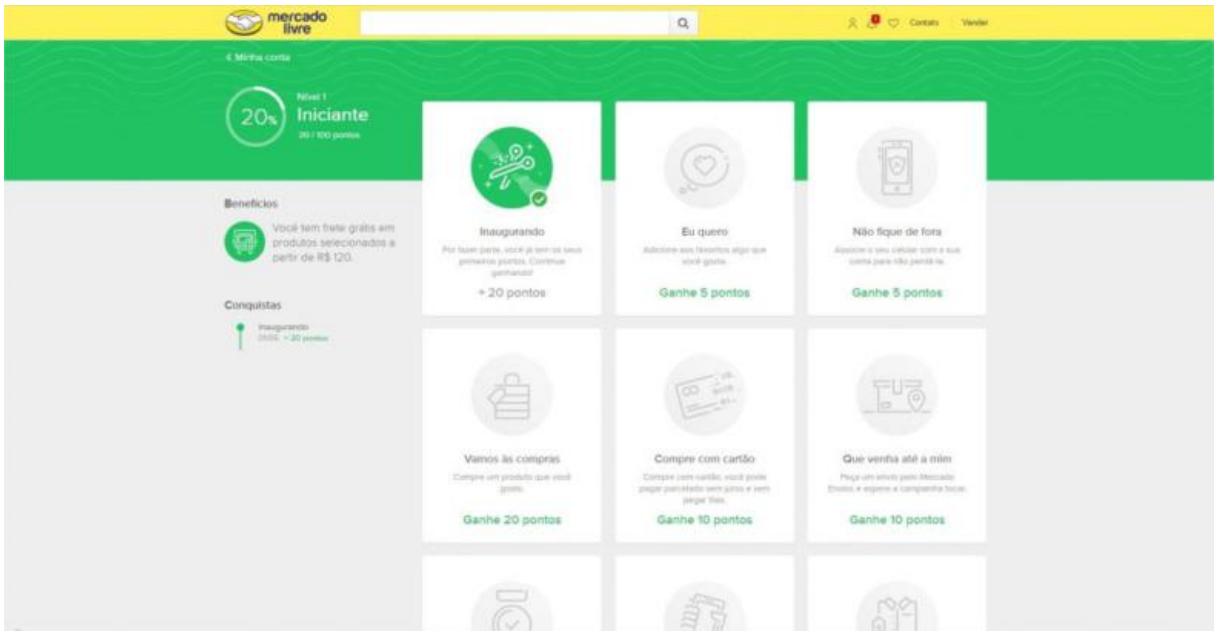

Figura 10 – Medalhas de recompensa do MercadoLivre

A aplicação de mapas Google Maps também utiliza reconhecimento e empoderamento para aumentar a confiabilidade das recomendações dos lugares disponíveis nos seus mapas. Cada usuário recebe uma pontuação diferente por comentário, avaliação e postagem de fotos de um local. Ao acumular pontos, o usuário exibe uma medalha no seu perfil e todos os outros usuários passam a reconhecê-lo como um guia local dependendo do seu nível. Foi dessa maneira que a Google engajou os usuários a comentar e escrever resenhas de estabelecimentos e localidades que já frequentaram.

4 IMPLEMENTAÇÃO

4.1 VALIDAÇÃO DAS PREMISSAS

Com base nas experiências pessoais dos autores, foram elaboradas algumas premissas que eram necessárias serem validadas para que o desenvolvimento do software fosse justificado. Desta forma, para guiar tal validação, foi utilizado o validation board , processo descrito na seção 3.1.

Para iniciar o procedimento, foi necessário definir o perfil de usuário do sistema. Assim, chegou-se aos perfis "alunos de computação da UFRJ que buscam uma plataforma para divulgar conhecimento", um perfil produtor, e "alunos de computação da UFRJ que buscam uma plataforma para buscar conhecimento", um perfil consumidor.

No início da segunda etapa foi primeiramente escolhido o perfil consumidor, dado que este foi considerado um perfil essencial para a validação do Motiva. Sendo assim, elaborou-se a seguinte hipótese de problema: "os alunos de computação buscam conteúdo fora das aulas", motivada pela premissa de que os alunos buscam se aprofundar em conteúdos fora da sala de aula, sendo em casa ou em seu deslocamento.

Definidas as hipóteses iniciais de cliente e problema, iniciou-se a etapa de experimentação. Para validar a hipótese de problema, foi elaborado um questionário a ser aplicado a alguns alunos de diferentes períodos do curso de ciência da computação da UFRJ.

O questionário foi aplicado para 44 alunos ativos ou formados, buscando-se sempre uma variedade de períodos entre os entrevistados. Foi conseguido obter pelo menos um representante de cada período, como ilustrado na figura 11, sendo os alunos que já estão além do 9º os mais representados nesse resultado. Foram realizados questionamentos que fossem possível extrair a informação de que "os alunos de computação buscam conteúdo fora das aulas". Para considerar um resultado positivo, foram estabelecidas metas de resposta para cada pergunta. A pergunta "De que maneira você complementa os seus estudos?" invalidaria a ideia se a opção "não complemento os meus estudos" possuísse mais de 20% das respostas. Se esta situação ocorresse, seria considerado que os alunos não buscam informação extra para seus estudos. Como mostrado na figura 12, apenas 6,8% não complementam os estudos. Para a pergunta "os conteúdos apresentados em sala de aula são suficientes para total aprendizado de um determinado assunto", foi definido que se 40% das respostas fossem mais próximas da opção "discordo", ou seja, as opções 1 e 2 de uma escala de 1 até 5, onde 5 é "concordo com a afirmação", se teria uma validação de que há a necessidade de conteúdo fora de sala de aula. Como mostrado na figura 13, 50% dos entrevistados responderam que discordam da afirmação apresentada. A última pergunta considerada necessária para validar a premissa foi "Procuro aprender com os meus colegas de faculdade". Em uma escala de 1 até 5, onde 5 é "concordo

Figura 11 – Resultado do questionário - período dos entrevistados

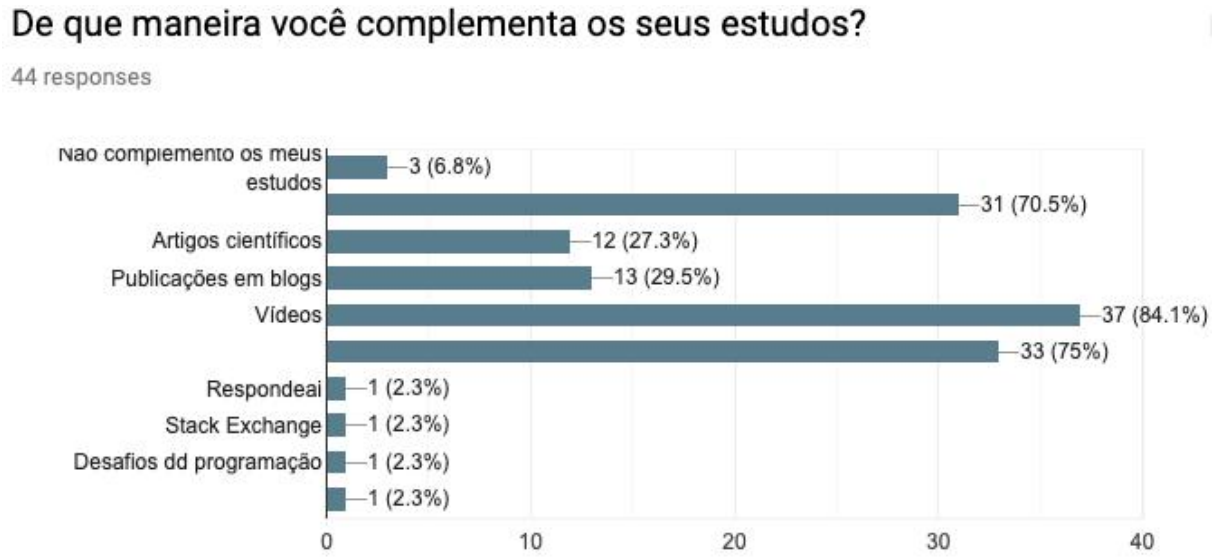

Figura 12 – Resultado do questionário - complemento de estudo fora da classe

com a afirmação", definiu-se que se 50% das respostas fossem mais próximas da opção "concordo", ou seja, as opções 4 e 5, teria-se a premissa "os alunos de computação buscam conteúdo fora das aulas" validada. Na figura 14 está ilustrado o resultado da pesquisa, onde 68,2% marcaram as opções 4 ou 5, concordando com a afirmação. Com o perfil consumidor validado, partiu-se para a validação do perfil produtor, definindo a hipótese de problema "os alunos de computação estão dispostos a escrever sobre um determinado assunto", motivada pela premissa de que os alunos estão dispostos a ensinar sobre um

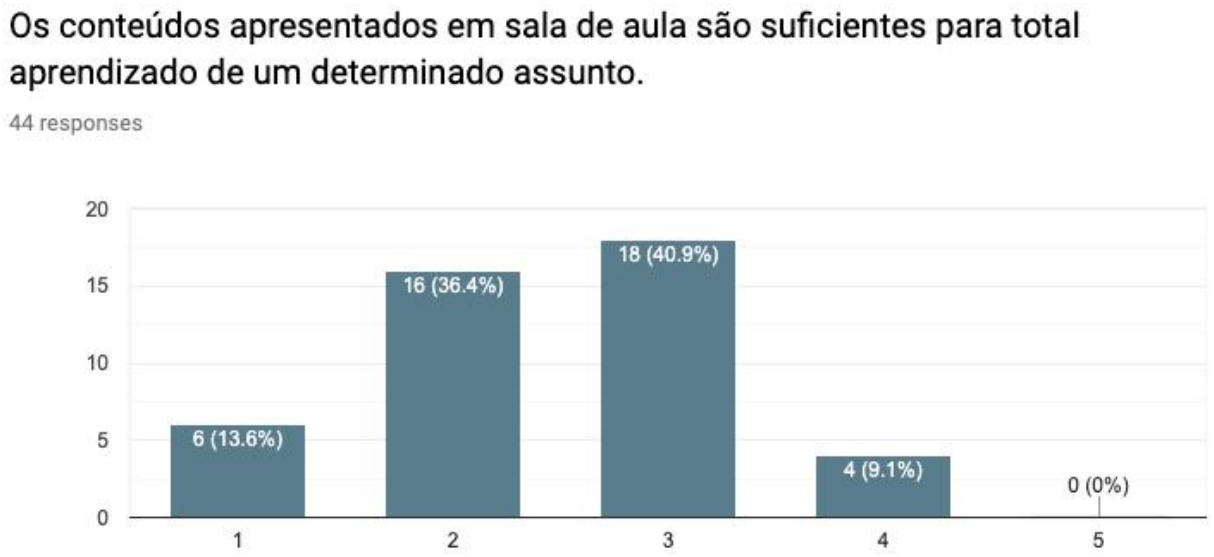

Figura 13 – Resultado do questionário - suficiência dos conteúdos apresentados em sala de aula

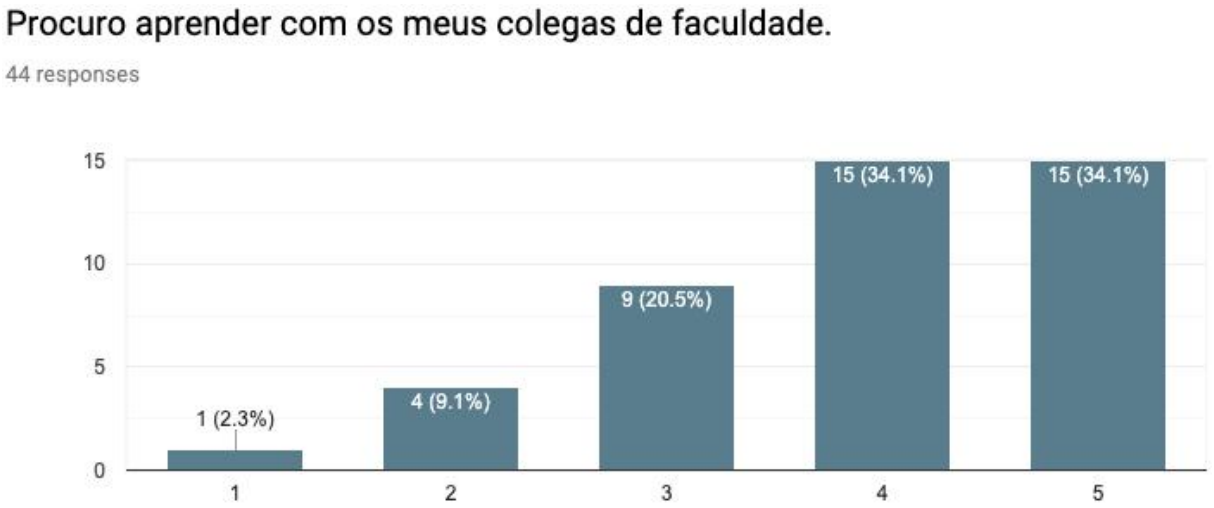

Figura 14 – Resultado do questionário - aprendizado com os colegas do curso

assunto de seu interesse.

Com as novas hipóteses de problema e cliente definidas, iniciou-se a etapa de experimentação. Foi aplicado um questionário para os mesmos 44 alunos da pesquisa anterior, buscando entender o perfil produtor de conteúdo.

Com a afirmação "gosto de compartilhar meus conhecimentos", buscou-se validar a ideia de que há alunos interessados em escrever conteúdo para os demais colegas. Foi definido que se 30% das respostas concordassem com a afirmação, ou seja, as opções 4 e 5 em uma escala de 1 até 5, onde 5 é "concordo com a afirmação", poderia-se avançar com o desenvolvimento da plataforma. Como mostrado na figura 15, 77,3% dos entrevistados responderam que concordam da afirmação apresentada. Validada a premissa de que há

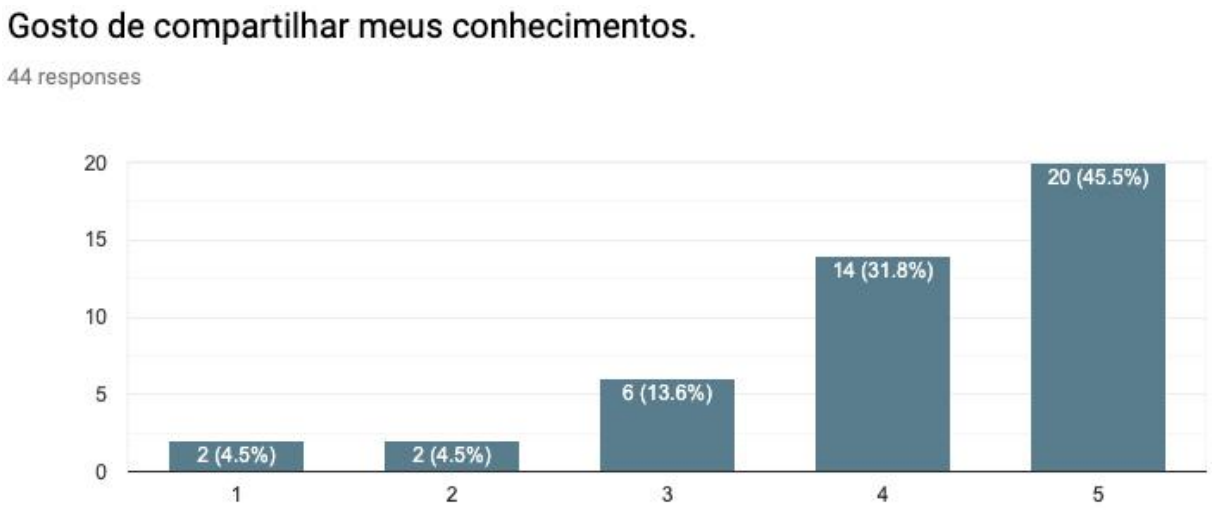

Figura 15 – Resultado do questionário - interesse em compartilhar conhecimento

produtores de conteúdo para a plataforma, buscou-se entender qual tipo de conteúdo seria gerado. Para tal entendimento, foi realizada a seguinte pergunta: "Que tipo de material é gerado durante os seus estudos?", com o objetivo de verificar a afirmação de que há conteúdo a ser publicado no Motiva. Em caso de 30% ou mais de respostas na opção "Não gero material durante meus estudos", tal premissa seria invalidada. Com o resultado de 11,4%, conforme figura 16, de respostas nesta opção, a premissa foi validada. Com 63,6% das respostas para a opção "resumos", pode-se ainda pensar na seguinte

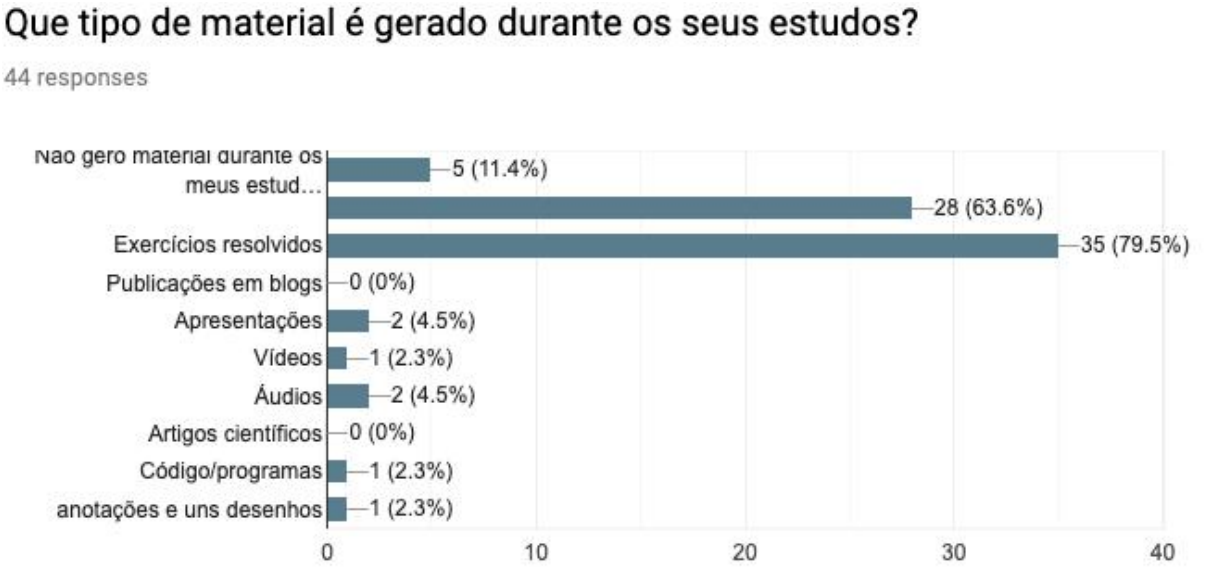

Figura 16 – Resultado do questionário - material gerado durante estudo

hipótese de problema: "os alunos de computação que produzem resumos em seus estudos buscam uma plataforma para armazenar o conteúdo gerado". Para validar esta hipótese, optou-se pelo desenvolvimento da plataforma Motiva e o acompanhamento do uso dela.

Tal validação não foi concluída pelo motivo de não ter havido um momento em que se pudesse testar o uso da plataforma.

4.2 DESENVOLVIMENTO DA PLATAFORMA

A decisão de qual o modelo de controle de versão a ser utilizado para o Motiva levou em consideração alguns aspectos da estruturação do projeto. Para que o projeto fosse portável para outras tecnologias, decidiu-se que haveria uma divisão entre as lógicas de armazenamento e de visualização de dados.

Por conta dessa divisão de responsabilidades, foi necessário criar dois projetos para serem gerenciados separadamente. Cada projeto possuiu suas particularidades ao longo do desenvolvimento, mas em ambos projetos foi adotado o sistema de controle de versão distribuído Git, sendo utilizado os servidores da empresa GitHub como repositório de origem.

A utilização dos serviços dessa empresa foi escolhida pelo fato de ser um dos mais utilizados na comunidade de desenvolvedores. Outro fator que também agregou para essa escolha foi pelo fato de possibilitar uma colaboração mais ampla, não limitando o desenvolvimento do projeto apenas para algumas pessoas, podendo qualquer usuário do GitHub poder solicitar autorização para colaborar com o desenvolvimento do Motiva.

4.2.1 Visualização dos Dados

Antes de se iniciar o desenvolvimento do projeto de visualização dos dados, foi preciso escolher a tecnologia a ser utilizada. Tinha-se como objetivo desenvolver uma solução de construção rápida, onde fosse possível utilizar a plataforma independentemente do dispositivo.

Desta forma, optou-se por desenvolver uma solução para ser utilizada via navegador, presente nos sistemas operacionais mais comuns da atualidade. Com esta restrição, buscou-se um framework que fornecesse facilidades no desenvolvimento de elementos visuais e que tivesse um bom desempenho ao processar as informações a serem apresentadas.

Somando-se às tais características, o framework a ser escolhida deveria ter como base uma linguagem já conhecida pelos autores deste trabalho, para que o tempo de desenvolvimento não se prolongasse demasiadamente. Sendo assim, definiu-se que o framework deveria ter como base a linguagem Javascript, ou suas variações, Python ou PHP.

Com todas as características e restrições definidas, buscou-se os frameworks comumente utilizadas pela comunidade de desenvolvedores. Dentre as soluções pesquisadas, a que mais agradou foi o framework Angular, que, além de possuir os requisitos já listados, é de fácil configuração e já havia sido amplamente utilizado por um dos autores.

4.2.2 Armazenamento dos Dados

Com o intuito de se deixar a separação de responsabilidade bem definida, optou-se por criar um projeto para o armazenamento de dados completamente separado do projeto de visualização de dados. Desta forma, foi feita uma análise de requisitos técnicos específica para as necessidades de armazenamento de dados.

O principal conceito do projeto de armazenamento dados é fornecer a armazenagem de dados e possibilitar o consumo destes dados. Tais funcionalidades devem ser independentes de qual a tecnologia a ser utilizada para acessar essa troca de informação. Esse conjunto de características está presente no conceito de uma REST API.

Sendo assim, a tecnologia a ser escolhida deveria apresentar afinidade com a arquitetura REST. Somando-se a isso, deve possuir um bom desempenho para processos assíncronos, ser de fácil implementação e configuração e, assim como na escolha da tecnologia para o projeto de visualização, deve ser uma linguagem já conhecida dos autores deste trabalho.

Após uma pesquisa sobre quais os frameworks que poderiam se adequar aos requisitos levantados, chegou-se ao framework Django. Mesmo sendo um framework simples, o Django é capaz de ser bastante escalável, além fornecer ferramentas que evitam que os desenvolvedores cometam falhas de segurança e ser bastante rápido nos processamentos.

Além de tais características, o Django é um framework que possui Python como linguagem base, linguagem previamente conhecida pelos autores, e é de fácil configuração e implementação. Somando-se tudo o que foi levantado, o Django se tornou uma boa escolha para o desenvolvimento do projeto de armazenamento de dados.

4.2.3 WebSocket

O WebSocket foi implementado no Motiva com o objetivo de trazer ao usuário a experiência de comunicação e interação em tempo real por meio de notificações, similar às diversas redes sociais atuais. A comunicação entre o servidor e os usuários é feita através de um canal que é estabelecido ao logar no sistema ou ao retornar com um login previamente salvo. A arquitetura abaixo ilustra como está estruturado o anúncio de mensagens na aplicação do sistema.

Quando um usuário realiza a operação de login – inserindo suas credenciais ou automaticamente – é criado um canal de comunicação bilateral onde o nome de usuário é o seu identificador. Isso significa que uma vez que o cliente esteja ativo no sistema, ele poderá receber e enviar mensagens utilizando WebSockets. Para que isso fosse possível a nível de implementação, utilizamos a biblioteca Django Channel no backend e a biblioteca WS no frontend.

Também é necessário persistir os canais de comunicação criados e gerenciar esses dados de uma maneira performática. Para isso, utilizamos o Redis, que é um banco de dados não

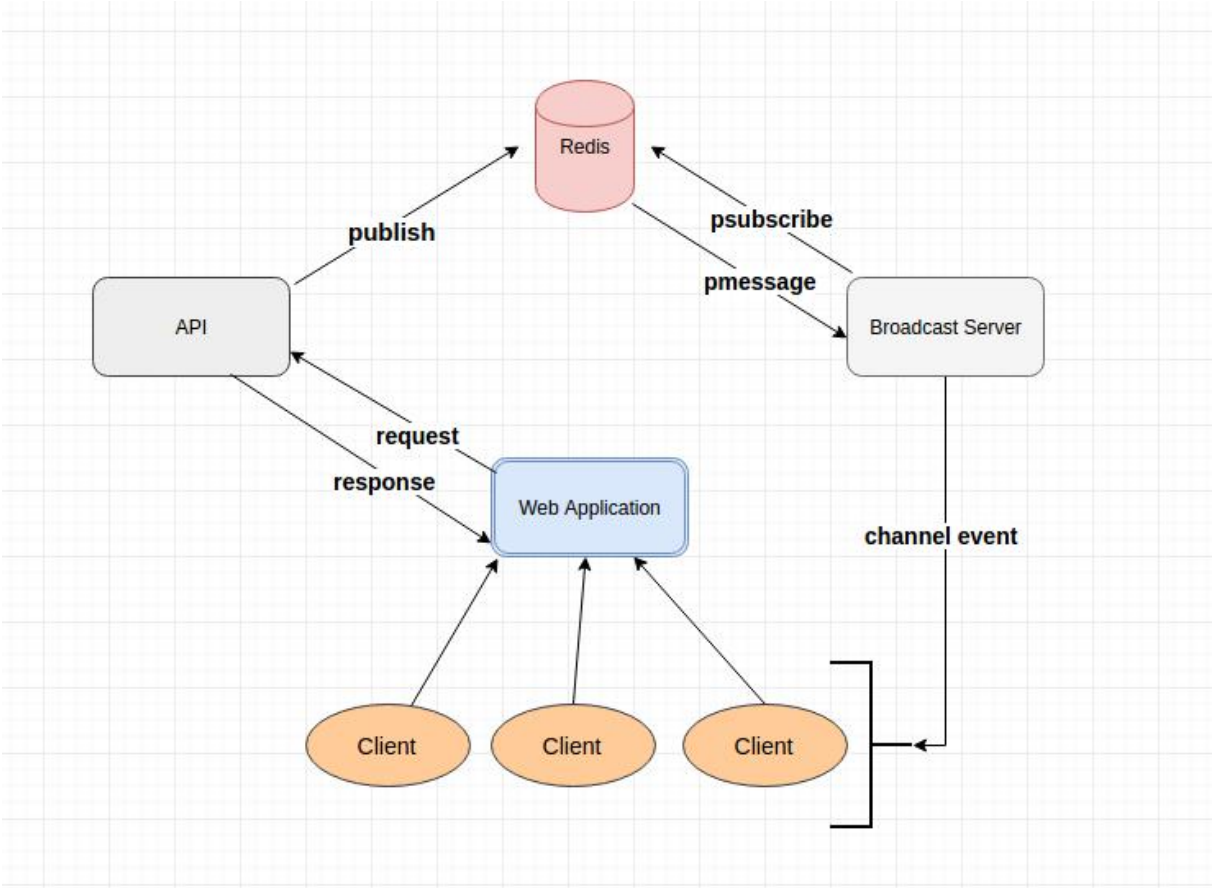

Figura 17 – Arquitetura de WebSockets do Motiva

relacional (NoSQL) que armazena dados em estrutura de pares chave-valor em memória. Uma vez que o canal de comunicação é criado pelo Django Channel, seus identificadores são armazenados no banco de dados do Redis.

Uma implementação mais simples é a ação de gostar de uma postagem. Quando um leitor clica no botão gostar, uma requisição HTTP é enviada com a ação de gostar. Quando o banco de dados atualiza com sucesso a contagem de *likes*, uma mensagem é disparada pelo servidor para o autor da postagem através do canal de comunicação WebSocket com seu nome de usuário. Essa mensagem contém a ação de notificação, a descrição de que foi realizada uma ação de gostar e o nome do leitor que gostou da publicação. A biblioteca WS se encarrega de receber essa mensagem e, através de um indicador de ações, perceber que é uma notificação e exibi-la para o usuário autor.

A ação de inserir um comentário em uma postagem é uma implementação mais abrangente. A regra utilizada é a seguinte: o autor, todos os usuários que gostaram de uma publicação e todos os usuários que comentaram na mesma receberão uma notificação caso um comentário seja feito nessa postagem. Essa comunicação ocorre um pouco diferente da anterior. Uma vez que a postagem seja criada, um canal com a identificação da postagem é criado e seu autor é inscrito nesse canal, da mesma forma que posteriormente os leitores

que gostarem da publicação também serão inscritos no canal da publicação. Quando um usuário faz um comentário, caso ele ainda não tenha gostado da postagem do autor, ele também é inscrito no canal de comunicação da publicação e sua ação de comentar anuncia para todos os participantes – exceto ele mesmo – o seu comentário.

Caso um usuário desconecte, ou seja, feche a sessão ativa no navegador com o sistema, seu canal é removido do Redis e nenhuma mensagem será enviada até que o cliente se conecte novamente. Esse gerenciamento é feito para que nenhuma comunicação seja feita de forma desnecessária, visando a melhor performance do sistema.

4.2.4 Integração das Tecnologias

Para que o sistema tivesse um progresso mais completo, onde em todas as entregas houvesse uma parte do sistema completa e usável, optou-se por desenvolver os projetos de visualização e armazenamento de dados simultaneamente. Com esta decisão, foi necessário encontrar uma forma de padronizar a comunicação entre os projetos.

Como a relação entre os desenvolvedores era bem próxima, decidiu-se inicialmente estabelecer uma comunicação informal para o alinhamento entre os desenvolvedores de como a troca de informação entre os projetos deveria ocorrer. Nessa abordagem ocorreram alguns problemas, como mudança de parâmetros sem nenhum aviso, adição de parâmetros desnecessários assim como a remoção de outras propriedades que eram essenciais para o funcionamento do sistema como um todo.

Desta forma, buscou-se uma forma um pouco mais formal e com processos de validação bem definidos para que se evitasse um retrabalho ou mal funcionamento do sistema. Dentre todas as possibilidades encontradas, chegou-se à conclusão de que era necessário conseguir simular o sistema de armazenamento de dados, simulando não só os modelos de dados como também a comunicação com o projeto de visualização, utilizando a mesma arquitetura REST API.

Durante o processo de tentar simular esse ambiente com um baixo custo de implementação, foi identificada a plataforma online apiary. Nela é possível criar uma REST API de modo simples, escrevendo apenas um documento de texto onde declara-se os serviços disponíveis e seus respectivos atributos de entrada e saída. Sendo assim, foi possível padronizar os formatos de entrada e saída de dados em um único arquivo, disponível para todos os desenvolvedores.

Resolvido o problema de padronização de troca de informação, ainda havia uma falta de comunicação de mudanças realizadas. Como resolução para este problema, ficou estabelecido que para toda modificação deveria ser criada uma tarefa de validação associada ao desenvolvedor não responsável pela modificação. Desta forma, todos os desenvolvedores envolvidos no projeto ficam cientes de todas as modificações, assim como se responsabilizam pela coesão e coerência do sistema.

4.3 GAMIFICAÇÃO

A Gamificação foi implementada no Motiva com a missão de engajar os usuários e estabelecer uma competição saudável a fim de tornar o aprendizado e o compartilhamento de conhecimento mais divertido. A mecânica do jogo é composta por níveis e pontos de experiência, além de recompensas que visam incentivar tanto a participação como autor de postagens, assim como também a leitura de postagens.

Todos os usuários – que serão tratados como jogadores – quando ingressam no sistema, começam no nível 0, que é chamado Pré-Calouro. Para subir de nível, é necessário acumular pontos de experiência, que são dados como recompensa de acordo com determinadas ações de autor e leitor. Por exemplo, para obter pontos de experiência como autor é necessário ou publicar uma postagem e ganhar pontos imediatamente, ou ganhar pontos com curtidas em postagens publicadas. Como leitor, é possível obter pontos de experiência lendo postagens.

Especificamente, quando uma postagem é publicada, o autor ganha dez pontos de experiência. Para cada curtida em sua publicação, dois pontos de experiência. Como leitor, vale ressaltar que cada texto possui um tempo médio de leitura e, caso o usuário passe mais da metade desse tempo lendo um artigo, um ponto de experiência é obtido. O objetivo é motivar não somente o compartilhamento de conhecimento, mas também incentivar o reconhecimento pelo esforço e incentivar a leitura de diversos temas visando um conhecimento abrangente.

Em conjunto com os pontos de experiência, existem as recompensas de desafios, que visam motivar ainda mais os jogadores e fazer com que os mesmos compitam entre si e melhorem seus trabalhos. Por exemplo, entre várias, existem recompensas para postagens com mais de 100 curtidas e para leitores que conseguirem ler pelo menos um artigo por dia em uma semana. Esses tipos de recompensas incentivam a qualidade das publicações e o ato de criar uma rotina de estudos, afetando os autores e também os leitores.

A mecânica do jogo consiste em sempre incentivar a publicação de postagens, uma vez que os autores ganham mais pontos de experiência do que os leitores. O jogo proposto pela Gamificação não possui fim, ou seja, não possui nível máximo e torna necessário que sempre existam novas postagens para que os jogadores consigam atingir níveis maiores tanto como autores e como leitores.

4.4 GERENCIAMENTO DO DESENVOLVIMENTO

4.4.1 Implementação do Scrum

O desenvolvimento do Motiva teve algumas características que tornaram o gerenciamento do projeto uma tarefa distante do que é comumente encontrado no desenvolvimento de outros softwares. Desta forma, a aplicação do Scrum conforme a literatura não foi pos-

sível de ser realizada, sendo necessária algumas adaptações para que o projeto tivesse continuidade.

4.4.1.1 As sprints

A principal adaptação, e possivelmente a de mais impacto sob a evolução do projeto, foi realizada no conceito de sprint. Como definido na literatura, as sprints devem possuir intervalos regulares onde ao final deste intervalo deve haver parte do produto a ser entregue. A definição do que deve ser entregue ao final de cada sprint é realizada numa reunião, a Sprint Planning, antes do início da sprint.

Devido ao fato dos desenvolvedores possuírem outras tarefas de grande importância, que afetam a disponibilidade dos mesmos para o projeto, não foi possível seguir exatamente como uma sprint deveria ser. Desta forma, pensou-se em duas maneiras de se implementar as sprints.

A primeira opção seria manter os períodos, em dias, de término da sprint. Nesta maneira haveriam sprints em que nada de valor seria entregue, já em alguns momentos não se teria tempo disponível para o desenvolvimento e, assim, os dados de velocidade do time e de previsão de entrega seriam muito pouco confiáveis.

A segunda opção seria considerar o período de sprint de acordo com o tempo disponível para o desenvolvimento. Desta forma, ao final de cada sprint se teria algo de valor entregue e uma confiabilidade maior quanto a velocidade do time em relação ao tempo disponível de desenvolvimento.

Mesmo possuindo uma falta de confiabilidade quanto à previsão de entrega, a segunda opção se mostrou ser mais coerente com os objetivos da metodologia Scrum. Desta forma aplicou-se a segunda opção, onde em cada sprit planning se definia também o dia em que a sprint terminaria, de acordo com o tempo disponível do time.

4.4.1.2 O Product Owner

Segundo a literatura, o Product Owner é a pessoa que deve entender do produto que está sendo criado e o valor de negócio entregue em cada tarefa. Na implementação do Motiva, uma pequena mudança para este papel foi necessária.

Seria fundamental escolher uma pessoa da equipe que possuísse conhecimento do valor do produto para os diversos tipos de usuário que o sistema tem. Concentrar esse entendimento em uma pessoa apenas, em uma equipe onde há apenas duas pessoas, traria um custo de tempo muito elevado, prejudicando a velocidade do time.

Desta forma, decidiu-se dividir essa responsabilidade em duas pessoas, de modo em que houvesse um entendimento melhor de todas as particularidades do projeto mas sem ter um alto custo. Sendo assim, foram definidos dois perfis de Product Owner. Um com

o conhecimento mais técnico de como o sistema está sendo desenvolvido e outro com um entendimento melhor de qual o valor a ser entregue pelo software.

Para executar o papel de P.O., as duas pessoas realizavam reuniões de tempos em tempos para definir e priorizar as tarefas a serem realizadas. Desta forma, se chegava a um consenso entre as partes e a pessoa com os conhecimentos mais técnicos repassava tais prioridades nas Sprint Planning do time de desenvolvimento.

5 TRABALHOS FUTUROS

Em diversas áreas da computação, muitas vezes é necessário um projeto completo para que possamos apenas desenvolver e testar pequenas hipóteses. O Motiva é um projeto estruturado com uma arquitetura que facilita o entendimento e a manutenção do seu código, tanto frontend quanto backend, justamente para que possa ser reutilizado ou incrementado no futuro. Considerando o fato citado, este capítulo visa propor alguns trabalhos futuros a quem possa interessar e não deseje utilizar recursos para implementar um sistema completo para validar sua hipótese ou simplesmente desenvolver uma aptidão prática em alguma área voltada para engenharia de software.

5.1 SISTEMAS DE RECOMENDAÇÃO

As postagens possuem identificadores que são propostas pelos seus autores de acordo com o conteúdo da publicação. Cada usuário tem sua preferência de tópicos para leitura e produção de conteúdo. Atentando a esse fato, estudantes interessados em Inteligência Artificial podem propor um estudo de sistemas de recomendação com predição de publicações baseadas no gosto que cada usuário possui por determinado assunto.

5.2 APLICAÇÃO MOBILE

O Motiva possui seu backend estruturado como uma API REST que pode ser consumida por qualquer tipo de frontend. Dessa forma, é possível propor a construção de aplicações mobile Android e iOS nativas, assim como aplicações híbridas com React Native e Flutter. Esse tipo de projeto poderia auxiliar pessoas que querem evoluir como desenvolvedor de aplicações móveis, designer de experiência de usuário e interfaces gráficas e até papéis voltados para a visão de produto como Product Owner e Scrum Master.

5.3 TESTES

Como um conceito de produto, o sistema necessita de testes para manter a qualidade da aplicação, ou seja, para garantir que todas as funcionalidades estão sendo executadas de maneira correta. É possível implementar todos os tipos de testes, de acordo com o conceito de pirâmide de testes, como testes unitários, testes end to end, testes de interface e testes de integração. Além disso, também é possível trabalhar com o desenvolvimento de cenários de testes com escrita funcional orientada ao comportamento (Behaviour Driven Development).

5.4 GAMIFICAÇÃO

A gamificação no Motiva é como um jogo sem fim, justamente porque não possui o conceito de nível máximo. Sendo assim, conforme os usuários vão colaborando e crescendo junto com o sistema, é necessário propor novos desafios, novos níveis e novas recompensas com a finalidade de manter a competitividade e motivação dos usuários.

5.5 ANÁLISE DE REDES SOCIAIS

A estrutura da aplicação pode ser vista sob a ótica de uma rede social. Partindo do princípio que cada usuário é um nó de uma rede social, é possível fazer análises sobre a influência de cada usuário para um determinado tópico, o quanto um tópico pode conectar usuários, assim como quanto uma postagem e comentários podem fazer o mesmo, o desempenho de um aluno de acordo com a sua participação no Motiva, entre outras diversas análises.

6 CONCLUSÃO

O Motiva não foi criado apenas sob a ótica de um projeto de conclusão de curso. Além de reunir competências do curso de Bacharelado em Ciência da Computação, o sistema forma um produto moderno – como uma rede social – que pode agregar valor ao curso se implementado em produção, como também ajudar os futuros profissionais com os trabalhos propostos.

Como produto, foi pensado com a possibilidade de atingir não só os alunos, mas também os professores, grupos de extensão e grupos de estudo. Existem alguns problemas que o Motiva pode solucionar, como a comunicação no curso, a separação, exposição e atualização dos materiais de estudo durante a graduação. Seu formato de rede social, junto com a gamificação, propõe interação contínua de forma motivadora para os usuários do sistema, fazendo com que todos cresçam academicamente e profissionalmente simplesmente consumindo e gerando conteúdo.

Existem casos na graduação em que os debates e o conhecimento são privados, propositalmente ou não, e esse tipo de situação priva alguns de uma possível evolução. Para o curso, o sistema deixa um legado de motivação à implementação de uma forma de comunicação simples, direta, transparente e em tempo real, e também a possibilidade de propor e elevar o nível de discussões acadêmicas, fazendo com que cada vez mais pessoas possam participar das mesmas.

Uma das principais razões de reclamações de alguns discentes é a falta de conteúdo prático em algumas disciplinas que são agravadas pelo excesso de teoria das mesmas. Dessa forma, o sistema foi pensado para também incentivar discussões que não sejam diretamente acadêmicas e a formação de novos grupos com a finalidade de surgirem projetos práticos paralelos ao curso de Ciência da Computação, visto que parte dos alunos não visam seguir a carreira acadêmica. Com o mesmo pretexto, foram expostas ideias de trabalhos futuros que utilizariam o próprio Motiva como oficina prática.

Desde sempre, o Motiva foi criado sob a ótica de um sistema aberto e incremental, com a ideia de que qualquer pessoa que esteja apta a ajudar possa fazê-lo. Levá-lo à implementação e produção é um trabalho conjunto para quem possuir disponibilidade e aspiração de desenvolver e aperfeiçoar o curso de Bacharelado em Ciência da Computação da Universidade Federal do Rio de Janeiro.

REFERÊNCIAS

CHACON, S. **Pro Git**. Apress, 2010. ISBN 9781430216803. Disponível em: <https://books.google.com.br/books?id=8qsXogEACAAJ>.

CHOU, Y. kai. **Actionable Gamification: Beyond Points, Badges, and Leaderboards**. [S.l.]: Octalysis Media, 2015. ISBN 1511744049.

FETTE, I.; MELNIKOV, A. **The WebSocket Protocol**. 2011. <https://www.rfc-editor.org/info/rfc6455>. Accessado: 13 de Abril de 2019.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 76–106 p. Tese (Dissertação de Doutorado), University of California, Irvine, CA, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a Internet: uma abordagem top-down - 6ª edição**. [S.l.]: Pearson Education do Brasil, 2013. ISBN 978-85-430-1443-2.

Lean Startup Machine. **Validation Board - Free tool for testing startup ideas, stop wasting time and money**. 2014. <https://www.leanstartupmachine.com/validationboard/>. Accessado: 07 de Março de 2019.

MARTIN, R. C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. Boston, MA: Prentice Hall, 2017. (Robert C. Martin Series). ISBN 978-0-13-449416-6. Disponível em: <https://www.safaribooksonline.com/library/view/clean-architecture-a/9780134494272/>.

MASSé, M. **REST API Design Rulebook**. Sebastopol, CA: O'Reilly Media, Inc., 2012. ISBN 978-1-449-31050-9.

RIES, E. **The lean startup : how today's entrepreneurs use continuous innovation to create radically successful businesses**. New York: Crown Business, 2011. ISBN 0307887898.

SUTHERLAND, J. **Scrum: A arte de fazer o dobro de trabalho na metade do tempo**. LeYa, 2016. ISBN 978-8-544-10451-4. Disponível em: <https://www.amazon.com.br/Scrum-Fazer-Dobro-Trabalho-Metade/dp/8544104517/ref=sr_1_1?__mk_pt_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&keywords=scrum&qid=1551800185&s=gateway&sr=8-1>.