UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

RUBEN BLENICIO TAVARES SILVA

CONSTRUÇÃO DE UM SERVIÇO WEB PARA PROMOVER A INTEGRAÇÃO DE FERRAMENTAS DE CRIATIVIDADE

QUIXADÁ – CEARÁ

2016

RUBEN BLENICIO TAVARES SILVA

CONSTRUÇÃO DE UM SERVIÇO WEB PARA PROMOVER A INTEGRAÇÃO DE FERRAMENTAS DE CRIATIVIDADE

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Orientador: Prof. Me. Camilo Camilo Almendra

QUIXADÁ – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S583c Silva, Ruben Blenicio Tavares.
Construção de um serviço web para promover a integração de ferramentas de criatividade / Ruben Blenicio Tavares Silva. – 2016.
51 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Me. Camilo Camilo Almendra.

1. Engenharia de software. 2. Engenharia de requisitos. 3. Software - Desenvolvimento. I. Título.
CDD 005.1

RUBEN BLENICIO TAVARES SILVA

CONSTRUÇÃO DE UM SERVIÇO WEB PARA PROMOVER A INTEGRAÇÃO DE FERRAMENTAS DE CRIATIVIDADE

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Me. Camilo Camilo Almendra (Orientador)
Campus Quixadá
Universidade Federal do Ceará – UFC

Prof. Me. Bruno Góis Mateus
Campus Quixadá
Universidade Federal do Ceará - UFC

Prof. Me. Carlos Diego Andrade de Almeida
Campus Quixadá
Universidade Federal do Ceará - UFC

A Deus.

Aos meus pais, Benedito e Consuelo.

Aos meus irmãos e amigos.

AGRADECIMENTOS

Agradeço primeiramente à Deus que sempre me ajudou nesta jornada, me concedendo sabedoria e graça pelo que sou eternamente grato.

Aos meus pais que me incentivaram e me apoiaram em todo momento, através dos quais cheguei até aqui.

Ao Prof. Me. Camilo Camilo Almendra, que dedicou seu tempo e conhecimento para nos orientar.

Aos professores participantes da banca examinadora Prof. Me. Bruno Góis Mateus e Prof. Me. Carlos Diego Andrade de Almeida pelo seu tempo e conselhos.

Aos meus colegas de graduação Allef Araujo Lobo e Robson Cavalcante de Negreiros, por sua colaboração e disponibilidade para nos ajudar.

"Porque o Senhor dá a sabedoria, e da sua boca
vem o conhecimento e o entendimento."

(Provérbios 2.6)

RESUMO

Técnicas de criatividade são frequentemente utilizadas por inúmeras organizações e existe um numero considerável de técnicas criativas para diversos propósitos. Algumas técnicas de criatividade também são apoiadas por ferramentas computacionais, chamadas ferramentas de criatividade, entretanto não se encontram grandes esforços no sentido de integrar estas ferramentas. A grande motivação da integração de ferramentas de criatividade se encontra no fato de que isto poderia facilitar o compartilhamento de informações entre aplicações distintas mas que implementam uma mesma técnica criativa. Este trabalho tem como objetivo desenvolver uma maneira de integrar ferramentas de criatividade de forma rápida e consistente. Foram escolhidas duas técnicas de criatividade, sendo elas o *Brainwriting* e *Ideia advocate*, para cada técnica foram identificados cenários de uso e foi definido um modelo de dados. Foi desenvolvido um serviço web para promover a integração de ferramentas de criatividade e em seguida foi realizado um estudo de caso para validar o serviço através da avaliação do uso do mesmo por duas ferramentas que implementam as técnicas do *Brainwriting* e *Ideia advocate*, essas ferramentas foram criadas em dois trabalhos de conclusão de curso e serviram para avaliar o nosso serviço.

Palavras-chave: Técnicas de Criatividade. Ferramentas de Criatividade. Integração.

ABSTRACT

Creativity techniques are often used by numerous organizations and there are a considerable number of creative techniques for various purposes. Some creativity techniques are also supported by computational tools, called creativity tools, but there are no great efforts to integrate these tools. The great motivation of the integration of creativity tools lies in the fact that this could facilitate the sharing of information between different applications but implementing the same creative technique. This work aims to develop a way to integrate creative tools quickly and consistently. Two techniques of creativity were chosen: Brainwriting and Idea advocate; for each technique, usage scenarios were identified and a data model was defined. A web service was developed to promote the integration of creativity tools, followed by a case study to validate the service through the evaluation of its use by two tools that implement the techniques of Brainwriting and Ideia advocate. These tools were created In two course completion work and served to evaluate our service.

Keywords: Creativity Techniques. Creativity Tools. Integration.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1 INTRODUÇÃO

Técnicas de criatividade são utilizadas por diversas organizações, com o intuito de obter produtos de qualidade, que se aproximem das expectativas do cliente. No ambiente empresarial, é imprescindível que os produtos gerados tragam consigo algo de inédito e inovador em relação objetivo para o qual foram construídos, para que possam competir no mercado (MORAES, 2014).

Em mercados com alto nível de competitividade, pequenos diferenciais no produto que introduzam algo inovador no modelo de negócio, fazem diferença em relação ao sucesso da organização (MORAES, 2014). A criatividade é um fator que pode determinar o sucesso das organizações, muitas empresas têm investido em criatividade para se contrapor às dificuldades, sobretudo através do uso de Técnicas de criatividade, visando, assim, utilizar toda a capacidade criativa das pessoas (RODRIGUES, 2009). Profissionais de diversas áreas como, design, publicidade e tecnologia, têm obtido sucesso ao adotar Técnicas Criativas como uma alternativa de geração de ideias inovadoras (PLENTZ, 2011).

Na Engenharia de Requisitos, as Técnicas de criatividade atuam como facilitadoras para obtenção de requisitos mais completos. Consequentemente, esses requisitos darão origem a produtos de software coesos, inovadores, e que melhor atendem às especificações dos stakeholders (NGUYEN; SHANKS, 2009).

Embora sejam amplamente difundidas e utilizadas tanto no campo acadêmico como no mercado, existem algumas barreiras relacionadas à seleção e emprego das Técnicas criativas. A literatura por sua vez apresenta uma ampla coleção de Técnicas de criatividade, no entanto faltam padrões para classificar as técnicas de acordo com suas especificidades, provocando muita incerteza na eleição da técnica mais apropriada para determinado contexto. Não obstante, a bibliografia nem sempre apresenta as Técnicas de criatividade da mesma forma, alguns autores procuram ser mais detalhistas e introduzem pequenas variações nas técnicas, adaptando-as a um contexto de aplicação específico (PLENTZ, 2011).

Algumas obras procuram aplicar uma taxonomia às Técnicas criativas, entretanto enfocam mais na perspectiva da técnica em si, e não levam em conta o contexto de uso. Por este motivo, muitos gerentes têm receio de adotar Técnicas de criatividade nas suas empresas, visto a dificuldade em selecionar as técnicas com base nas particularidades de cada organização (PLENTZ, 2011).

Uma solução possível para resolver este problema seria o uso das Ferramentas de

criatividade, elas incorporam a metodologia adotada nas Técnicas de criatividade e são voltadas para as necessidades das pessoas. Além disso, essas ferramentas guiam o processo de geração e/ou análise de ideias, para que seus usuários não precisem necessariamente ter experiência na técnica adotada pela ferramenta.

Para fins de comodidade, existe a possibilidade de querer conectar duas ou mais Ferramentas de criatividade, de forma que as ideias sejam geradas em uma, e avaliadas em outra, por exemplo. Novamente, para uma mesma técnica devem existir diversas ferramentas que a implementam, e pode surgir a necessidade de conectar as mesmas. De uma perspectiva parecida, uma mesma ferramenta pode ter várias versões em diferentes plataformas, e é necessário integrá-las. Percebe-se então um problema de interoperabilidade, é preciso prover meios para integrar essas ferramentas de forma que os dados transitem entre elas de forma consistente. Interoperabilidade é um atributo de qualidade que se relaciona ao fato de como vários sistemas podem realizar a troca de informações de forma que haja tanto consistência dos próprios dados compartilhados (sintaxe), como do significado dos dados (semântica) (BASS; CLEMENTS; KAZMAN, 2012).

Integrar diferentes Ferramentas de criatividade não é uma tarefa trivial, é preciso ter em vista que as características de cada Técnica de criatividade definem o tipo de informação gerado pela Ferramenta de criatividade correspondente. Em segundo lugar, as restrições tecnológicas também devem ser consideradas, pois existe uma grande diversidade de plataformas e sistemas operacionais, sendo necessário adotar uma linguagem em comum, para possibilitar a comunicação entre as Ferramentas de criatividade de diferentes ambientes.

Várias abordagens de arquitetura de software se propõem a resolver o problema da falta de interoperabilidade entre sistemas. Uma delas é a arquitetura orientada a serviços, que orquestra um sistema de software de forma que cada macro funcionalidade atua como um serviço independente. A arquitetura orientada a serviços também facilita a integração entre dois ou mais sistemas (SCHREIER, 2011). No presente trabalho adotamos a arquitetura orientada a serviços para prover a integração entre as Ferramentas de criatividade.

Este trabalho tem como objetivo projetar e implementar um serviço que irá padronizar a comunicação e facilitar o compartilhamento de informações geradas em diferentes Ferramentas de criatividade, para tanto, serão selecionadas algumas técnicas de criatividade e sucederá o desenvolvimento do serviço, por fim será realizado um estudo de caso através da integração do serviço com duas ferramentas de criatividade, com o objetivo de validar o mesmo.

2 TRABALHOS RELACIONADOS

A arquitetura orientada a serviços é útil quando se quer ter um sistema de software com alta interoperabilidade (Sá, 2016). Sá (2016) realizou a migração de um sistema web monolítico para uma arquitetura orientada a serviços visando obter uma arquitetura mais decomposta, o que diminui a replicação de código-fonte e dinamiza a comunicação entre as entidades do sistema. O autor modelou uma API orientada a serviços para o novo sistema e separou o mesmo em módulos – serviços – que compreendem uma funcionalidade completa. Sá (2016) utilizou o modelo REST juntamente com o framework web Spring MVC para implementar os serviços.

Neste trabalho, devido ao sucesso obtido por Sá (2016), também adotaremos o modelo REST e o framework Spring MVC. Nosso objetivo, entretanto, não é realizar uma migração de sistema web monolítico para uma arquitetura orientada a serviços, mas desenvolver um novo serviço.

Lampkowski e Silva (2014), em seu trabalho, tentaram avaliar a importância da interoperabilidade entre sistemas através um estudo de caso com dois sistemas web da Universidade Estadual Paulista (UNESP), o SGAd responsável pela gestão administrativa, e o SISOFC, responsável pela gestão financeira e orçamentária da universidade. Eles identificaram dependências entre os dois sistemas e elencou situações em que as informações poderiam ser reutilizadas. Lampkowski e Silva (2014) definiram os serviços que poderiam ser disponibilizados e mapearam os mesmos em verbos HTTP de acordo com o tipo de operação realizada, dessa forma as informações dos sistemas poderiam ser compartilhadas mais facilmente. Neste trabalho será desenvolvido um serviço para integrar Ferramentas de Criatividade de forma a compartilhar dados entre elas.

Rosa (2016), desenvolveu e avaliou um método para o desenvolvimento de software baseado em microserviços. Uma das questões levantadas no trabalho é que, com o advento da computação em nuvem as aplicações orientadas a serviços ganharam força, isso se deve dentre outros fatores, a algumas limitações impostas pela arquitetura monolítica, tornando-a inadequada para grandes aplicações que demandam uma grande performance e disponibilidade. Surge então, a necessidade de se estabelecerem padrões para que as aplicações orientadas a serviços não se tornem apenas um amontoado de código-fonte o que dificulta a manutenção do software, tornando-o propenso a reescrita desnecessária. Por isso o autor desenvolveu um método formado por um conjunto de passos que objetivam guiar o processo de desenvolvimento

de microserviços de forma a maximizar a manutenibilidade e a interoperabilidade do sistema como um todo. Nesse trabalho nós iremos executar alguns passos do método definido por Rosa (2016), no que diz respeito à definição de funcionalidades do serviço.

Os trabalhos citados acima enfatizam a importância da interoperabilidade e fornecem algo concreto para a construção do serviço de integração de ferramentas de criatividade, atualmente existem algumas dessas ferramentas que dão suporte a Técnicas de Criatividade, com o objetivo apoiar a geração de ideias e facilitar a colaboração entre uma equipe.

A ferramenta Realtime Board é um sistema presente nas versões web e mobile, que consiste basicamente em um quadro branco onde os usuários podem afixar desde notas, documentos, imagens, entre outros tipos. A ferramenta também dá suporte a comentários nos itens adicionados. A ferramenta Realtime Board tem como resultado uma árvore de itens que podem ser uma ideia ou uma categoria de ideias, cada item pode conter também uma lista de comentários (REALTIMEBOARD, 2016).

HatParty é uma ferramenta de criatividade que usa elementos de gamificação para apoiar a geração de ideias criativas. Ela adota o cenário de uma corrida de cavalos, onde cada participante é um corredor. O objetivo é inicialmente gerar o maior número possível de ideias; depois vem uma fase de seleção de ideias, quando ideias absurdas ou impraticáveis são descartadas; e por último as ideias são priorizadas umas em relação as outras, formando assim um ranking de ideias. A ferramenta HatParty gera uma lista de ideias ranqueadas de forma decrescente, começando com as ideias que foram melhores avaliadas (COLLAGE, 2016).

Creative Leaf é uma ferramenta de criatividade que apoia a geração de requisitos. A ferramenta Creative Leaf engloba grande parte do processo criativo, desde a modelagem do problema, passando pela geração e análise de ideias, chegando à consolidação e implementação das ideias. Ela foi criada com o intuito de concentrar grande parte do conhecimento da comunidade de engenharia de software para ajudar a entender as necessidades do cliente e obter requisitos de qualidade. A ferramenta Creative Leaf possui uma paleta com alguns itens predefinidos, sendo eles: objetivo, tarefa, objetivo fraco, recurso, ator, ideia e hipótese. O resultado final gerado pela ferramenta consiste em uma estrutura parecida com uma árvore composta por uma combinação desses itens, onde está descrita a modelagem do problema, e ideias para solucioná-lo (LANGUAGES, 2016).

Para o contexto de Ferramentas de criatividade não foram encontrados trabalhos

significativos no sentido de promover a integração entre as mesmas, por isso este trabalho tem foco em dar suporte a essas ferramentas. Esse suporte se dará por meio de um serviço web capaz de integrar diferentes Ferramentas de Criatividade permitindo compartilhar os dados de forma segura e consistente.

3 FUNDAMENTAÇÃO TEÓRICA

A integração entre Ferramentas de Criatividade pode ser útil para facilitar o processo de geração, avaliação e implementação de ideias criativas. Nesta seção serão apresentados os principais conceitos que serão utilizados no decorrer deste trabalho, a saber: Técnicas de Criatividade – uma visão geral sobre as Técnicas de Criatividade e suas classificações –, e o conceito de SOA e REST.

3.1 Técnicas de Criatividade

Existem diversas Técnicas de Criatividade, e para cada uma podem existir diversas variações dependendo da necessidade e do ambiente de aplicação. Alguns trabalhos tentam classificar as técnicas atualmente existentes separando-as por categorias de acordo com o propósito de cada. Segue uma classificação para as técnicas de acordo com a apresentada por Mycoted (2016b): definição de problemas, geração de ideias, seleção de ideias e processo criativo.

3.1.1 Definição de problemas

Essas Técnicas de Criatividade são voltadas para a análise de problemas, elas visam obter o máximo de entendimento possível sobre uma problemática em questão. A técnica da Redefinição Múltipla é uma delas, esta tem como objetivo chegar a uma definição concreta do problema com o qual se está lidando, isso porque as pessoas podem ainda não ter uma noção clara do que se está tentando resolver. Dessa forma algumas perguntas instigantes podem despertar novas perspectivas nos participantes, e levá-los a descobrir algo novo, ou algo que está implícito, mas que ainda não tinha sido detectado em relação ao problema original (MYCOTED, 2016b).

Uma forma de aplicação da técnica da Redefinição Múltipla é apresentada por Mycoted (2016b). Primeiramente escolhe-se um problema em aberto para o qual ainda não se tem respostas concretas, depois responde-se a perguntas provocativas sobre o problema em questão e, por último, retorna-se à definição original para verificar se algo mudou em relação ao entendimento geral do problema. As técnicas de definição de problemas se limitam exatamente a traçar as fronteiras para um problema em específico (MYCOTED, 2016b).

3.1.2 Geração de ideias

Uma vez definido o problema que se quer tratar, as técnicas do tipo geração de ideias ajudam a levantar possíveis soluções para a resolução de problemas. Elas objetivam capturar o máximo de soluções criativas para um determinado problema, não se preocupando com a qualidade dessas ideias. Dentro dessa categoria temos uma das técnicas de geração de ideias mais conhecida, o Brainstorming, que possui diversas ramificações como o Brainstorming Imaginário e o Brainwriting. A essência é sempre a mesma: a geração de ideias por um grupo de pessoas com o princípio de suspender o julgamento em relação as ideias geradas (MYCOTED, 2016b).

O passo a passo da execução do Brainstorming pode ser descrito da seguinte forma: primeiro há uma fase de convergência em que os participantes serão convocados; depois o tema é apresentado e se assegura que todos os participantes entenderam a problemática em questão; em seguida existe uma fase de geração de ideias quando cada participante emite sugestões de resolução do problema; por fim conclui-se a sessão, e as ideias duplicadas são agrupadas (MYCOTED, 2016b). Vale ressaltar que não pode existir qualquer tipo de discussão e censura em relação as ideias geradas, para isso escolhe-se um moderador que será responsável por fazer cumprir as regras e anotar as ideias. Uma desvantagem do Brainstorming Clássico é a questão do ruído, pois as pessoas podem se sentir constrangidas em se expressar oralmente, visando minimizar este problema, foi criada uma variação da técnica original, o Brainwriting, onde os participantes se expressam de forma escrita.

Na técnica do Brainwriting, assim como no Brainstorming existe a necessidade de um moderador que guiará a discussão. O processo também é parecido, inicialmente ocorre a etapa de apresentação do problema e logo em seguida vem a fase de geração de ideias, nesse momento os participantes expressam suas ideias de forma escrita, normalmente em pequenos cartões, que são recolhidos após determinado período de tempo, depois disto as ideias são lidas pelo moderador e novamente as ideias duplicadas são agrupadas. Esta técnica é muito usada em equipes com pessoas tímidas, pois executada corretamente, garante o anonimato dos participantes (MYCOTED, 2016b).

3.1.3 Seleção de ideias

As técnicas de seleção se preocupam em escolher dentre um conjunto de ideias aquelas que fazem sentido em relação ao problema e que refletem mais nitidamente os objetivos

da discussão. Uma das técnicas de seleção de ideias é a Votação Anônima. Em se tratando de Técnicas de Criatividade, o anonimato pode ser um fator determinante para que as pessoas se sintam livres para opinar sobre determinada ideia. Opinar significa dizer estar disposto a assumir riscos. O anonimato é muito usado em ambientes onde existe muita pressão como em algumas empresas (MYCOTED, 2016b).

A técnica de votação anônima usa o anonimato para dar mais liberdade aos participantes. A partir de um conjunto de ideias, o líder – moderador da dinâmica – distribui uma seleção de ideias para as quais cada participante pode opinar. Nesse primeiro momento também é definida uma convenção de avaliação – letras, faixas de números, etc. – que servirá para avaliar cada ideia selecionada. Em seguida os participantes escolhem uma lista de ideias para as quais deseja opinar levando em consideração o limite de ideias para cada participante, logo após os participantes votam, atribuindo a cada ideia uma classificação de acordo com a convenção acordada no início da dinâmica. Por fim, o moderador recolhe e contabiliza os votos, mostrando em seguida o ranking das ideias avaliadas (MYCOTED, 2016b).

Outra técnica de seleção de ideias se chama Ideia advocate, esta é aplicada em situações onde um conjunto relativamente pequeno de ideias já foi pré-selecionado, porém não se consegue chegar a um consenso com relação às ideias restantes. Neste caso elege-se um "advogado"para cada uma das ideias, normalmente o autor da ideia, e esta pessoa terá que defender seu ponto de vista, se todos chegarem a um acordo a ideia vencedora é escolhida e a discussão se encerra, caso contrário realizam-se sucessivas rodadas de eliminação de ideias até que sobre uma vencedora (MYCOTED, 2016b).

3.1.4 Processo criativo

Por fim temos técnicas mais amplas que guiam todo o processo criativo desde a definição do problema até a fase de seleção e posteriormente a implementação das ideias. O CPS, do inglês Creative Problem Solving ou Resolução Criativa de Problemas, é uma técnica de criatividade que possui estruturas de definição de problemas de geração e seleção de ideias, e uma fase final para planejamento da implementação das idéias. Para cada etapa do processo de criatividade o CPS estabelece técnicas de convergência e divergência de ideias, as técnicas divergentes são de caráter exploratório, importantes para extrair informações, já as técnicas convergentes são responsáveis por consolidar as informações levantadas (MYCOTED, 2016b).

O CPS é composto por seis etapas, em que cada etapa possui a fase de divergência e

convergência, constituindo assim o modelo 6x2. As etapas do CPS estão descritas a seguir:

- *Mess Finding* – essa primeira fase é responsável por fazer uma varredura e identificar possíveis problemas.
- *Data Finding* – nesta fase o objetivo é coletar informações para definir de forma mais precisa o problema e possíveis soluções.
- *Problem Finding* – acontece o refinamento da definição do problema, agora sabe-se exatamente o que se quer resolver.
- *Idea Finding* – nesta fase o objetivo é gerar o maior número de ideias possível, aqui ainda não é feita uma avaliação das ideias geradas, no máximo são eliminadas as duplicações.
- *Solution Finding* – essa fase consiste em avaliar a qualidade das ideias geradas, e selecionar aquelas que realmente fazem sentido para resolver o problema em questão.
- *Acceptance Finding* – a partir das ideias selecionadas é feito um plano de ação para ser possível executar as ideias e rastrear os impactos da implementação das mesmas.

As técnicas mostradas influenciam no design do serviço no sentido da modelagem de como as informações serão armazenadas. Se uma aplicação resolver usar o serviço e aplicar diversas técnicas, o serviço tem que estar preparado para tratar de forma consistente as informações geradas.

Em seguida vamos falar em termos gerais da arquitetura orientada a serviços, este modelo arquitetural foi usado neste trabalho pois favorece a interoperabilidade do software, um ponto importante pois o nosso objetivo aqui é se comunicar com diversas ferramentas independentemente do ambiente no qual ela se encontra e da tecnologia usada em sua implementação, em segundo lugar, a arquitetura orientada a serviços permite que o sistema seja dividido em módulos independentes, dessa forma podemos ter um módulo para cada técnica que desejamos implementar, e isso sem afetar a manutenibilidade do serviço, pois a modificação em módulos existentes ou a adição de novas técnicas não impactará em outros módulos.

3.2 Serviços, SOA e REST

Um serviço pode ser definido como um sistema de software que traz consigo uma interface de comunicação para poder ser acessado, o chamado contrato de serviço, em sistemas

web essa interface vem na forma de uma API do inglês *Application Programming Interface* que é a porta de acesso para que aplicações externas possam enviar e receber mensagens do serviço (ERL et al., 2012). A computação orientada a serviços traz consigo diversos benefícios tais como: aumento da interoperabilidade, capacidade de usar mais de um provedor de serviços, alinhamento dos objetivos de negócio com objetivos tecnológicos, aumento do retorno sobre investimento (ROI), agilidade nas mudanças, entre outros (ERL et al., 2012). Aqui cada sistema de software é visto como uma unidade lógica independente (Serviço), que pode ser usado por diversas aplicações (ERL et al., 2012).

Um novo paradigma que surgiu e tem ganhado força é a Arquitetura Orientada a Serviços (SOA), esse modelo arquitetural tem mostrado grande valor de negócio sobretudo em se tratando de computação distribuída, onde normalmente existe uma grande necessidade de performance. SOA trata-se de um modelo composto por regras e restrições que dão suporte aos princípios da orientação a serviços, dessa forma SOA constitui-se em uma base sólida para a construção de diversas aplicações orientadas a serviços.

Parte essencial do projeto de aplicações orientadas a serviços é pensar no design da interface de comunicação dos serviços, o modelo REST é uma alternativa para isso. Baseado na World Wide Web, REST, do inglês *Representational State Transfer*, é um modelo de troca de mensagens na web, também considerado uma arquitetura. O modelo REST fornece uma vasta capacidade de tomada de decisão tecnológica de design, porém não garante trazer valor de negócio para a organização, ou seja, apenas o uso do REST pode não ser suficiente para produzir uma boa aplicação, visto que ele é apenas uma padrão de troca de mensagens (ERL et al., 2012). Uma prática comum para resolver este problema é promover a junção de SOA com REST, que são compatíveis devido às características dos dois estilos arquitetônicos primordialmente voltados para a web (ERL, 2005).

REST é composto por algumas restrições que o definem arquiteturalmente, algumas dessas restrições são:

- **Cliente-Servidor**

  O REST é baseado no padrão cliente-servidor, em que os clientes são os consumidores dos serviços, e o lado servidor é representado pelos serviços que provém alguma funcionalidade (ERL et al., 2012).

- **Stateless**

  Em REST, a comunicação entre cliente e servidor não deve conter informações

sobre o estado da operação, em vez o contexto da requisição deve ser suficiente para a interpretação das mensagens e para a compreensão do que está sendo solicitado (ERL et al., 2012).

- **Interface uniforme**

  Todos serviços e consumidores devem compartilhar uma interface de Comunicação padrão. Geralmente são utilizados os protocolos existentes, nativos da web, isso porque esses padrões já são largamente usados e podem ser reutilizados (ERL et al., 2012).

O modelo REST é capaz de prover alguns atributos de qualidade como portabilidade, escalabilidade, modificabilidade, que são essenciais em aplicações web distribuídas. REST pode suprir esses atributos de qualidade normalmente, porém vale ressaltar que algumas decisões de design que não são necessariamente do REST podem ajudar a atingi-los (ERL et al., 2012). Neste trabalho nós decidimos adotar o modelo REST juntamente com SOA para que o Serviço de integração de ferramentas de criatividade possa ser escalável, facilmente modificável e para que possua capacidade de integração.

4 PROCEDIMENTOS METODOLÓGICOS

Este trabalho tem como objetivo definir e implementar um serviço para promover a integração entre Técnicas de Criatividade em vários estágios do processo criativo que será baseado no modelo REST.

4.1 Analisar necessidades de informação e fazer a modelagem dos dados

Antes de qualquer ação no sentido de integração, é necessário fazer um levantamento das Técnicas de Criatividade existentes e dividi-las por categorias de acordo com seus objetivos, então será escolhido um conjunto de técnicas para prosseguir com o trabalho. Depois disso será possível fazer uma análise dos cenários de uso de cada técnica para tentar estabelecer um padrão de geração e compartilhamento de dados. Será necessário modelar as entidades da forma como serão armazenadas na ferramenta de integração.

4.2 Definir a arquitetura do serviço de integração

Com o modelo de dados pronto e com um resumo dos cenários de uso para cada técnica de criatividade, será projetada a estrutura interna dos módulos e as funcionalidades que irão compor o serviço.

4.3 Implementar o serviço de integração

Depois de uma fase de planejamento e projeto, já se tem informações suficientes para dar início à implementação do sistema. Primeiramente, serão definidas as ferramentas de apoio ao desenvolvimento, e por fim, o serviço de apoio a integração de ferramentas criativas será implementado.

4.4 Realizar estudo de caso

Para avaliar o serviço desenvolvido, será feito um estudo de caso da integração do mesmo com duas ferramentas de criatividade onde uma delas é baseada na técnica de criatividade do Brainwriting e apoia a geração de ideias, e outra ferramenta pode fazer uso de duas técnicas criativas de forma combinada, Brainwriting e Ideia Advocate. A integração das duas ferramentas com o serviço desenvolvido nesse trabalho está no contexto de dois outros trabalhos de conclusão

de curso. No contexto deste trabalho, o estudo de caso será baseado na coleta de opiniões através de entrevistas, e análise de código das ferramentas.

5 RESULTADOS

A seguir serão apresentados os resultados do desenvolvimento do Serviço web para integrar ferramentas de criatividade. Serão descritas as atividades realizadas em cada fase definida nos procedimentos metodológicos.

5.1 Modelagem dos dados

Duas técnicas foram selecionadas para implementação na versão inicial do serviço: *Brainwriting* e *Ideia advocate*. A técnica de *Brainwriting* é classificada como de geração de ideias, já *IdeiaAdvocate* é classificada como de avaliação de ideias. Essas técnicas foram escolhidas para serem implementadas em duas aplicações de usuário final que vão utilizar o serviço como retaguarda.

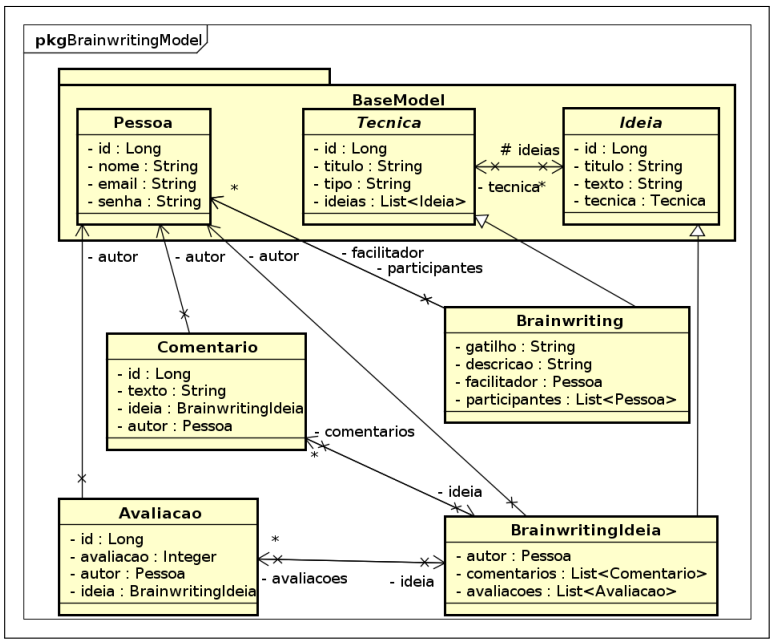
Com foco na geração de ideias, a técnica de *Brainwriting* possui um tema central, uma pergunta instigadora para fomentar a discussão, pode também ser necessária uma breve descrição apenas para situar os participantes, e um moderador que será responsável por guiar a discussão. Em um *Brainwriting* será gerado um conjunto de ideias que consistem apenas em um breve texto, os participantes da discussão podem fazer comentários para as ideias geradas e podem atribuir sua avaliação a uma ideia (MYCOTED, 2016a).

Já a técnica de *IdeiaAdvocate* tem como objetivo selecionar ideias de um conjunto inicial, assim como no *Brainwriting* existem um tema central, um gatilho, uma descrição da discussão, e a figura de um moderador. O *IdeiaAdvocate* possui um conjunto pré-definido de ideias, que podem ser avaliadas como forma de seleciona-las das demais e podem ser atribuídos comentários as ideias (MYCOTED, 2016c).

Para cada técnica foram identificadas entidades, atributos e relacionamentos. Com essas informações foi possível elaborar o modelo de dados para que o serviço pudesse prover as funcionalidades para cada técnica. O modelo de entidades gerado pode ser visto nas figuras 1 e 2, uma figura para cada pacote de entidades representado uma técnica distinta.

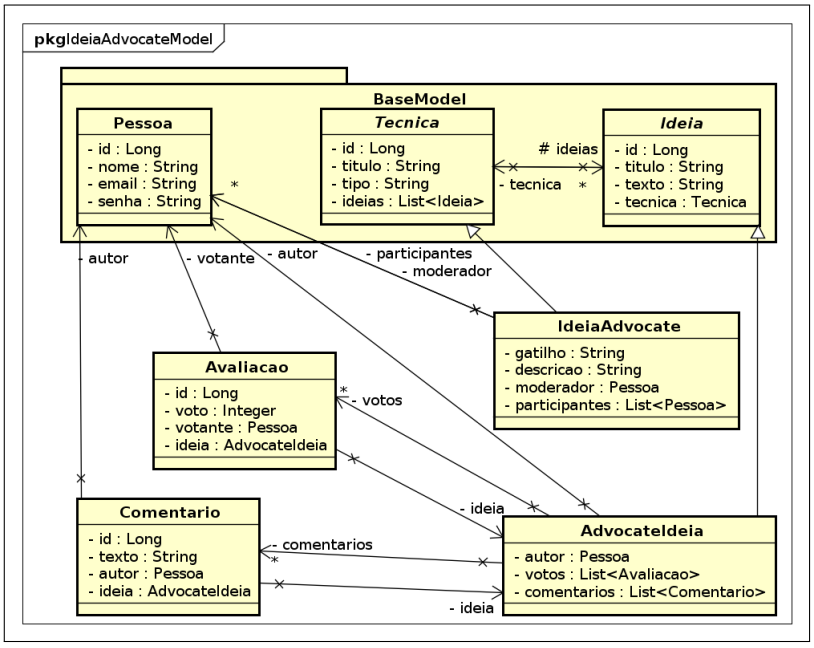
Como pôde ser visto as técnicas demandaram modelagens de entidades distintas. Entretanto para que as mesmas pudessem ser associadas a um mesmo tipo, criamos um tipo abstrato chamado 'Tecnica' contendo um identificador, o título e um atributo chamado 'tipo' para diferenciar a técnica que está em uso no momento (ver Figuras 1 e 2). Processo semelhante foi realizado para modelar o tipo Ideia que está associado a uma técnica de criatividade.

Figura 1 – Entidades da técnica *Brainwriting*



Fonte: Elaborado pelo autor

Figura 2 – Entidades da técnica *Ideia advocate*



Fonte: Elaborado pelo autor

5.2 Modelagem do serviço

Internamente os módulos das técnicas e o módulo de pessoas possuem submódulos de quatro tipos básicos, sendo eles: *model, repository, service e controller*. Esses submódulos refletem a arquitetura em camadas do serviço, na parte inferior fica a camada *repository*, logo

acima vem a camada *service* e na parte superior fica a camada *controller*, a camada *model* é transversal a todas as anteriores, pois as mesmas utilizam o modelo para executar seus métodos. A seguir uma descrição mais detalhada de cada camada.

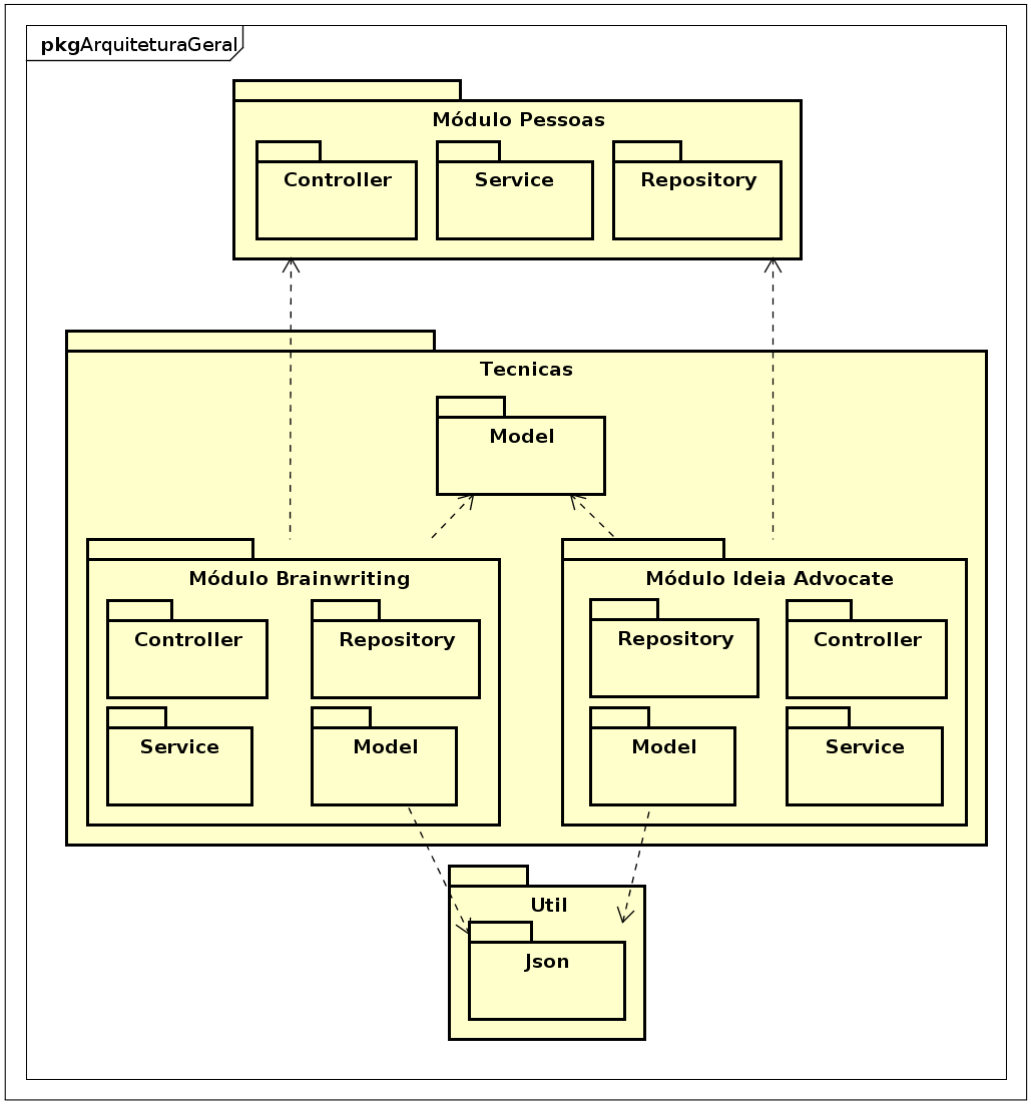
- **Model:** Contém as classes POJO e o mapeamento com JPA para o banco de dados.
- **Repository:** Contém os repositórios de dados para cada classe do modelo, esses são responsáveis por fazer as buscas no banco de dados.
- **Service:** Aqui ficam as classes de serviço, são elas que implementam as regras de negócio associadas a cada funcionalidade, fazendo o processamento dos dados, encaminhando as consultas a camada de *repository* e devolvendo as devidas respostas a camada de *controller*.
- **Controller:** Contém os controladores web responsáveis por receber as requisições e encaminha-las a camada de serviço, além de processar e enviar as respostas aos remetentes.

Cada técnica de criatividade foi organizada em um módulo. Para controle de acesso e identificação de participantes que forem utilizar as técnicas, foi organizado o módulo de Pessoas (ver Figura 3). Esse módulo possui a responsabilidade de adicionar novos usuários e editar as informações dos usuários do sistema, não possuindo dependências com os demais módulos.

Cada técnica possui seu próprio modelo e compartilham apenas o módulo de pessoas que, dessa forma é possível uma mesma pessoa participar tanto de uma sessão de brainwriting como de ideia advocate.

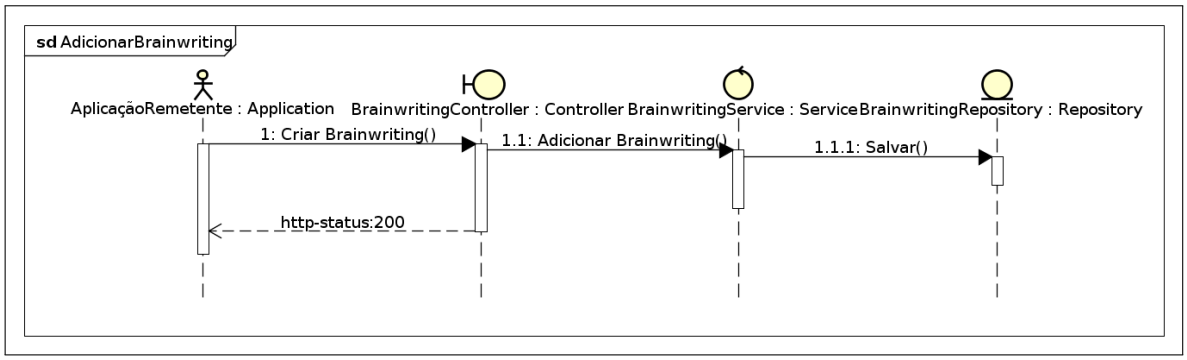
A figura 4 descreve o processo executado pelo serviço para tratar uma requisição que deseja criar um novo *brainwriting*. Ao receber a requisição o controlador envia as informações recebidas para a camada de serviço que faz a validação dos dados e depois manda o repositório fazer a persistência no banco de dados. Depois disto se não ocorrerem erros o controlador envia uma resposta para o remetente com o *status http 200* contendo um resumo do novo *brainwriting* adicionado.

Figura 3 – Arquitetura geral do serviço



Fonte: Elaborado pelo autor

Figura 4 – Fluxo de operação para adicionar um *brainwriting*



Fonte: Elaborado pelo autor

5.3 Desenvolvimento do serviço

Nesta seção será apresentado o desenvolvimento do Serviço para integração de ferramentas e criatividade de acordo com o modelo de dados e arquitetura base definidos. A seguir descreveremos todas as fases do processo de desenvolvimento desde a configuração do ambiente de desenvolvimento, até configuração da documentação automática com Swagger[1].

5.3.1 Ambiente de desenvolvimento

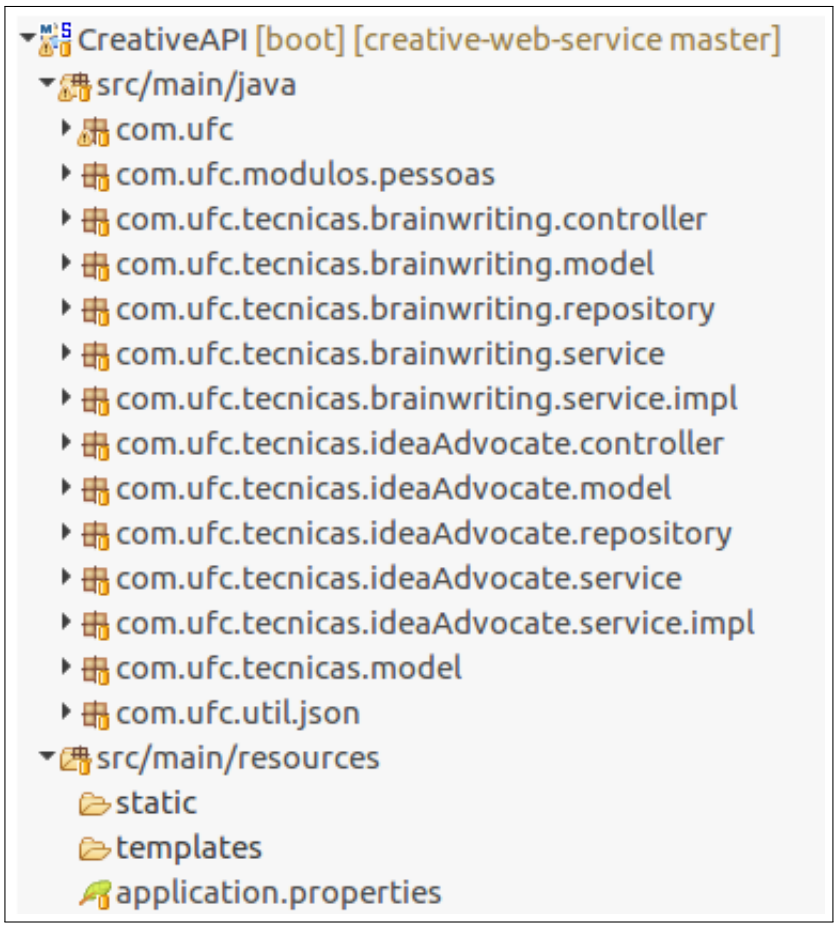
Primeiramente criamos um projeto Spring Boot[2]utilizando a IDE Eclipse[3]e o STS(Spring Tool Suite)[4], que esta disponível como um plugin para a mesma, e que auxilia o desenvolvimento de projetos em Java utilizando o framework Spring. Dessa forma foi criado um projeto em branco com as principais configurações criadas. A estrutura de diretórios do projeto pode ser vista na figura 5.

Criamos ainda um repositório remoto no GitHub[5]para guardar os artefatos do projeto e garantir a disponibilidade do mesmo, dessa forma em caso de problemas com a máquina de desenvolvimento a aplicação estaria a salvo no repositório remoto. Criamos também um projeto no GitHub para acompanhar as atividades do serviço. As figuras 6 e 7 mostram a página inicial do projeto no GitHub e parte do acompanhamento de atividades em determinado momento do mesmo, respectivamente. O código-fonte do projeto encontra-se atualmente disponível no GitHub[6]sobre a licença GPL[7](*General Public License*).

---

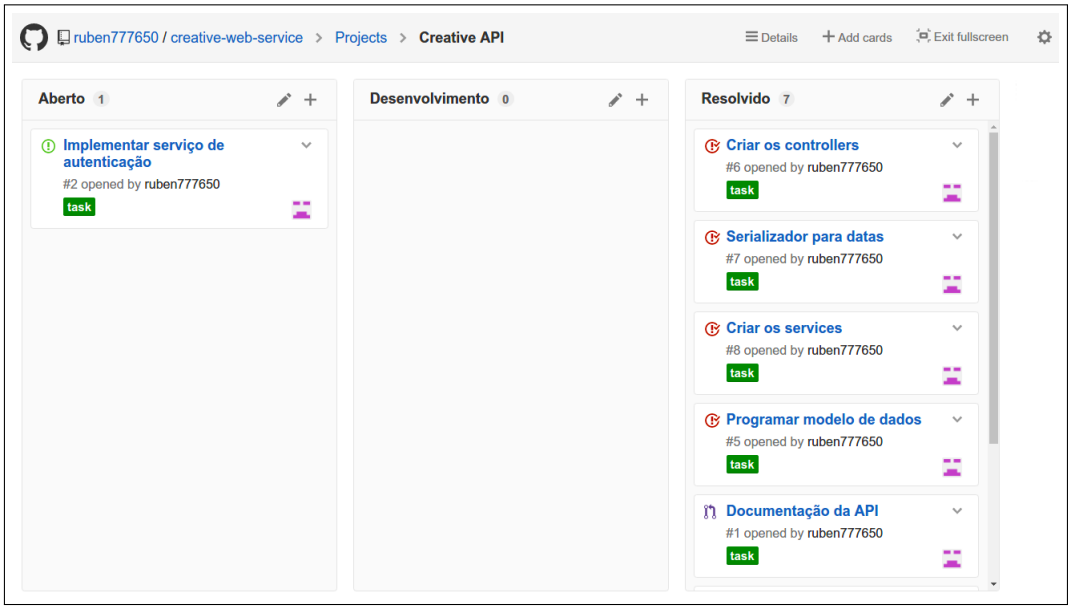
1 <http://swagger.io/>
2 <https://projects.spring.io/spring-boot/>
3 <https://eclipse.org/>
4 <https://spring.io/tools>
5 <https://github.com/>
6 <https://github.com/ruben777650/creative-web-service>
7 <https://www.gnu.org/licenses/gpl-3.0.en.html>

Figura 5 – Estrutura de diretórios do projeto



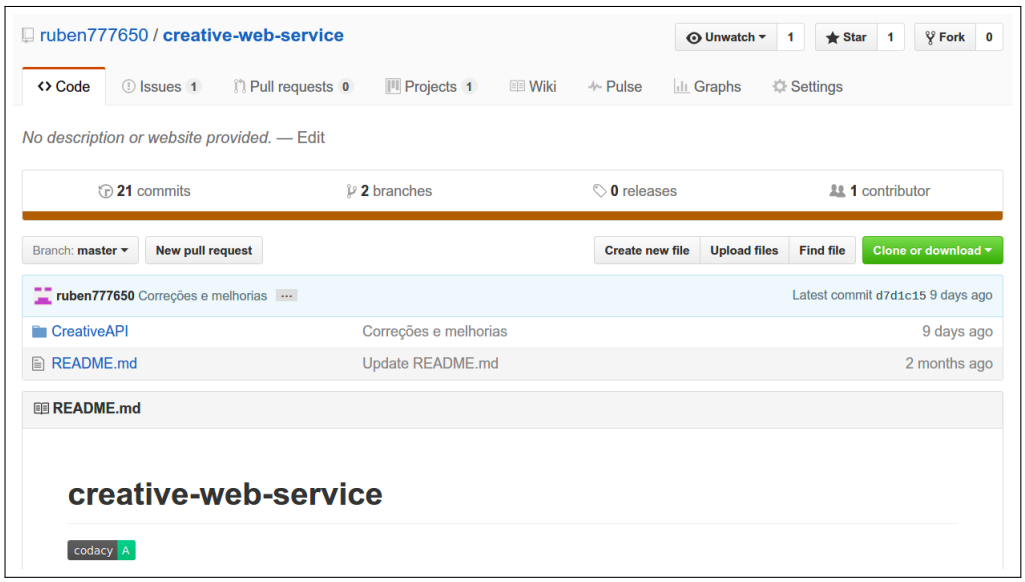
Fonte: Elaborado pelo autor

Figura 7 – Atividades no sistema de controle de mudanças do GitHub



Fonte: Elaborado pelo autor - Link: <https://github.com/ruben777650/creative-web-service/projects/1>

Figura 6 – Página inicial do repositório remoto no GitHub



Fonte: Elaborado pelo autor - Link: <https://github.com/ruben777650/creative-web-service>

5.3.2 Implementação

Por se tratar de um projeto com Spring Boot não houve necessidade de empregar um esforço inicial de configuração do projeto para que o mesmo pudesse vir a funcionar de maneira adequada. Só para gerar a documentação da API será necessário algum esforço de configuração.

5.3.2.1 Mapeamento Objeto-Relacional

Primeiramente foi necessário fazer o mapeamento objeto-relacional com JPA. Devido à experiência adquirida em outros projetos fazer o mapeamento foi uma tarefa que não demandou muita dificuldade, entretanto por se tratar de um serviço de integração de Ferramentas de criatividade foi necessário empregar algum tempo para modelar uma solução que viesse a mapear a herança das técnicas e ideias para o banco de dados, este foi o principal problema quanto ao mapeamento. Depois de alguma pesquisa descobrimos que o JPA possui suporte a herança e então pudemos começar o trabalho. A seguir podemos ver o trecho de código onde é modelada a herança para o Tipo Tecnica.

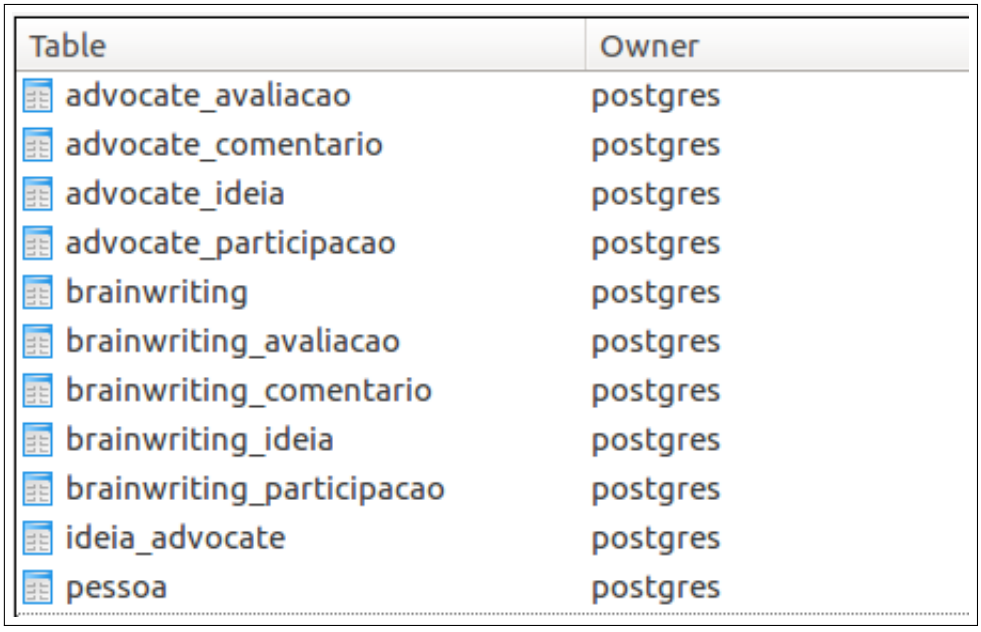
Código-fonte 1 – Mapeamento com JPA da classe Tecnica

```
1 @Entity
2 @Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
3 public abstract class Tecnica {
4
```

```
5   @Id
6   @GeneratedValue
7   private Long id;
8
9   private String titulo;
10
11  private String tipo;
12
13  @OneToMany
14  @JoinColumn(name = "tecnica_id")
15  protected List<Ideia> ideias;
16 }
```

Através de anotações podemos dizer para o JPA que esta é uma superclasse e que queremos que cada subclasse tenha uma tabela própria no banco de dados. O resultado das tabelas geradas no banco pode ser visto na figura 8, nesse caso as técnicas de Brainwriting e Ideia Advocate possuem cada uma a própria tabela, e o mapeamento entre as diferentes técnicas é feito automaticamente pelo JPA.

Figura 8 – Esquema final no banco de dados PostgreSQL



Fonte: Elaborado pelo autor

5.3.2.2 Implementação das funcionalidades

Neste trabalho escolhemos as técnicas de Brainwriting e Ideia advocate para compor o conjunto inicial de técnicas abrangido pelo serviço. O nosso primeiro esforço foi em, identificar

para cada técnica as funcionalidades que o serviço teria de ofertar, esta tarefa já tinha sido pre-realizada na etapa de geração do modelo, pois tivemos que identificar possíveis funcionalidades como base para o modelo de dados. Dessa forma fizemos um refinamento do trabalho anterior e chegamos a um conjunto de funcionalidades que estariam no serviço para cada técnica. As funcionalidades estão descritas a seguir.

1. **Brainwriting**
    - Adicionar um novo brainwriting
    - Atualizar os dados de um brainwriting
    - Buscar um brainwriting em específico
    - Buscar todos os brainwritings de uma pessoa
    - Adicionar uma ideia em um brainwriting
    - Buscar uma ideia em específico
    - Buscar todas as ideias de um brainwriting
    - Adicionar um avaliação em uma ideia
    - Adicionar um comentário em uma ideia
    - Adicionar um participante em um brainwriting
    - Buscar todos os participantes de um brainwriting
2. **Ideia advocate**
    - Buscar um ideia advocate em específico
    - Buscar todos os ideia advocate de uma pessoa
    - Criar um novo ideia advocate
    - Adicionar uma uma nova ideia em um ideia advocate
    - Buscar uma ideia em específico
    - Adicionar uma avaliação em uma ideia
    - Adicionar um comentário em uma ideia
    - Adicionar um participante em um ideia advocate
3. **Módulo de pessoas**
    - Buscar todas as pessoas no serviço
    - Buscar uma pessoa em espesífico
    - Adicionar uma nova pessoa
    - Atualizar as informações de uma pessoa

Foram criados três módulos com as funcionalidades especificadas, os módulos de

brainwriting, ideia advocate e o módulo de pessoas para dar suporte aos outros dois, este ultimo é basicamente um CRUD de pessoas. Cada módulo é formado por um controlador, uma classe de serviço e por vários repositórios dependendo da quantidade de entidades presente no modelo.

Quanto à camada de repositório o Spring Boot foi de grande ajuda pois facilitou a comunicação com o banco de dados, permitindo fazer buscas relativamente complexas com grande facilidade através de uma sintaxe própria, além de prover alguns tipos de operações de busca e persistência por padrão. A seguir temos um trecho de código onde é feita uma busca pela entidade Brainwriting com base em dois dos seus atributos.

Código-fonte 2 – Repositório do módulo de Brainwriting

```
1  public interface BrainwritingRepository extends JpaRepository<
       Brainwriting, Long> {
2
3    List<Brainwriting> findDistinctByFacilitadorOrParticipantes(Pessoa
       facilitador, Pessoa participante);
4  }
```

Para cada técnica nós criamos uma camada de serviço que é responsável por prover as funcionalidades da técnica. A classe que implementa as regras de negócio obedece a uma interface que dita os métodos necessários, sendo que a camada superior, dos controladores, somente conhece esta interface. Dessa forma podem haver várias implementações para a mesma interface, e o Spring, que suporta a injeção de dependências, instanciará a classe necessária. A seguir temos um trecho de código que mostra a interface da camada de serviço do módulo de brainwriting.

Código-fonte 3 – Interface de serviço do módulo de Brainwriting

```
1  public interface IBrainwritingService {
2
3    void adicionar(Brainwriting brainwriting);
4
5    void alterar(Long idBrainwriting, Brainwriting brainwriting);
6
7    void vincularIdeia(Brainwriting brainwriting, BrainwritingIdeia ideia
       );
8
9    List<BrainwritingIdeia> buscarIdeias(Brainwriting brainwriting);
10
11   void vincularParticipante(Pessoa pessoa, Brainwriting brainwriting);
12
13   public void adicionarAvaliacao(BrainwritingIdeia ideia, Avaliacao
       avaliacao);
14
```

```
15    public void adicionarComentario(BrainwritingIdeia ideia, Comentario
          comentario);
16
17    public List<Brainwriting> buscarBrainwritingPorPessoa(Pessoa pessoa);
18  }
```

Criamos também, para cada técnica a camada de controle, onde são recebidas as requisições e enviadas as respostas, nessa fase o Spring Boot foi de grande ajuda pois facilitou o mapeamento dos caminhos das URLs e dos métodos HTTP, através de anotações. Temos a seguir um trecho de código onde a funcionalidade de adicionar um novo brainwriting é mapeada para a url: *"/brainwriting"* no método POST.

Código-fonte 4 – Classe controller do módulo de Brainwriting

```
1  @RestController
2  @RequestMapping(value = "/brainwriting")
3  public class BrainwritingController {
4
5      @PostMapping(consumes = MediaType.APPLICATION_JSON_UTF8_VALUE)
6    public ResponseEntity<Brainwriting> adicionarBrainwriting(
         @RequestBody Brainwriting brainwriting) {
7
8      brainwritingService.adicionar(brainwriting);
9      return new ResponseEntity<>(brainwriting, HttpStatus.OK);
10   }
11
12 }
```

5.3.2.3 Documentação da API

Para facilitar o uso do Serviço para integrar ferramentas de criatividade, e para possibilitar maior segurança quanto às informações a serem enviadas e o tipo de retorno, foi criada uma documentação para a API do serviço. Por ter integração com o Spring Boot na forma de uma biblioteca, e também por ser amplamente utilizada no mercado, nós utilizamos a ferramenta Swagger para gerar a documentação do serviço.

Inicialmente tivemos alguma dificuldade para configurar o Swagger no serviço pois o mesmo precisa saber o caminho para os controladores no projeto, além disso havia total desconhecimento de nossa parte com relação ao funcionamento da biblioteca, entretanto, depois de algum tempo obtivemos êxito na configuração da biblioteca no projeto. A configuração do Swagger pode ser vista no trocho de código que segue.

Código-fonte 5 – Configuração do Swagger no serviço

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

  @Bean
  public Docket api() {
    return new Docket(DocumentationType.SWAGGER_2).select()
        .apis(RequestHandlerSelectors.basePackage("com.ufc")).paths(
           PathSelectors.any()).build()
        .apiInfo(apiInfo());
  }

  private ApiInfo apiInfo() {
    ApiInfoBuilder builder = new ApiInfoBuilder();

    builder.title("Creative API").description("Servico Web para
       integrar Ferramentas de criatividade").version("0.1");

    return builder.build();

  }

}
```

O Swagger funciona através de anotações nos controladores, ao iniciar o projeto essas anotações são lidas e então o Swagger gera uma documentação para a API do serviço, que pode ser acessada pelo navegador, como um componente do serviço.

Nós usamos basicamente dois tipos de anotações, a anotação *@Api* que é colocada acima do cabeçalho da classe para informar que aquele arquivo é um controlador e deve ser lido, a segunda anotação é *@ApiOperation*, usada acima do cabeçalho dos métodos para informar que aquele método deve ser incluído na documentação. A seguir temos um trecho de código onde são usadas essas duas anotações. A documentação gerada pelo Swagger pode ser vista nas figuras 9 e 10.

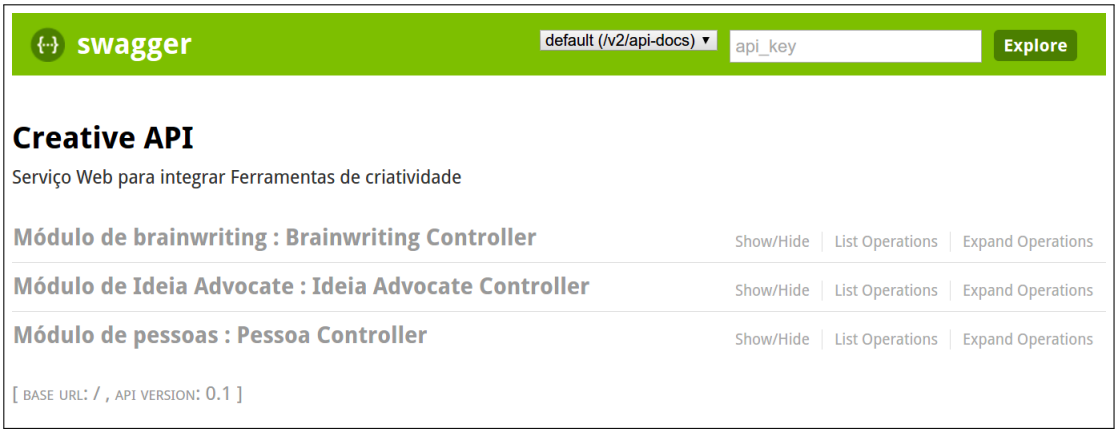
Código-fonte 6 – Mapeamento do BrainwritingController com Swagger

```
@Api(tags = "Modulo de brainwriting")
public class BrainwritingController {

  @ApiOperation(value = "Retorna todos os brainwriting de uma pessoa",
     notes = "O metodo retorna tanto os brainwriting que uma pessoa
     modera quanto aqueles que ela participa.")
  public List<Brainwriting> getBrainwriting(@RequestParam("idPessoa")
     Pessoa pessoa) {
    return brainwritingService.buscarBrainwritingPorPessoa(pessoa);
```

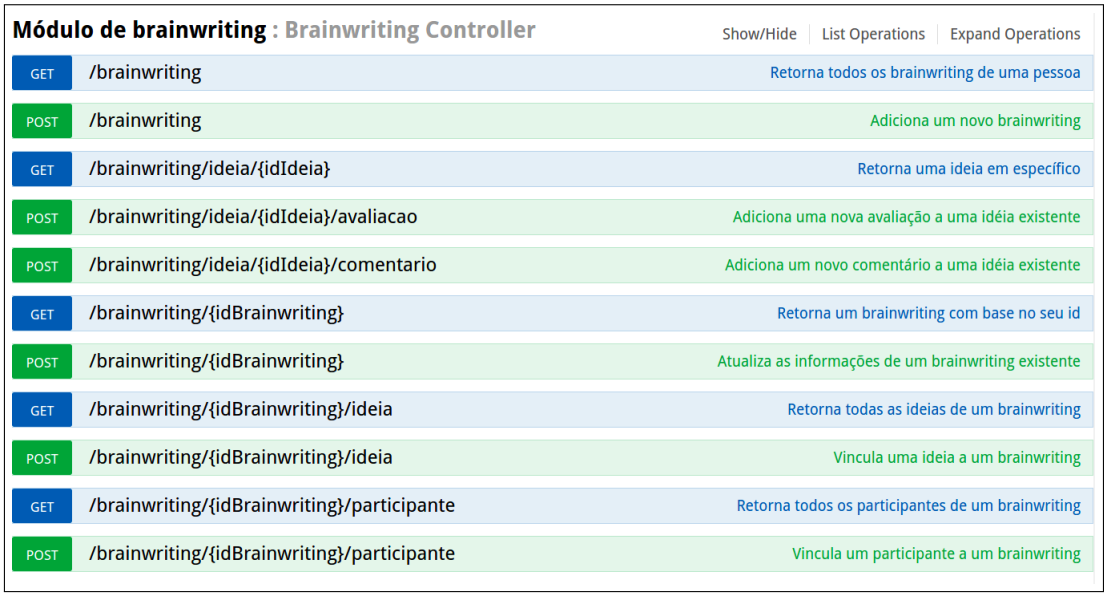
```
7     }
8 }
```

Figura 9 – Visão geral da documentação do Swagger



Fonte: Elaborado pelo autor

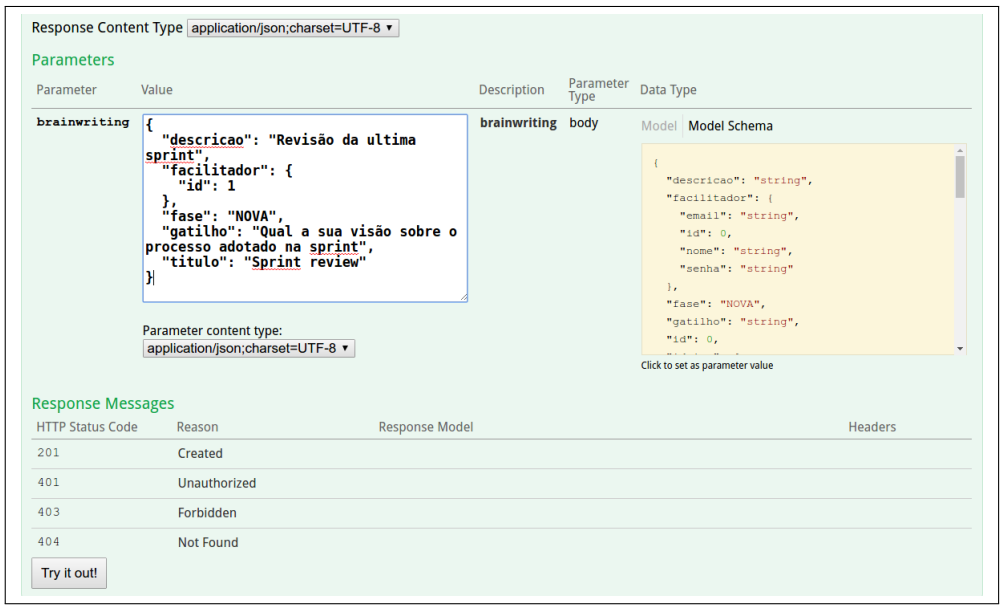
Figura 10 – Visão geral do módulo de brainwriting



Fonte: Elaborado pelo autor

Além da documentação, o Swagger permite enviar requisições ao serviço, simulando uma aplicação externa, com isso pudemos testar de maneira mais fácil a API do serviço, fazendo as requisições e verificando as respostas. As figuras 11 e 12 mostram a realização de uma requisição com Swagger e a resposta obtida, respectivamente.

Figura 11 – Envio de uma requisição com Swagger



Fonte: Elaborado pelo autor

Figura 12 – Resposta de uma requisição com Swagger

Fonte: Elaborado pelo autor

5.4 Cenários de integração

O propósito do serviço desenvolvido neste trabalho é possibilitar a integração de ferramentas de criatividade, visto isto, destacamos dois cenários simples de integração de ferramentas através do serviço: integração de ferramentas em um mesmo módulo, ou integração de ferramentas em módulos diferentes, estes cenários estão descritos a seguir.

5.4.1 Cenário 1 - Ferramentas usando o mesmo módulo

Duas ou mais ferramentas de criatividade independentes resolvem utilizar o serviço no módulo de *Brainwriting*, por exemplo. A ferramenta A se conecta ao serviço, é criada uma sessão de *Brainwriting* e são adicionadas algumas ideias em um primeiro momento, posteriormente a aplicação B acessa a discussão criada e adiciona mais ideias, mais tarde, quando a ferramenta A voltar a se conectar com o serviço, as alterações geradas por B poderão ser visualizadas.

Este cenário pode ser claramente visualizado em um contexto onde duas ferramentas de criatividade, sendo elas, *mobile* e *web* respectivamente, implementam a técnica do *Brainwriting* e se conectam ao serviço para guardar as discussões criadas, desta forma as alterações realizadas em qualquer das duas seria imediatamente refletida na outra ferramenta.

5.4.2 Cenário 2 - Ferramentas usando módulos diferentes

Várias ferramentas de criatividade estão utilizando o serviço para guardar as suas discussões de *Brainwriting*, uma ferramenta A por sua vez, decide fazer a aplicação do *IdeiaAdvocate* em cima de uma sessão de *Brainwriting* existente, a ferramenta importa uma sessão de *Brainwriting* e instancia uma nova sessão de *IdeiaAdvocate* no serviço contendo todas as ideias geradas anteriormente no *Brainwriting*.

Este cenário de uso pode ser ilustrado no contexto de duas ferramentas de criatividade que utilizam o *Brainwriting* e o *IdeiaAdvocate*, respectivamente, neste caso a ferramenta de *Brainwriting* compartilha indiretamente as ideias geradas com a ferramenta de *IdeiaAdvocate*.

Estes dois cenários de uso são passiveis de implementação no mundo real e podem servir para alinhar as funcionalidades de cada técnica presentes no serviço com as necessidades das ferramentas de criatividade que venham a consumi-lo.

5.5 Avaliação

Nesta seção será apresentada a avaliação do serviço desenvolvido, nesta etapa foi realizado um estudo de caso para avaliar o uso do serviço por duas ferramentas que dão suporte a técnicas de criatividade, essas ferramentas foram desenvolvidos em dois trabalhos de conclusão de curso da Universidade Federal do Ceará no campus de Quixadá, uma delas é um aplicativo *mobile* que implementa a técnica do *Brainwriting*, esta primeira fez uso do módulo de *Brainwriting* do serviço, a segunda ferramenta pertence a plataforma *web* e implementa duas técnicas de criatividade, o *Brainwriting* e *Ideia advocate*, esta ultima fez uso do serviço apenas no módulo de *Ideia advocate*.

Ao longo do trabalho foram feitas reuniões de alinhamento com os desenvolvedores na tentativa de aproximar o serviço das ferramentas *web* e *mobile*, estas reuniões fizeram com que o serviço se integrasse melhor com as ferramentas, entretanto com o decorrer do tempo pode ter ocorrido certo desalinhamento entre ambos, por isso a validação do serviço se torna necessária.

Para realizar a avaliação do serviço foram usadas as técnicas de entrevista e análise de código. A entrevista foi usada para coletar a visão dos desenvolvedores de ambas as ferramentas sobre a experiencia de uso do serviço, e a análise de código para verificar quais recursos do serviço foram utilizados por cada ferramenta e tentar identificar inconsistências de uso.

5.5.1 Método de avaliação

O método de avaliação utilizado neste trabalho é composto por entrevistas com os desenvolvedores e em seguida uma etapa de análise de código dos sistemas que fizeram uso do serviço.

5.5.1.1 Entrevista

Para realizar as entrevistas com os desenvolvedores foi elaborado um questionário com três perguntas básicas, essas perguntas objetivam saber como o serviço foi capaz de atender as funcionalidades de cada sistema. As perguntas elaboradas são apresentadas em seguida.

1. Quais os casos de uso/estorias de usuário da aplicação foram suportados pelo serviço?
2. Quais os casos de uso/estorias de usuário da aplicação que deveriam ter sido suportados pelo serviço?

3. O que você recomendaria para a evolução do serviço?

As duas primeiras perguntas visam investigar quanto do escopo atual de cada ferramenta de criatividade foi alcançado pelo serviço, já a ultima pergunta quer saber quais melhorias seriam necessárias, e que funcionalidades poderiam ser inseridas ou modificadas para aprimorar o uso do serviço pelas ferramentas de criatividade. Este modelo de entrevista foi aplicado para cada ferramenta e para cada módulo do serviço que foi utilizado pela mesma.

5.5.1.2 Análise de código

A análise de código consiste em explorar o código-fonte das ferramentas de criatividade tomadas como exemplo neste trabalho, esta tarefa pretende averiguar a maneira como o serviço foi utilizado. A análise do código se deteve em três pontos:

- Identificar os métodos chamados no serviço e quantidade de pontos distintos no código em que o mesmo é utilizado
- Descobrir quais métodos do serviço não foram usados pela ferramenta
- Verificar se os métodos foram usados de alguma forma errada

A análise não visa fiscalizar a qualidade do código-fonte de cada ferramenta, mas verificar a qualidade de uso do serviço por parte de cada ferramenta. Esta análise foi aplicada no aplicativo que implementa a técnica do *Brainwriting* e no sistema web apenas no módulo de *Ideia advocate*.

5.5.2 Avaliação do módulo Brainwriting

5.5.2.1 Entrevista

Foi realizada uma entrevista por videoconferência com o desenvolvedor do aplicativo *mobile* da técnica do *Brainwriting*. Primeiramente foi apresentado o objetivo da atividade e seguiu-se a aplicação das perguntas.

O desenvolvedor do aplicativo *mobile* modelou os requisitos na forma de estórias de usuário, ao ser questionado sobre quais estórias o serviço foi capaz de atender o mesmo declarou que todas elas foram atendidas pelo serviço. As estórias criadas para o aplicativo podem ser vistas no Quadro 1.

Segundo o entrevistado todas estas estórias foram suportadas pelo serviço, então a resposta da segunda pergunta ficou implícita, pois segundo o mesmo nenhuma estória foi deixada

de lado pelo serviço. Partimos então para a terceira pergunta, foi levantado o questionamento sobre possíveis evoluções do serviço, o entrevistado levantou algumas sugestões e melhorias, são elas:

- Implementar a busca por e-mail no módulo de pessoas
- Simplificar o envio de datas que demandavam muitos campos desnecessários
- Melhorar a documentação no que se refere ao exemplo de JSON enviado na requisição, que não está apresentado de forma clara

Com o resultado da entrevista pudemos perceber que o serviço de fato pôde suportar as funcionalidades do aplicativo, entretanto faltou clareza quanto as informações que deveriam ser incluídas no JSON da requisição, isso devido a uma falha na documentação da API do serviço. Percebemos também que houve uma pequena dificuldade na busca por uma pessoa cadastrada no serviço, pois a forma ideal seria a busca por e-mail, como foi sugerido pelo desenvolvedor.

Quadro 1 – Estórias de usuário

R01	**Eu** como uma Participante **posso** realizar o cadastro no sistema **para** utilizar a ferramenta
R02	**Eu** como Participante **posso** logar no sistema com email e senha **para** obter acesso as discussões que participo ou facilito
R03	**Eu** como Participante **posso** sair do sistema **para** não obter acesso as discussões que participo
R04	**Eu** como Participante **posso** ver os grupos que participo **para** obter acesso a suas configurações como participante ou facilitador
R05	**Eu** como Participante **posso** aceitar o convite de vinculação a uma discussão **para** obter acesso às funções da discussão
R06	**Eu** como Participante **posso** ver o tema de discussão **para** saber obter orientação sobre o que deve ser discutido
R07	**Eu** como Participante **posso** sugerir ideia sobre o tema **para** contribuir na discussão.
R08	**Eu** como Participante **posso** ver as ideias escritas por outros participantes **para** poder ler as ideias descritas e poder fazer novas ideias com base nessas ideias.
R09	**Eu** como Participante **posso** comentar ideias minhas e de outros participantes **para** discutir detalhes sobre uma ideia sugerida.
R10	**Eu** como Participante **posso** avaliar as ideias de outros participantes **para** emitir valor sobre a qualidade da ideia gerada.
R11	**Eu** como Participante **posso** ver o resultado final da discussão **para** saber quais as ideias mais valorizadas por todos os participantes.
R12	**Eu** como Facilitador **posso** criar um novo grupo de discussão **para** iniciar a rodada de Brainwriting.
R13	**Eu** como Facilitador **posso** convidar participantes para uma discussão **para** que os participantes possam gerar novas ideias no grupo de discussão.
R14	**Eu** como Facilitador **posso** definir o tema de discussão do grupo **para** guiar as contribuições dos participantes durante a discussão.
R15	**Eu** como Facilitador **posso** iniciar a etapa de recebimento de ideias **para** que os participantes enviem suas ideias isoladamente.
R16	**Eu** como Facilitador **posso** iniciar a etapa de discussão das ideias **para** divulgar as ideias geradas para todos os participantes.
R17	**Eu** como Facilitador **posso** iniciar a etapa de avaliação das ideias **para** fomentar uma seleção das melhores ideias pelos participantes.
R18	**Eu** como Facilitador **posso** encerrar a discussão **para** divulgar aos participantes o resultado final da discussão.

Fonte: Elaborado pelo desenvolvedor do aplicativo *mobile*

5.5.2.2 Análise de código

Conforme planejado foi realizada a análise do código do aplicativo *mobile*, a análise do aplicativo se restringiu apenas a parte do código que continha os acessos a API do serviço,

pois esta é a parte onde são montadas as requisições e onde são tratados os resultados vindos do serviço.

Primeiramente pedimos ao desenvolvedor do aplicativo *mobile* para que o mesmo nos concedesse acesso aos trechos de código onde eram feitas as requisições ao serviço, o código-fonte do aplicativo foi disponibilizado através de um repositório remoto no GitHub[8]. De posse do código-fonte iniciamos a primeira fase da análise que consiste em verificar os métodos da API chamados e a quantidade de vezes que eles ocorriam no decorrer do código, nesta fase foram identificadas as seguintes chamadas:

- Adicionar participante em um *Brainwriting*
- Buscar todos os *Brainwriting* de uma pessoa
- Buscar um *Brainwriting* pelo identificador
- Atualizar um *Brainwriting* existente
- Criar um *Brainwriting*
- Buscar todas as ideias de um *Brainwriting*
- Buscar uma ideia pelo identificador
- Adicionar uma ideia em um *Brainwriting*
- Adicionar um comentário em uma ideia
- Adicionar uma avaliação em uma ideia

Destes métodos somente "Buscar uma ideia pelo identificador"foi chamado mais de uma vez, todas os outras foram encontrados em apenas um local no código, a seguir podemos ver um trecho de código onde é feita uma requisição ao serviço.

Código-fonte 7 – Aplicativo *mobile* - Buscar uma ideia

```
1 .factory("getIdeaService", ["$http", function($http){
2   return {
3     getIdea: function(id){
4       return $http.get("http://localhost:8080/CreativeAPI/brainwriting/
            ideia/"+ id);
5     }
6   };
7 }])
```

Na segunda parte da análise procuramos por métodos da API do serviço que não foram utilizados pelo aplicativo *mobile*, nesta etapa conseguimos detectar que apenas o método "Buscar todos os participantes de um *Brainwriting*"não foi acessado, um dos motivos pelos quais

[8] <https://github.com/AllefLobo/Elec-Brain>

isso aconteceu se deve ao fato de que os participantes podem ser recuperados através de um *Brainwriting*.

Para finalizar a análise de código nós tentamos identificar métodos da API sendo usados de forma errada no aplicativo *mobile*, nesta ultima análise identificamos uma pequena inconsistência no uso do método "Buscar uma ideia pelo identificador"que foi usado para recuperar os comentários de uma ideia já que o serviço não proveu um método para recuperar especificamente a lista de comentários.

A análise de código nos permitiu verificar que o serviço conseguiu de fato atender a demanda do aplicativo, todavia ficou patente a necessidade de um acesso mais especializado aos recursos do módulo de *Brainwriting*.

5.5.3 Avaliação do módulo IdeiaAdvocate

5.5.3.1 Entrevista

Para avaliar o módulo de *IdeiaAdvocate* foi primeiramente realizada uma entrevista com o desenvolvedor do sistema web, de acordo com o planejado. A entrevista foi realizada de forma presencial e foi feita uma breve introdução apresentando o objetivo da entrevista.

Terminada a introdução, foram feitas as perguntas. Para a primeira pergunta o entrevistado que modelou suas funcionalidades na forma de estórias de usuário respondeu que o serviço foi capaz de dar suporte a todas as funcionalidades requeridas, entretanto o entrevistado ressaltou que algumas funcionalidades foram supridas de forma pouco eficiente, um exemplo é a funcionalidade de *"Adicionar ideia"*, era esperado que o serviço pudesse receber uma lista de ideias, como só havia a opção de enviar uma única ideia foi necessário fazer várias requisições para adicionar um conjunto de ideias.

Como o serviço foi capaz de suprir todas as necessidades funcionais do sistema *web*, então a segunda pergunta não foi necessária, pulamos então para a terceira pergunta. Ao ser questionado sobre possíveis melhorias e evoluções do serviço, o entrevistado listou alguns itens que poderiam ser melhorados:

- Inserir a funcionalidade de adicionar um conjunto de ideias de uma só vez
- Inserir a busca de ideias por status

A partir do resultado da entrevista pudemos perceber que o serviço de integração, supriu todas as funcionalidades requeridas pelo sistema *web*, porém houve uma deficiência na

detecção de cenários de uso na modelagem do serviço para o módulo de *IdeiaAdvocate*, por isso foram incluídas apenas as funcionalidades básicas, impactando negativamente no sistema.

5.5.3.2 Análise de código

Como planejado, foi realizada a análise de código do sistema *web* para verificar alguns aspectos do mesmo em relação a utilização do serviço. Para viabilizar a análise foi solicitado ao desenvolvedor do sistema que fosse disponibilizado o código-fonte do projeto, o mesmo pôde ser acessado por intermédio de um repositório remoto no GitHub[9].

Após o código estar disponível foi iniciada a análise, primeiramente procuramos identificar quais métodos da API do serviço foram chamados e os pontos no código em que o mesmo se repetia, para atender ao primeiro ponto do planejamento da análise. Descobrimos que sete métodos do serviço foram utilizados no sistema, são eles:

- Buscar todos os *IdeiaAdvocate* de uma pessoa
- Adicionar um *IdeiaAdvocate*
- Buscar uma ideia
- Adicionar uma avaliação em uma ideia
- Adicionar comentário em uma ideia
- Buscar um *IdeiaAdvocate*
- Adicionar uma ideia

Cada um destes métodos foram usados uma só vez no código do sistema web. Um exemplo de uso pode ser visto no código a seguir, onde é feita uma requisição para adicionar uma ideia em um *IdeiaAdvocate*, primeiro é montado o objeto da requisição com todos os atributos necessários e em seguida a requisição é enviada.

Código-fonte 8 – Sistema *web* - adicionar uma ideia

```
1  this.addIdeiaAdvocate = function (idAdvocate, ideia_aux) {
2      var ideia = {
3          "autor": {
4              "id": ideia_aux.autor._id
5          },
6          "data": {
7              "timeInMillis": 0
8          },
9          "status": ideia_aux.status,
10         "texto": ideia_aux.texto,
```

9 <https://github.com/RobsonCN2015/API-Creative>

```
11          "titulo": ideia_aux.titulo,
12      };
13
14      var data = {
15          "ideia" : ideia,
16          "idAdvocate" : idAdvocate
17      };
18
19      var query = {
20          "method": "POST",
21          "url": API_URL_ADVOCATE+"/advocate/"+idAdvocate+"/ideia",
22          "data": data,
23          "headers": {
24              "Content-Type": "application/json;charset=UTF-8"
25          }
26      };
27
28    return $http.post(query);
29  };
```

No segundo momento da análise buscamos descobrir quais métodos da API do serviço não foram utilizados no sistema web. Descobrimos então que apenas o método de "Adicionar participante em um *IdeiaAdvocate*"não foi utilizado, após investigar mais a fundo, descobrimos que o sistema *web* adiciona todos os participantes no momento da criação de um *IdeiaAdvocate* através de um atributo, o que é permitido pelo serviço.

Por fim tentamos identificar algum método da API do serviço sendo chamado de forma errônea, a análise não revelou evidências suficientes para afirmar que os métodos estavam sendo mal utilizados, entretanto foram encontradas algumas pequenas inconsistências no nome da função que encapsulava a requisição em relação ao método correspondente na API do serviço, por exemplo, a função "buscarIdeiasDeUmaPessoa"faz uma chamada ao método da API que retorna todos os *IdeiaAdvocate* de uma pessoa, o nome da função indica o recebimento de um conjunto de ideias mas o que é trazido é na verdade um conjunto de discussões do tipo *IdeiaAdvocate*.

A partir da análise do código pudemos notar que houve um real entendimento dos métodos providos pelo serviço, e que o desenvolvedor pôde usá-los para alcançar os objetivos do sistema *web*.

5.5.4 Discussão

Com o desenvolvimento do serviço de integração de ferramentas de criatividade pudemos dar suporte a duas ferramentas de criatividade. Inicialmente foram escolhidas as

técnicas de *Brainwriting* e *IdeiaAdvocate* para compor o serviço, esta escolha teve como base as duas ferramentas, que por sua vez foram desenvolvidas em trabalhos de conclusão de curso, as técnicas escolhidas serviram de base para guiar o desenvolvimento do serviço.

A integração do serviço com as ferramentas obteve sucesso e permitiu que as mesmas pudessem implementar uma técnica de criatividade a partir da definição de métodos da API. As reuniões de alinhamento no decorrer do trabalho também contribuíram para que houvesse coerência entre o serviço desenvolvido e as ferramentas de criatividade, o que foi claramente percebido na avaliação, pois todas as funcionalidades das ferramentas foram suportadas pelo serviço.

Um ponto a ser considerado é que a integração poderia ser mais produtiva se a segunda ferramenta utilizada como para o estudo de caso (sistema *web*) tivesse utilizado o módulo de *Brainwriting* do serviço, isso nos permitiria validar o aspecto de interoperabilidade do serviço. Contudo o resultado da integração foi bastante satisfatório no que diz respeito ao que o serviço foi capaz de fazer para suprir as necessidades funcionais de cada ferramenta.

6 CONSIDERAÇÕES FINAIS

Neste trabalho nós desenvolvemos um serviço de integração de ferramentas de criatividade, para tanto foram selecionadas as técnicas para compor o serviço, sendo elas o *Brainwriting* e o *IdeiaAdvocate*, a partir destas técnicas elicitamos os cenários de uso que deram origem a modelagem de dados e arquitetura do serviço, e então seguiram-se as fases de desenvolvimento e avaliação do serviço, de acordo com o planejado.

Este trabalho fornece a base para a construção de um serviço mais robusto para integrar diversas ferramentas de criatividade e para um número maior de técnicas. O método aqui utilizado pode ser usado ou adaptado para adicionar suporte a mais ferramentas, os problemas por nós enfrentados podem evitar retrabalho por parte de quem precisar implementar outras técnicas de criatividade, e além disso, a questão da comunicação com os desenvolvedores das ferramentas, que foi um ponto crucial para o sucesso da integração, servirá como exemplo de boa prática para a evolução do serviço.

Algumas das limitações presentes se devem ao fato de que as ferramentas escolhidas para estudo de caso, não compartilharam módulos, o que enriqueceria ainda mais o trabalho pois seria possível analisar a interoperabilidade entre as ferramentas e como o serviço teria que se comportar ao ter um de seus módulos sendo usado por mais de uma ferramenta.

Para trabalhos futuros, sugerimos a especificação mais abstrata da API do serviço, uma possível implementação seria a modificação da divisão dos módulos, ao invés de divisão por técnica poderia ser adotada a divisão por categoria, cada módulo para representar uma categoria de técnicas de criatividade, por exemplo, *"Geração de ideias"* e *"Avaliação de ideias"*, deste modo a API do serviço poderia atender a um conjunto bem mais abrangente de ferramentas de criatividade de uma só vez, e a necessidade de retrabalho e modificação no serviço seria minimizada, pois vários conceitos estariam sendo reaproveitados.

Este trabalho viabilizou a integração de ferramentas de criatividade para as técnicas de *Brainwriting* e *IdeiaAdvocate*. Dessa forma tanto as aplicações usadas no estudo de caso, bem como novas aplicações podem fazer uso do serviço desenvolvido para implementar as técnicas de criatividade.

REFERÊNCIAS

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2012. ISBN 0321815734, 9780321815736.

COLLAGE. **HatParty**. 2016. Disponível em:<https://realtimeboard.com/>. Acessado em 14/05/2016.

ERL, T. **Service-oriented architecture: concepts, technology, and design**. [S.l.]: Pearson Education India, 2005.

ERL, T.; CARLYLE, B.; PAUTASSO, C.; BALASUBRAMANIAN, R. **SOA with REST: Principles, Patterns &Constraints for Building Enterprise Solutions with REST**. [S.l.]: Prentice Hall Press, 2012.

LAMPKOWSKI, M.; SILVA, J. H. Integração de sistemas de informação: A importância das integrações e como realizá-las com interoperabilidade de forma a produzirem e consumirem informações reutilizáveis. In: **III JORNACITEC**. [S.l.: s.n.], 2014.

LANGUAGES, M. **Creative Leaf: Browser-based Creative Requirements Modeling**. 2016. Disponível em:<http://modeling-languages.com/creative-leaf-requirements-modeling/>. Acessado em 20/11/2016.

MORAES, C. C. **Utilização de Técnicas de Criatividade da melhoria do processo de requisitos do Núcleo De Práticas Em Informática**. 2014. Universidade Federal do Ceará.

MYCOTED. **Brainwriting**. 2016. Disponível em:<https://www.mycoted.com/Brainwriting>. Acessado em 07/12/2016.

MYCOTED. **Creativity Techniques**. 2016. Disponível em:<https://www.mycoted.com/Category:Creativity_Techniques>. Acessado em 20/11/2016.

MYCOTED. **Idea Advocate**. 2016. Disponível em:<https://www.mycoted.com/Idea_Advocate>. Acessado em 07/12/2016.

NGUYEN, L.; SHANKS, G. A framework for understanding creativity in requirements engineering. **Information and software technology**, Elsevier, v. 51, n. 3, p. 655–662, 2009.

PLENTZ, S. S. Taxonomia para técnicas criativas aplicadas ao processo de projeto. 2011.

REALTIMEBOARD. **Online Whiteboard & Online Collaboration Tool RealtimeBoard**. 2016. Disponível em:<https://realtimeboard.com/>. Acessado em 20/11/2016.

RODRIGUES, J. F. **Influência das técnicas de criatividade nos resultados de inovação em uma empresa do ramo metalúrgico em Ponta Grossa–PR**. Tese (Doutorado) — Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná. Engenharia de Produção, 2009.

ROSA, T. P. **Um método para o desenvolvimento de software baseado em microsserviços**. 2016. Universidade Federal do Ceará.

SCHREIER, S. Modeling restful applications. In: ACM. **Proceedings of the second international workshop on restful design**. [S.l.], 2011. p. 15–21.

Sá, A. L. F. **Migração de uma arquitetura monolítica para uma arquitetura orientada a serviços**. 2016. Universidade Federal do Ceará.