



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
BACHARELADO EM ENGENHARIA DE SOFTWARE



Twydi: Uma Ferramenta de Documentação para Equipes de Desenvolvimento de Software

Igor Marques da Silva

Natal-RN
Dezembro 2015

Igor Marques da Silva

Twydi: Uma Ferramenta de Documentação para Equipes de Desenvolvimento de Software

Monografia de Graduação apresentada ao
Departamento de Informática e Matemática
Aplicada do Centro de Ciências Exatas e da
Terra da Universidade Federal do Rio Grande
do Norte como requisito parcial para a ob-
tenção do grau de bacharel em Engenharia
de Software.

Orientador

Prof. Dr. Fernando Marques Figueira Filho

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA – DIMAP

Natal-RN

Dezembro de 2015

Monografia de Graduação sob o título *Twydi: Uma Ferramenta de Documentação para Equipes de Desenvolvimento de Software* apresentada por Igor Marques da Silva e aceita pelo Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Prof. Dr. Fernando Marques Figueira Filho
Orientador
Departamento de Informática e Matemática Aplicada
UFRN

Prof. Dr. Gibeon Soares De Aquino Junior
Departamento de Informática e Matemática Aplicada
UFRN

Profa. Dra. Marcia Jacyntha Nunes Rodrigues Lucena
Departamento de Informática e Matemática Aplicada
UFRN

Natal-RN, oito de dezembro de dois mil e quize

À todos que fizeram isso possível:

Meus pais, **Roberto** e **Marcia**, por todo amor, educação e oportunidades que me deram ao longo da minha vida.

Meu irmão, **Ivan**, por alegrar os meus dias com o seu jeito de ser.

Minha querida **Mayane**, por todo amor e carinho além do que um ser humano pode receber.

Agradecimentos

Em quatro anos de graduação, muito amadureci e muito aprendi. Sem dúvida tudo isso não seria possível sem uma boa parcela de companheirismo, apoio e dedicação. Tudo o que alcancei foi graças a uma boa parcela de pessoas.

Obrigado à minha família, por tanto apoio. Não seria nada sem vocês. Literalmente. Obrigado por dia após dia me aturarem, me educarem e me proverem com tudo que há de melhor nesse mundo.

Obrigado à minha companheira de todas as horas. Obrigado por tanta paz, tanto carinho e tanto amor todos os dias.

Agradeço também a todos os meus amigos de ensino médio da querida turma 401 do IFRN, porque vocês merecem mesmo não tendo nada a ver com esse trabalho.

Devo agradecer também a todos que me ajudaram a adquirir tantos ensinamentos ao longo desses quatro anos. Sei o que sei hoje da área de computação graças a uma grande parcela de amigos, colegas e professores.

Obrigado especialmente à 4Soft e a todos que fizeram ou fazem parte dela. Não seria o profissional que sou hoje sem essa tão carismática (e problemática) empresa júnior. Foram três anos e meio de muita luta, aprendizados e ódio acumulado (por fatores que não vem ao caso) e que sem dúvida abriram meus olhos com relação ao mundo.

Um agradecimento especial também aos meus companheiros de curso que mais me ensinaram nesse tempo todo (e que certamente continuarão ensinando). Vou citá-los em ordem alfabética para que ninguém fique com ciúmes, ok? Obrigado Bernardo, Iago, Lucas, Luiz Rogério e Waldyr por tantas dicas trocadas, *pulls requests* revisados, dúvidas esclarecidas e trabalhos feitos.

Agradeço também a todos os professores do Departamento de Informática e Matemática Aplicada da UFRN, em especial ao professor Fernando, pelas infindáveis orientações e revisões deste trabalho. Obrigado também ao professor Nélcio Cacho por ter passado o trabalho mais desafiador na disciplina mais insana que paguei durante a graduação e que serviu de um amadurecimento enorme.

Obrigado também a qualquer um que se sinta válido de estar mencionado aqui mas acabou não sendo por algum motivo qualquer. Desculpa.

Obrigado a todos que torceram por mim.

Obrigado.

*Life moves pretty fast. If you don't stop and look around once in a while, you could miss
it.*

Ferris Bueller

Twydi: Uma Ferramenta de Documentação para Equipes de Desenvolvimento de Software

Autor: Igor Marques da Silva

Orientador: Prof. Dr. Fernando Marques Figueira Filho

RESUMO

Desenvolvedores de software constantemente trocam informações entre si para o aprimoramento contínuo de suas habilidades e resolução de problemas. Muitas dessas informações são perdidas devido ao uso de mídias que não oferecem bom suporte ao armazenamento, recuperação e visualização de informações.

Este trabalho apresenta uma ferramenta que permite aos desenvolvedores de agregar descrições textuais, código, referências externas em artefatos, além de possibilitar a recuperação destes. Este trabalho também descreve os estudos realizados ao longo do seu processo de desenvolvimento e avaliação. Apesar de melhorias no quesito de pré-visualização dos artefatos serem necessárias, a ferramenta atende aos requisitos de suporte a troca de conhecimento de desenvolvedores através do armazenamento do registro de soluções e compartilhamento de informações e referências para a equipe. Os estudos realizados neste trabalho, bem como as lições aprendidas durante o desenvolvimento da ferramenta desenvolvida em si, servem como base para trabalhos futuros de melhoria da qualidade de documentação e processos de troca de conhecimento em equipes de desenvolvimento de software.

Palavras-chave: Documentação de Software. Reuso de Software. Desenvolvimento de Software.

Twydi: A tool for Documentation for Software Development Teams

Author: Igo Marques da Silva

Advisor: Prof. Dr. Fernando Marques Figueira Filho

ABSTRACT

Software developers exchange information all the time to improve skills and solve problems. Many of this informations are lost due to the use of medias that don't offer support to information storage, recovery and visualization.

This work presents a tool that allows developers to combine text, code and external references to artifacts and later its retrieving. It also presents studies during its development and evaluation. Even though improvements related to artifacts preview be needed, it supports knowledge sharing between developers through the storage of solutions, information and references. The study, as well as the lessons learned through the development of the tool, provides a startup point for future works on the improvement of the quality of documentation processes and knowledge sharing in software teams.

Keywords: Software Documentation. Software Reuse. Software Development.

Lista de figuras

1	Exemplo de arquivo “Leia-me”	p. 25
2	Dúvida do tipo 1. Como	p. 29
3	Dúvida do tipo 2. Tomada de decisão	p. 29
4	Dúvida do tipo 3. Revisão de Código	p. 29
5	Dúvida do tipo 4. Erro	p. 30
6	Dúvida do tipo 4. Erro (continuação)	p. 30
7	Dúvida do tipo 5. Discrepância	p. 31
8	Listagem de Documentos	p. 36
9	Listagem de Documentos (parte inferior da página)	p. 36
10	Lista de <i>tags</i>	p. 37
11	Novo documento	p. 37
12	Exibição de documento	p. 39
13	Importação de Arquivo	p. 41
14	Importação de Linha de Arquivo	p. 42
15	Importação de Múltiplas Linhas de Arquivo	p. 42
16	Importação de <i>Commit</i>	p. 42
17	Importação de <i>Pull Request</i>	p. 43
18	Exibição de documento (início)	p. 44
19	Exibição de documento (meio)	p. 45
20	Exibição de documento (fim)	p. 45
21	Lista de documentos de um usuário	p. 47
22	Visualização dos “Curtir” em um documento	p. 48

23	Visualização dos “Curtir” na lista de documentos	p. 48
24	Dúvida do tipo 1	p. 61
25	Dúvida do tipo 1 (continuação)	p. 62
26	Dúvida do tipo 2	p. 63
27	Dúvida do tipo 5	p. 64

Lista de tabelas

- 1 Características das mídias de comunicação (adaptado de Olson et. al, 2000) p. 20
- 2 Tipos de importação suportadas pela aplicação p. 40

Sumário

1	Introdução	p. 15
1.1	Exemplo de Uso da Ferramenta	p. 17
1.2	Objetivos e Perguntas de Pesquisa	p. 17
1.3	Estrutura do Trabalho	p. 18
2	Revisão de Literatura	p. 19
2.1	Uso das mídias pelo engenheiro de software	p. 19
2.2	Rotatividade de pessoal	p. 21
2.3	Gestão e recuperação de informação	p. 21
2.4	Reuso em engenharia de software	p. 22
2.5	GitHub	p. 22
3	Metodologia	p. 24
3.1	Estudo de Aplicações Existentes	p. 24
3.1.1	Evernote	p. 24
3.1.2	Wikis	p. 25
3.1.3	GitHub e GitHub Gists	p. 25
3.1.4	Blogs e sites	p. 26
3.2	Inquérito contextual	p. 27
3.2.1	Contexto	p. 27
3.2.2	Inquérito	p. 28
3.3	Definição dos Requisitos da Ferramenta e Implementação	p. 33

3.4	Implantação, Observação do uso da Ferramenta e Aplicação de questionário	p. 33
4	Projeto de Solução	p. 35
4.1	Listagem de Documentos	p. 35
4.2	Criação de um Documento	p. 37
4.2.1	Elementos de um documento	p. 38
4.2.2	Implementação	p. 38
4.2.3	Suporte a <i>Markdown</i>	p. 43
4.2.4	Exibição do Documento	p. 44
4.3	Recuperação de Documentos	p. 46
4.3.1	Através de <i>Tags</i>	p. 46
4.3.2	Através de Conteúdo dos Atributos	p. 46
4.4	Autenticação com GitHub	p. 47
4.5	“Curtir” Documentos	p. 47
4.6	Limitações da Ferramenta	p. 49
5	Análise e Discussão dos Resultados	p. 50
5.1	Avaliação da ferramenta	p. 50
5.1.1	Sobre o que era seu twydi?	p. 50
5.1.2	No seu dia-a-dia de desenvolvimento, como você resolve os problemas pelos quais você passa ou como descobre como implementar algo novo pela primeira vez?	p. 50
5.1.3	Com relação aos Twydis, qual a utilidade que você vê nos <i>links</i> relacionados?	p. 51
5.1.4	O que você mudaria na forma como os links relacionados estão implementados atualmente?	p. 51
5.1.5	E qual a utilidade que você vê nas tags?	p. 51
5.1.6	O que você mudaria na forma como as tags estão implementados atualmente?	p. 52

5.1.7	Que importações de código você utilizou?	p. 52
5.1.8	Que importação de código você achou mais útil? Por quê? . . .	p. 52
5.1.9	Algum resultado da importação de código ficou ruim/ confuso? Por quê?	p. 53
5.1.10	No caso da importação de <i>commits</i> e <i>pull requests</i> , é mais impor- tante:	p. 53
5.1.11	Como você utilizaria a ferramenta no seu dia-a-dia?	p. 53
5.1.12	O que pode ser melhorado na ferramenta para facilitar a escrita de documentos?	p. 54
5.1.13	Além dos campos já presentes em um Twydi, que outras informa- ções você acrescentaria?	p. 54
5.1.14	Houve alguma dificuldade para o preenchimento deste questioná- rio ou para o entendimento das atividades solicitadas?	p. 54
5.2	Conclusões Obtidas e Respostas das Perguntas de Pesquisa	p. 54
5.2.1	1. A ferramenta proposta oferece suporte à maneira como conhe- cimento é trocado em equipes de software?	p. 54
5.2.2	2. Que outras funcionalidades podem ser agregadas a ferramenta para melhorar este suporte, caso exista?	p. 55
5.2.3	3. Quais são os casos potenciais de uso da ferramenta segundo os desenvolvedores?	p. 55
5.3	Limitações do Estudo	p. 55
6	Considerações Finais	p. 56
	Referências	p. 57
	Apêndice A – Questionário	p. 59
	Anexo A – Outros exemplos de Dúvidas das Equipes	p. 61

1 Introdução

Trabalhadores do conhecimento necessitam estar em constante aprendizado, de modo a proporcionar um melhor desempenho para toda organização (DRUCKER, 1993). Desenvolvedores de software se enquadram neste conjunto e constantemente necessitam reutilizar o conhecimento aprendido por outros desenvolvedores (WIIG; JOOSTE, 2004). Assim, é esperado que os desenvolvedores melhorem continuamente seu trabalho em um processo em culmina na melhora significativa da sua empresa (KAVITHA; AHMED, 2011). Essa melhoria constante dos membros de uma equipe está amplamente relacionada com o conceito de organizações que aprendem, onde seus membros estão em constante processo de melhoria e expansão de consciência e capacidades (SENGE, 2014).

Em equipes de desenvolvimento de software, o reuso de conhecimento ocorre utilizando uma variedade de artefatos, como código fonte, requisitos, modelos, dados e padrões (LEVY; HAZZAN, 2009), bem como através de interações face a face, comunicação escrita e também via o repasse de referências de documentação, links, dentre outros (STOREY et al., 2014) (OLSON; OLSON, 2000) (CUBRANIĆ et al., 2004).

Neste quesito, entra a gerência de conhecimento, encarregada da elicitación, armazenamento, gerenciamento e reuso do conhecimento (LEVY; HAZZAN, 2009). Dentre as formas típicas de reuso, se encontra, por exemplo, como se deu a implementação de uma determinada funcionalidade em um determinado projeto de software.

A troca de informações é um mecanismo fundamental para que o reuso de conhecimento ocorra de maneira eficiente em equipes de desenvolvimento. Desenvolvedores, principalmente em fase de aprendizado de uma determinada tecnologia, comumente passam por situações e problemas semelhantes. Quando se deparam com algum tipo de adversidade, procuram por alguma fonte de informação capaz de auxiliá-los por tal problema. Existe então o desperdício de tempo de se recuperar tal informação (muitas vezes, interferindo nas atividades de um colega de trabalho) para se resolver um problema que já é de conhecimento da equipe dado alguma experiência anterior.

Com a falta de uma boa gerência de conhecimento, é comum desenvolvedores implementarem funcionalidades semelhantes em diferentes contextos usando abordagens *ad-hoc* (SANGMOK, 2011). Essas novas implementações, principalmente se feitas por desenvolvedores menos experientes, tendem a ser menos otimizadas e elegantes que soluções já previamente implementadas pela equipe, podendo gerar um débito técnico¹ a longo prazo.

Desenvolvedores mais experientes em um determinado projeto tendem a atuar como mentores (CUBRANIĆ et al., 2004) e tal ato, como atividade de gerência de conhecimento, tende a consumir recursos como o tempo dos citados mentores (WIIG; JOOSTE, 2004). Um dos pilares da gerência de conhecimento é a melhoria da produtividade através do compartilhamento e transferência eficiente de conhecimento, que tende a consumir tempo e às vezes sendo até impossível de se realizar (LEVY; HAZZAN, 2009).

Muitas das interações entre desenvolvedores são informais e não há registro que possa ser consultado para proporcionar o reuso do conhecimento trocado (OLSON; OLSON, 2000). Um agravante típico é a frequente rotatividade de membros em equipes. Nesses casos, a saída de um membro da equipe que detém determinado conhecimento pode prejudicar toda organização. De fato, a falta de uma abordagem mais sistemática para proporcionar o reuso de conhecimento pode estar associada ao fracasso de projetos em organizações (HALL et al., 2008).

Assim, a elaboração de uma ferramenta capaz de agregar referências de código a soluções pode trazer enormes benefícios a equipes de desenvolvimento (CUBRANIĆ et al., 2004). Este estudo propôs a elaboração de tal ferramenta baseando-se principalmente nas limitações existentes em ferramentas de propósito similar. Esta atua como um catálogo, agregando referências de código, informações externas e comentários fornecidos por desenvolvedores da equipe com o intuito de auxiliar outros desenvolvedores a buscar em fontes da própria equipe (ou de confiança da mesma) como se deram implementações de funcionalidades já feitas em outros projetos.

Vale reforçar que a ferramenta procura melhorar a gestão de conhecimento explícito, ou seja, aquele que pode ser registrado de alguma maneira.

¹Metáfora para as eventuais consequências de uma implementação não apropriada de determinada tarefa e que pode causar problemas para a manutenção do software no futuro.

1.1 Exemplo de Uso da Ferramenta

Um determinado desenvolvedor João, experiente dentro de sua equipe, percebe que muitos membros novatos procuram sua ajuda para implementar uma funcionalidade de exportação de uma página web para um documento em formato PDF. João então pode fazer uso da ferramenta para registrar como se realiza a implementação de tal funcionalidade solicitada. Ele informa um título, descrição curta (para facilitar futuras buscas) e uma descrição de como se dá tal implementação. Como exemplificação em formato de código, o desenvolvedor pode utilizar de links de trechos de código disponíveis artefatos de código já implementados pela sua equipe. Esses links são renderizados no editor de texto, sem a necessidade de João copiar e colar o código em si dentro do editor. João também pode complementar sua descrição adicionando links para outras páginas web (respostas de sites como Stack Overflow, outros tutoriais, etc...) em forma de anexo da documentação. Por fim, João informa *tags* relevantes ao artefato que está produzindo, facilitando a sua recuperação por parte de outros membros de sua equipe.

Outros desenvolvedores, a partir de então, podem recorrer diretamente a ferramenta quando precisarem implementar a funcionalidade de exportação de página em formato PDF. Dessa forma, João poderá se dedicar mais a outras atividades de seu dia-a-dia de trabalho.

1.2 Objetivos e Perguntas de Pesquisa

Este estudo descreve a elaboração de tal ferramenta com o objetivo de melhorar o compartilhamento de informação em equipes de desenvolvimento de software e sua avaliação em dois contextos. O primeiro é o de uma equipe real de desenvolvimento de software composta por alunos da Universidade Federal do Rio Grande do Norte e o segundo contexto é o de alunos de graduação da área de computação da mesma universidade enquanto cursavam a disciplina de Desenvolvimento Colaborativo de Software no segundo semestre de 2015. A ferramenta foi utilizada para permitir que os alunos se auxiliassem durante o desenvolvimento do projeto final de disciplina.

O objetivo geral deste trabalho é responder as seguintes perguntas:

1. A ferramenta proposta oferece suporte à maneira como conhecimento é trocado em equipes de software?

2. Que outras funcionalidades podem ser agregadas a ferramenta para melhorar este suporte, caso exista?
3. Quais são os casos potenciais de uso da ferramenta segundo os desenvolvedores?

Para responder as perguntas de pesquisa propostas, foi aplicado um questionário com usuários de ferramenta.

Concluiu-se que, para a pergunta de pesquisa 1, sim, a ferramenta oferece suporte à troca de conhecimento entre desenvolvedores. Para a pergunta de pesquisa 2, um melhor suporte a pré-visualização e formatação e dos documentos pode aprimorar o suporte existente. Para a pergunta de pesquisa 3, os participantes informaram que a divulgação de tutoriais ou soluções para suas equipes e registro pessoal sobre como o próprio desenvolvedor solucionou determinado problema seriam casos de uso nos quais utilizariam a ferramenta em sua rotina.

1.3 Estrutura do Trabalho

O Capítulo 2 apresenta uma revisão da literatura relacionada a este trabalho, focando no uso de mídias pelo engenheiro de software, os efeitos da rotatividade em equipes de software, além de gestão e recuperação de informação. O Capítulo 3 explica em detalhes os procedimentos tomados para a construção e análise da ferramenta tendo como base os conhecimentos citados no capítulo anterior. O Capítulo 4 descreve em detalhes a ferramenta elaborada. O Capítulo 5 realiza a análise dos resultados dos experimentos bem como discute seus resultados. O Capítulo 6 aborda as considerações finais sobre o trabalho.

2 Revisão de Literatura

Este capítulo trata das referências utilizadas como base para o trabalho nas áreas de: uso de mídias pelo engenheiro de software, rotatividade de pessoal, gestão e recuperação de informações e reuso.

2.1 Uso das mídias pelo engenheiro de software

A atividade de desenvolvimento de software requer que seus profissionais se mantenham constantemente atualizados, devido ao fluxo constante de inovações que surgem diariamente (SINGER; FILHO; STOREY, 2014).

Uma das formas que desenvolvedores utilizam para se manterem atualizados são as redes sociais (TREUDE et al., 2012) (STOREY et al., 2014) além de seus próprios colegas de trabalho, tendo em vista que desenvolvedores frequentemente recorrem a própria equipe para obter ajuda (WEINBERG, 1998).

Sobre as mídias em geral, Clark e Brennan (CLARK; BRENNAN, 1991) estabeleceram que estas podem apresentar as seguintes características.

- Copresença: está no mesmo ambiente físico.
- Visibilidade: está visível aos participantes da comunicação.
- Audibilidade: é possível de ser ouvida pelos participantes da comunicação.
- Contemporaneidade: a mensagem é recebida imediatamente.
- Simultaneidade: Ambos os participantes podem enviar e receber mensagens.
- Sequencialidade: turnos não podem sair de sequência.
- Revisibilidade: permite rever as mensagens enviadas.

- Revisabilidade: permite rever uma mensagem antes de ser enviada.

A comunicação com colegas de trabalho face-a-face comparado a outras mídias, provê uma série de vantagens, conforme apresentado na Tabela 1.

Tabela 1: Características das mídias de comunicação (adaptado de Olson et. al, 2000)

Mídia	Copresença	Visibilidade	Audibilidade	Contemporaneidade	Simultaneidade	Sequencialidade	Revisibilidade	Revisabilidade
Face-a-face	x	x	x	x	x	x		
Telefone			x	x	x	x		
Vídeoconferência		x	x	x	x	x		
Chat de duas vias				x	x	x	x	x
Secretária Eletrônica			x				x	
E-mail							x	x
Carta							x	x

fonte: Olson et. al, 2000 (adaptado pelo autor)

Apesar destas vantagens, a comunicação face-a-face não apresenta uma importante característica para a gestão do conhecimento de uma equipe: a revisibilidade. Tanto ela quanto a revisabilidade dão auxílio a ambos comunicadores permitindo que formulem cuidadosamente a mensagem que desejam transmitir, além de permitir várias chances de revisar o que já foi trocado (OLSON; OLSON, 2000). A revisibilidade, em especial, possui relação direta com a gestão e recuperação de informação. Informações já conhecidas pela equipe e já trocadas tem enorme valor, principalmente em equipes com alta rotatividade de pessoal.

Ferramentas de *chat*, por sua vez, apresentam a característica da revisabilidade, mas ainda sim possuem problemas de recuperação fácil de informações trocadas nela (variando de ferramenta para ferramenta). Informações cruciais são trocadas nessa mídia, mas muitas vezes são impossíveis de serem recuperadas devido ao alto volume de mensagens trocadas.

Dessa forma, mídias com um bom suporte a recuperação são de extrema importância para equipes de desenvolvimento de software.

2.2 Rotatividade de pessoal

Rotatividade de pessoal é inerente a qualquer área de atuação da indústria. Sua ocorrência acarreta em custos de transferência de conhecimento e treinamento (HALL et al., 2008), possuindo relação com sucesso ou fracasso de projetos, incluindo aqueles que envolvem o desenvolvimento de software (HALL et al., 2008).

Em equipes de desenvolvedores de software, a partida de um membro pode acarretar em perda de conhecimento muitas vezes precioso para a equipe em casos onde não haja nenhuma outra forma de registro em mídia de tais informações.

Em geral, novos membros da equipe passam por um período de adaptação onde devem aprender padrões, práticas e ferramentas utilizadas pelo grupo. Além disso, os novatos estão suscetíveis a outras barreiras como quebras de expectativas, problemas com recepção, má configuração de ambiente de trabalho e curva de aprendizado (STEINMACHER et al., 2015).

Sobre a curva de aprendizado, esta tem peso ainda maior em casos específicos como desenvolvedores iniciantes (em geral chamados de “desenvolvedores júnior”, no mercado), pois existe a necessidade em se adquirir conhecimento sobre as tecnologias utilizadas pela equipe ao qual foi recém integrado.

Todo esse conhecimento é em geral transmitido oralmente, por via escrita ou repasse de referências (documentação, links externos, etc) (STOREY et al., 2014; OLSON; OLSON, 2000; CUBRANIĆ et al., 2004). Conforme mencionado anteriormente algumas mídias não possuem persistência ou possuem métodos precários de recuperação.

Sendo assim, uma ferramenta que estimule seus usuários a registrarem informações fundamentais para a equipe tende a evitar tanto que informações sejam perdidas no caso de partidas quanto facilitar o aprendizado e adaptação de novos membros à equipe.

2.3 Gestão e recuperação de informação

A gestão de conhecimento tem por objetivos a aquisição de novos conhecimentos, como lidar com o conhecimento existente para garantir seu uso futuro, passando pelo seu armazenamento e difusão, bem como provendo estratégias aplicáveis para novos contextos (BJØRNSON; DINGSØYR, 2008).

Uma boa gestão de informação e facilidade de se recuperar informações permitem

que organizações de desenvolvimento de software se mantenham inovadoras e competitivas (RABELO et al., 2015).

É possível ver então que uma equipe de desenvolvimento que apresente alguma forma de gestão de informações pode garantir, facilitar e estimular, por exemplo, o reuso de soluções já conhecidas de problemas recorrentes.

2.4 Reuso em engenharia de software

Possibilidade de reuso de software é uma das características-chaves de um código de qualidade. Reuso de software pode ser definido como o processo de construir sistemas de software a partir de software já existente em vez de “do zero” (KRUEGER, 1992).

Vale lembrar que os tipos de artefatos que podem ser reusados não estão limitados somente a código-fonte, mas também podem incluir *design* de estruturas, implementação de módulos, especificações, documentação e etc (FREEMAN, 1983).

Diversos cientistas da computação ainda vêem o reuso como uma potencial meio para melhorar as práticas da engenharia de software, além de suas vantagens serem amplamente conhecidas (KRUEGER, 1992).

É notável então que uma boa forma de garantir a gerência de informações visando o reuso de conteúdo produzido por uma equipe pode trazer melhorias para a qualidade dos projetos produzidos pela mesma.

2.5 GitHub

GitHub¹ é um serviço web de repositórios de código utilizando primariamente Git como controle de versão (FIGUEIRA FILHO et al., 2015). Atualmente possui cerca de onze milhões e meio de desenvolvedores e vinte e oito milhões de repositórios cadastrados².

O que destaca o GitHub das demais ferramentas de repositório online é o grande apelo deste para a disponibilidade de projetos de código aberto.

É comum que empresas de desenvolvimento de software criem organizações dentro do GitHub. Organizações funcionam como um agregado de usuários com acesso comum aos mesmos repositórios. Cada usuário continua mantendo uma conta própria para si.

¹<http://github.com>

²<https://github.com/about/press>

Por ser o maior centro de hospedagem de código da atualidade (GOUSIOS; SPINELLIS, 2012), se mostra uma excelente plataforma para se obter exemplos de código de diversas linguagens e finalidades a serem compartilhados entre desenvolvedores.

3 Metodologia

Este trabalho prevê o desenvolvimento da ferramenta de documentação de implementação funcionalidades. Este capítulo aborda principalmente as etapas de estudo de aplicações existentes, inquérito contextual e elicitação de requisitos.

3.1 Estudo de Aplicações Existentes

Foi feita uma busca por aplicações que realizem atividades semelhantes a ferramenta proposta, ou seja, criação e compartilhamento de notas textuais. O critério de seleção das ferramentas foi a popularidade entre usuários, principalmente entre desenvolvedores de software.

3.1.1 Evernote

Na época da elaboração deste trabalho, o Evernote¹ era uma das ferramentas mais populares de criação de anotações privadas. Em 2014, atingiu a marca de mais de 100 milhões de usuários². Ele possui um editor de texto com suporte a texto formatado, com listas, *checkboxes*, *links*, anexos, tabelas. Os documentos criados na ferramenta (que possui clientes web, desktop e para diversas plataformas móveis) podem receber *tags*, e ser colocados em pastas para melhor organização de modo a facilitar sua recuperação.

Documentos também podem ser compartilhados diretamente para outros usuários da ferramenta ou através de *links* públicos (acessíveis para qualquer pessoa).

Dentre suas limitações para o uso para a redação de tutoriais da área de desenvolvimento de código está a falta de suporte a exibição de código com *syntax-highlighting*, ou seja, com formatação apropriada.

¹<https://www.evernote.com/>

²<https://blog.evernote.com/blog/2014/05/13/evernote-reaches-100-million-users/>

3.1.2 Wikis

Editadas e estruturadas colaborativamente, *Wikis* são comumente utilizadas em equipes de desenvolvimento de software para o registro e transmissão de conhecimento. Muitas oferecem suporte a exibição de código e linguagem de marcação.

Com diversas possibilidades de uso, muitas vezes apresentam problemas na recuperação de informação devido a falta de catalogação e indexação precária por categorias ou assuntos (dependendo em certos casos que isso seja feito pelos próprios usuários).

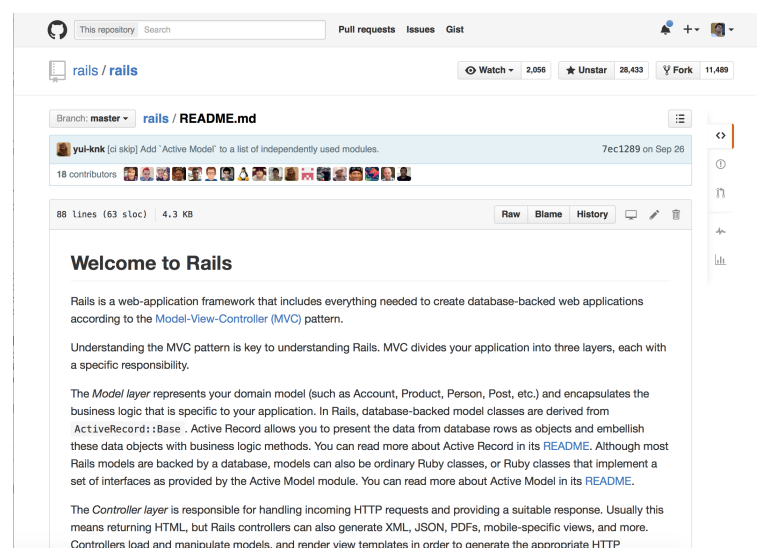
3.1.3 GitHub e GitHub Gists

Apesar de não ser sua principal proposta, o GitHub é muito utilizado por desenvolvedores para anotações, por possuir suporte a uma linguagem de marcação própria, chamada GitHub Flavored Markdown³. Esta linguagem funciona de maneira semelhante a outras do mesmo tipo como HTML ou LaTeX, permitindo que seus usuários escrevam textos apenas informando que determinadas seções do texto possuem tal significado como “Título”, “Lista”, entre outros.

Estas anotações normalmente são utilizadas em arquivos “Leia-me” de projetos e descrevem informações relevantes sobre estes (instalação, como executar testes, como fazer uso do projeto, etc).

³<https://help.github.com/articles/github-flavored-markdown/>

Figura 1: Exemplo de arquivo “Leia-me”



Fonte: GitHub (2015)

Outra forma de se registrar anotações no GitHub é através de Gists⁴. A comunidade de desenvolvimento de software comumente utiliza estes arquivos para registrar pequenos *snippets* ou *scripts* de código, com pouca ou nenhuma descrição sobre o mesmo.

Gists pertencem a somente um usuário (podendo ser públicos ou privados) e também possuem suporte a GitHub Flavored Markdown.

Vale notar que esta linguagem de marcação possui suporte a *syntax-highlighting*, formatando código de maneira clara.

Dentre as limitações das ferramentas acima citadas, está a precária indexação (repositórios do GitHub e Gists não possuem *tags* ou qualquer outra forma de catalogação, por exemplo) dificultando a recuperação de informação relevante ao usuário.

3.1.4 Blogs e sites

Uma das maneiras mais populares por desenvolvedores de se compartilhar tutoriais sobre questões de desenvolvimento são os *blogs* de autoria própria.

Implementados em diversas tecnologias, possuem a vantagem de muitas vezes estarem indexados por ferramentas de busca como Google⁵ e Bing⁶.

Além disso, existem uma série de *plugins* em diversas tecnologias que permitem a escrita de texto tanto em linguagens de marcação como WYSIWYG (*What You See Is What You Get*, ou seja, “O que você vê é o seu resultado”), se assemelhando a editores de texto como Microsoft Word⁷ ou Pages⁸. Alguns destes *plugins* possuem inclusive suporte a visualização de código com *syntax-highlighting*.

Porém, muitas vezes estes *plugins* possuem problemas para renderizar código que foi copiado de alguma outra fonte (como de arquivos de algum repositório do GitHub, por exemplo). Por exemplo, o *plugin* utilizado no website de perguntas e respostas de programação Stack Overflow exige que todo texto que representa código seja indentado utilizando quatro espaços. Códigos de aplicações desenvolvidas na linguagem *Ruby*, por convenção de estilo, geralmente estão indentados utilizando somente dois espaço. Assim, quando se copia e cola código já pronto desta linguagem, o editor de texto exibe um aviso exigindo a formatação correta do código para renderização correta. O redator então deve,

⁴<https://gist.github.com>

⁵www.google.com

⁶www.bing.com

⁷<https://products.office.com/en-us/word>

⁸<http://www.apple.com/mac/pages/>

linha por linha, alterar a quantidade de espaços dados, demandando de esforço e tempo.

Dessa forma, levou-se em consideração as limitações e pontos fortes de cada uma das ferramentas analisadas durante a elaboração da ferramenta deste trabalho.

3.2 Inquérito contextual

A partir da etapa anterior, o estudo para a concepção da ferramenta passou a levar em consideração requisitos e necessidades de desenvolvedores de software. O processo, que consistiu na análise de perguntas feitas por equipes, está descrito a seguir.

3.2.1 Contexto

Para a elaboração da ferramenta, houve participação da equipe de desenvolvedores da empresa júnior⁹ 4Soft¹⁰. A empresa é vinculada aos cursos de Bacharelado em Engenharia de Software¹¹ e Bacharelado em Tecnologia da Informação¹² da Universidade Federal do Rio Grande do Norte (UFRN)¹³. A empresa atua na área de desenvolvimento de software *web* para clientes de diversos ramos (em geral pequenos e médios negócios) e é formada exclusivamente por alunos dos cursos citados.

O autor do trabalho atua na empresa há três anos e meio, atuando em atividades de gestão de equipes e mentoria de novos membros, além de na própria área de desenvolvimento de software. Nesse período, passaram pela empresa cerca de dezessete desenvolvedores com nível de conhecimento variado. Participaram do estudo oito desenvolvedores com experiência em média de dois anos na empresa.

Além disso, foram analisadas as conversas no *chat* utilizado por uma turma de Desenvolvimento de Sistemas Web ofertada no primeiro semestre de 2014 para alunos dos cursos de Bacharelado em Engenharia de Software e Bacharelado em Ciência da Computação. A turma possuía um total de vinte e cinco alunos e seu projeto final envolvia a participação de todos colaborativamente sobre um mesmo projeto. A maioria dos alunos possuía pouca ou nenhuma experiência com a tecnologia utilizada na disciplina.

⁹http://en.wikipedia.org/wiki/Junior_enterprise

¹⁰<http://www.4softjr.com.br>

¹¹<http://www.dimap.ufrn.br/pt/graduacao/engenharia-de-software/apresentacao>

¹²http://www.imd.ufrn.br/curso_bacharelado.php

¹³<http://www.ufrn.br>

3.2.2 Inquérito

Entrevistas não estruturadas e seções de *brainstorm* foram feitas pelo autor com cinco participantes da empresa júnior mencionada. Além disso, por ser membro da empresa, o autor pôde observar o funcionamento interno da equipe. Procurou-se entender como se dava o fluxo da resolução de uma dúvida que algum desenvolvedor apresentava.

Foi feita a análise caso a caso das perguntas postadas no *chat* utilizado pela empresa no último ano bem como o *chat* utilizado pelos alunos da disciplina de Desenvolvimento Web. Pôde-se concluir que as dúvidas postadas geralmente se enquadravam em alguma das seguintes categorias:

1. **Como.** Instruções sobre implementar uma determinada funcionalidade ou ação.
2. **Tomada de decisão.** Qual a melhor maneira de realizar determinada implementação.
3. **Revisão de código.** Revisão de alguma implementação já feita.
4. **Erro.** Ajuda com a resolução de algum *bug* com mensagem de erro claramente fornecida.
5. **Discrepância.** Solicitação de explicação sobre comportamento inesperado apresentado pelo código.

Trechos de conversas extraídas do *chat* da empresa júnior estão presentes nas figuras 2 a 7 e exemplificam as categorias extraídas. Todos os participantes das conversas foram anonimizados utilizando cores. Os quadrados encontram-se sob avatares dos participantes enquanto os retângulos menores se encontram sob seus nomes de usuário. Um mesmo desenvolvedor pode apresentar diferentes cores em diferentes exemplos.

Figura 2: Dúvida do tipo 1. Como

October 1st

1:24 PM
@channel: acho que ja perguntei aqui, mas nao lembro da resposta...

Tenho uma entidade x que tem vários y
quero somar o retorno todos os métodos soma_doida de todos os y de jeito bolado

1:25 PM
added a Ruby snippet ▾

```

1  def people_reached
2    result = 0;
3    email_statistics.each do |e|
4      result += e.contacts_count
5    end
6
7    result
8  end

```

1:25 PM
fazer isso sem ser por um laço (eu acho que tem como)

2:09 PM
:

```
email_statistics.map(&:contacts_count).inject(&:+)
```

é um laço mas é oneliner
kkkk

2:09 PM
isssso
valeu,

Fonte: Adaptado de Slack (2015)

Figura 3: Dúvida do tipo 2. Tomada de decisão

October 26th

11:03 AM
@channel: qual o melhor tipo pra se usar quando eu quero armazenar só uma data? (só tipo 11/10/2015)

11:14 AM
Date

11:14 AM
jôia, valeu!

Fonte: Adaptado de Slack (2015)

Figura 4: Dúvida do tipo 3. Revisão de Código

August 22nd

11:48 PM
Esse meu application helper ta ok ou poderia ser mais bolado, @channel?

link para um arquivo em um projeto do GitHub

11:57 PM
👍

Fonte: Adaptado de Slack (2015)

Figura 5: Dúvida do tipo 4. Erro

October 8th

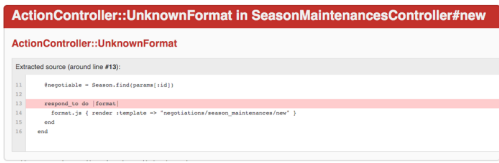
2:00 PM

@channel: alguém ja teve um problema assim? <https://www.dropbox.com/s/o23czwquon3ohfq/Screenshot%202015-10-08%2014.00.06.png?dl=0>

Dropbox

Screenshot 2015-10-08 14.00.06.png

Shared with Dropbox (32KB) ▾



```

ActionController::UnknownFormat in SeasonMaintenancesController#new
ActionController::UnknownFormat
Extracted source (around line #13):
13   respond_to do |format|
14     format.js { render :template => 'separations/season_maintenance/new' }
15   end
16 end

```

sendo que eu tenho um outro controller igualzinho mas que funciona

2:01 PM

Sei não

2:02 PM

ah

acho que o problema pode ser no botao que acessa isso

ver aqui

2:03 PM

assim ele soh vai responder a js

nao a json

2:03 PM

eu quero abrir um modal

Fonte: Adaptado de Slack (2015)

Figura 6: Dúvida do tipo 4. Erro (continuação)

2:04 PM

hm, ve se ta chamando certo

2:04 PM

era isso mesmo

faltava remote: true,

nao renderizou o modal, mas isso já é outra história

Fonte: Adaptado de Slack (2015)

Figura 7: Dúvida do tipo 5. Discrepância

October 19th

3:52 PM
@channel: alguém já teve problema com envio de email do rails sempre enviando o msm email ao invés de enviar diferentes emails por contexto?

(tenho variáveis que deveriam ser diferentes em cada email mas ele tá mandando tudo igual sempre, msm no LAÇO eu vendo que as variáveis são diferentes)

3:52 PM
nunca cheguei a mexer muito com envio de email 😞

3:53 PM
mostra o código

3:54 PM
added a Ruby snippet ▾

```
1 class ContactMailer < ApplicationMailer
2
3   def send_mail(body, to, title)
4     @body = replace_variables(body, to)
5   end
end
```

3:54 PM
se eu dou puts nesses to aí
cada um de fato é diferente do outro
mas o @title e @body são sempre ficando iguais

4:01 PM
bota um binding.pry antes do return
e veja o q dá

4:14 PM
todos são vindo iguais

: se eu imprimo o to, ele mostra certo
mas o text errado

Fonte: Adaptado de Slack (2015)

Estas categorias se enquadram dentro das dez categorias também encontradas por Treude et al. (TREUDE; BARZILAY; STOREY, 2011) em seu estudo sobre como desenvolvedores realizam perguntas no site de perguntas e respostas Stack Overflow. No estudo original foram encontradas um total de onze categorias. Optou-se apenas por selecionar apenas as cinco categorias em questão devido ao fato de as demais não terem sido encontradas com frequência dentro do conjunto de dúvidas estudado.

Além da análise das perguntas realizadas pelos desenvolvedores, foi feita uma análise sobre como estas eram respondidas. Para todos os casos citados, o fluxo de resolução do problema se dava na maioria dos casos da seguinte forma:

- i O desenvolvedor comunicava o problema (para todos os membros ou somente um) no *chat* utilizado pela equipe.
- ii Para todos os casos, um ou mais membros da equipe discutiam sobre como resolver a dúvida proposta e ofereciam uma solução. Em certos casos, exemplos de código eram fornecidos ou *links* relevantes eram enviados.
- iii O desenvolvedor que comunicou o problema tentava implementar a funcionalidade utilizando as recomendações oferecidas.
- iv Caso novas dúvidas surgissem durante a implementação, os demais membros da equipe tentavam dar recomendações mais precisas ou oferecer outros exemplos e *links* úteis.
- v O problema era solucionado.

Para casos em que a resolução problema encontrado não era de conhecimento de nenhum membro da equipe (seja por falta de contato com o contexto ou até mesmo por esquecimento), o desenvolvedor realizava pesquisas em ferramentas de busca para encontrar tutoriais ou fóruns de discussão que o ajudassem a resolver o problema.

Se tratando de esquecimento, desenvolvedores relataram que era comum que os próprios lembrassem que já enfrentaram determinado problema, mas não se lembram da solução que foi dada para tal. Isso era comum em especial em problemas do tipo “4. Erro”.

Exemplos de conversas que seguem este fluxo se encontram ao longo deste trabalho, bem como em seus anexos.

Observou-se também que era frequente que a mesma pergunta fosse feita por um outro desenvolvedor após certo tempo e respostas semelhantes eram oferecidas (e inclusive os mesmos *links* eram disponibilizados).

Desta etapa, concluiu-se que a ferramenta proposta deveria ser concebida levando em conta que:

- Exemplos de código da própria equipe são importantes para a resolução de uma dúvida.
- Um bom sistema de recuperação deve existir, pois desenvolvedores apresentam as mesmas dúvidas mais de uma vez.
- *Links* de tutoriais já prontos devem ser levados em consideração.

3.3 Definição dos Requisitos da Ferramenta e Implementação

Com base no Inquérito Contextual e Estudo de Aplicações Existentes, o autor pode elicitar os seguintes requisitos para uma ferramenta que diminua os esforços com mentoria/resolução de dúvidas de desenvolvedores:

- A ferramenta deve agir como um catálogo de anotações/ documentos feitos pela equipe.
- Deve haver o suporte a exibição de código (da própria equipe) e com formatação apropriada.
- Um bom sistema de recuperação de informações deve se fazer presente.
- Links de tutoriais já prontos devem ser levados em consideração.

Com base nisso, partiu-se para a implementação da ferramenta, denominada Twydi (um acrônimo para “*That’s the Way You Do It*”, em português: “É assim que se faz”). O detalhamento sobre este processo e das funcionalidades apresentadas por esta se encontram no próximo capítulo.

3.4 Implantação, Observação do uso da Ferramenta e Aplicação de questionário

A ferramenta então foi disponibilizada em dois servidores diferentes: um para uso da empresa júnior e de outros interessados e outro para uso dos alunos da disciplina de

Desenvolvimento Colaborativo de Software ocorrida no segundo semestre de 2015. Os artefatos gerados em um servidor não estavam disponíveis no outro. Ao todo, quatorze artefatos foram produzidos ao longo de três semanas por onze diferentes autores, sendo sete artefatos produzidos por quatro membros da empresa júnior e sete produzidos por sete membros da disciplina em questão.

Para avaliar a ferramenta, foi aplicado um questionário de satisfação. Ao todo o questionário possuía quinze perguntas e se encontra nos apêndices do trabalho. Foram obtidas oito respostas no questionário aplicado, sendo três por membros da empresa júnior e as demais por alunos da disciplina.

4 Projeto de Solução

Com base nos requisitos então elicitados, pôde-se ter uma noção clara de como seria feita a implementação da aplicação e este processo teve início. Ao longo dele, *feedback* a respeito da estética da ferramenta foi constantemente colhido com a equipe de desenvolvedores da empresa júnior, bem como de parceiros do grupo de pesquisa do qual o autor faz parte.

O intuito da ferramenta é permitir a criação de documento que vinculem recursos de repositórios no GitHub (arquivos, *commits* e *pull requests*), bem como outros referenciais (*links* para perguntas no Stack Overflow, tutoriais ou *blogs*, por exemplo) de forma a gerar um guia ou tutorial de como realizar tal implementação novamente no futuro. Dentro da ferramenta, um documento é chamado de *twydi*.

A ferramenta é de código livre e uma versão em execução pode ser acessada utilizada por qualquer desenvolvedor¹. Informações sobre licença e o código da ferramenta podem ser encontradas em seu repositório público².

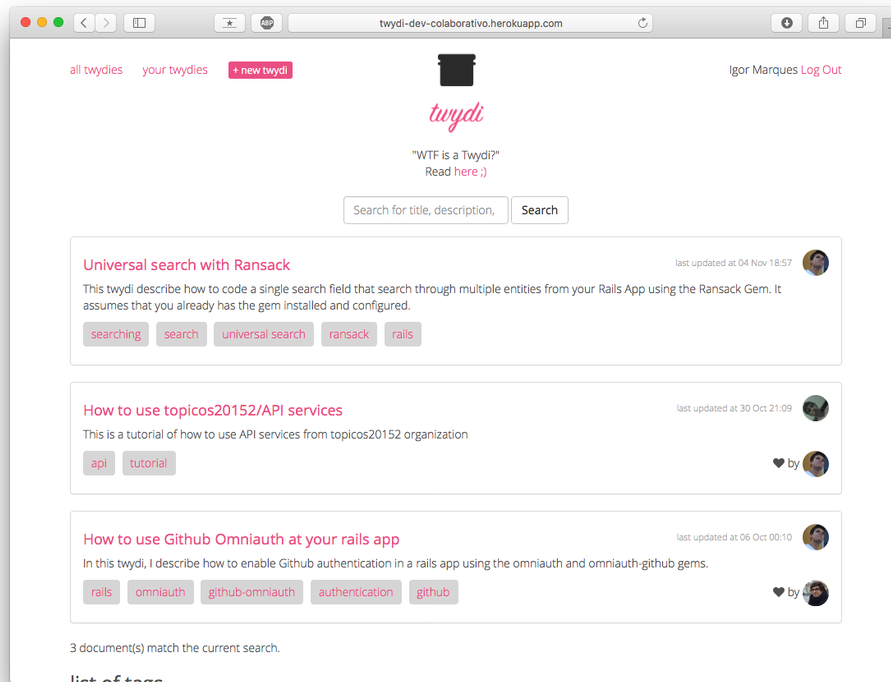
4.1 Listagem de Documentos

Na tela inicial da aplicação é possível ver uma lista com todos os documentos já criados. Cada documento é exibido com seu título, descrição, lista de *tags* e autor (caso tenha se identificado durante a criação do documento), conforme apresentado nas figuras 8 e 9.

¹<http://that-s-the-way-you-do-it.herokuapp.com>

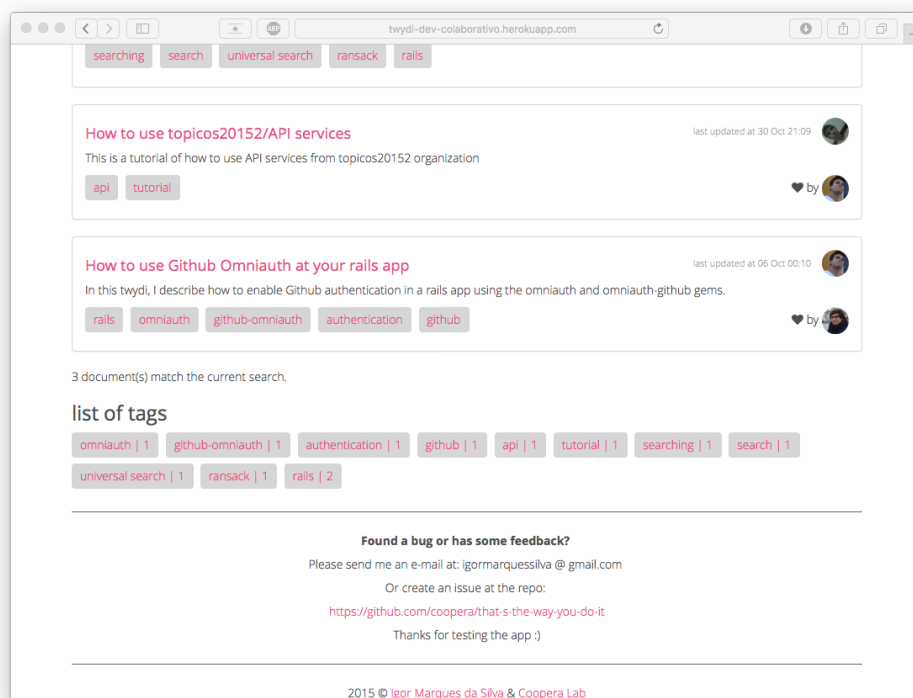
²<https://github.com/coopera/that-s-the-way-you-do-it>

Figura 8: Listagem de Documentos



Fonte: O autor (2015)

Figura 9: Listagem de Documentos (parte inferior da página)



Fonte: O autor (2015)

Ao fim da lista de documentos, é possível ver quantos documentos se encontram nela e também uma lista das *tags* presentes, incluindo sua quantidade de ocorrências dentro do grupo de documentos exibidos, como pode ser visto na figura 10.

Figura 10: Lista de *tags*

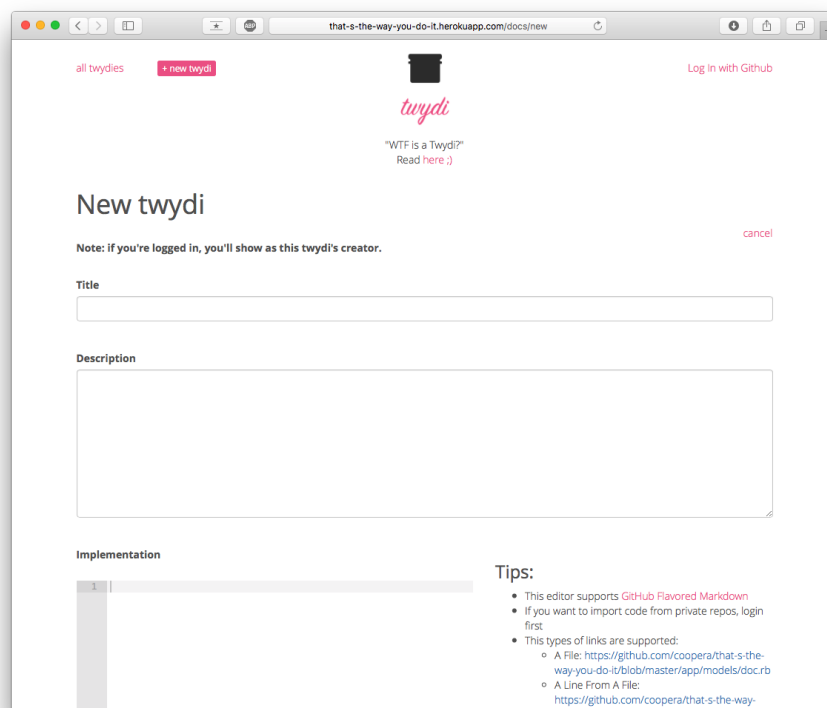


Fonte: O autor (2015)

4.2 Criação de um Documento

Qualquer usuário da aplicação pode criar um documento, mesmo sem estar autenticado na mesma. A figura 11 mostra a tela de criação de um documento.

Figura 11: Novo documento



Fonte: O autor (2015)

4.2.1 Elementos de um documento

Um documento é composto por 5 elementos, todos informados pelo seu autor:

- **Título:** Título do documento. Deve fornecer uma noção sobre o que aquele documento descreve.
- **Descrição:** Um resumo sobre o que o documento descreve. Descritivo o suficiente para que outros usuários não tenham que ler o documento inteiro para compreender sobre o que ele trata.
- **Implementação:** Procedimento sobre como implementar ou executar a funcionalidade descrita. Possui suporte a linguagem de marcação e importação de código armazenado em repositórios públicos e privados do GitHub.
- ***Links* Relacionados:** Lista de *links* relevantes para o documento, cada um com descrição própria. Podem conter referências utilizadas para elaborar o documento ou material mais avançado sobre o assunto.
- ***Tags*:** Lista de *tags* sobre o assunto do documento. Utilizada para facilitar sua recuperação posteriormente tanto pelo autor quanto por outros usuários.

Sobre o elemento Implementação, a próxima subseção o descreve em detalhes.

4.2.2 Implementação

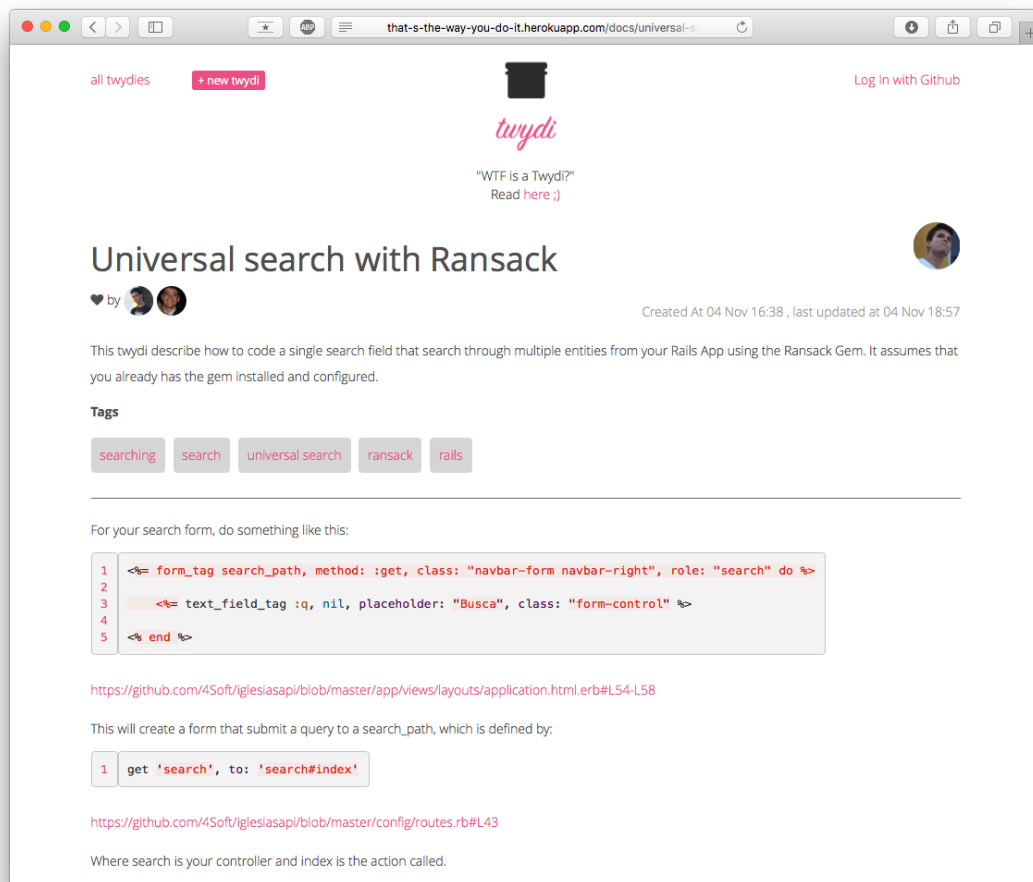
Elemento mais relevante de um documento e é utilizado para que o usuário descreva um passo-a-passo sobre como realizar a implementação de uma funcionalidade, ou deixe registrado determinado procedimento. Dentre suas funcionalidades que permitem um uso aprimorado por parte de equipes de desenvolvimento de software está a importação de código hospedado no GitHub.

No GitHub é possível acessar cada arquivo de um determinado projeto através de URLs simples e que seguem a própria estrutura de diretórios do projeto em questão.

Por exemplo, um arquivo com caminho “app/models/example.rb” se encontra URL “github.com/usuario/projeto/blob/master/app/models/example.rb”.

O arquivo é exibido utilizando uma formatação própria da linguagem, facilitando sua visualização. Um exemplo pode ser visto na figura 12.

Figura 12: Exibição de documento



Fonte: O autor (2015)

A partir dessa estrutura de URLs, foi elaborado um mecanismo para importar código no editor de texto da implementação toda vez que um *link* do GitHub é colado. Uma descrição sobre os tipos de importações possíveis e seus respectivos resultados encontra-se na Tabela 2.

Tabela 2: Tipos de importação suportadas pela aplicação

Tipo da Importação	Exemplo de link	Resultado
Arquivo	https://github.com/coopera/that-s-the-way-you-do-it/blob/master/app/models/doc.rb	Conteúdo Inteiro do Arquivo é importado.
Linha de Arquivo	https://github.com/coopera/that-s-the-way-you-do-it/blob/master/app/models/doc.rb#L1	Apenas a linha informada é importada.
Intervalo de Linhas de Arquivo	https://github.com/coopera/that-s-the-way-you-do-it/blob/master/app/models/doc.rb#L5-L7	O conteúdo intervalo entre as inclusivo entre linhas é importado.
<i>Commit</i>	https://github.com/coopera/that-s-the-way-you-do-it/commit/d7f365db9777b4cb6e1c5799a2e431c58aaf3a19	Um <i>diff</i> contendo o conteúdo adicionado e removido no <i>commit</i> é importado.
<i>Pull Request</i>	https://github.com/coopera/that-s-the-way-you-do-it/pull/7	Um <i>diff</i> contendo o conteúdo adicionado e removido em todos os <i>commits</i> do <i>pull request</i> é importado.

Fonte: O autor

Exemplos dos resultados de cada importação podem ser vistos nas figuras 13, 14, 15 e 16.

Figura 13: Importação de Arquivo

Implementation

```

1  ``ruby
2  class Doc < ActiveRecord::Base
3    extend FriendlyId
4
5    friendly_id :title, use: :slugged
6
7    has_many :related_links
8    belongs_to :user
9    has_many :likes
10   has_many :users, through: :likes
11
12   accepts_nested_attributes_for :related_links, reject_if: :all_blank, allow_destroy
13
14   validates_presence_of :title, :description, :implementation
15
16   validates :title, length: { minimum: 5 }
17
18   acts_as_ordered_taggable
19
20   def has_been_liked_by_this_user(user)
21     likes.where(user_id: user).exists?
22   end
23
24   def last_likes(x = 4)
25     users.last(x).reverse
26   end
27
28   def other_likes(x=4)
29     likes.count - last_likes(10).count
30   end
31
32   .

```

Fonte: O autor (2015)

Figura 14: Importação de Linha de Arquivo

Implementation

```

1  ```ruby
2  class Doc < ActiveRecord::Base
3  ```
4  https://github.com/coopera/that-s-the-way-you-do-it/blob/master/app/models/doc.rb#L1

```

Fonte: O autor (2015)

Figura 15: Importação de Múltiplas Linhas de Arquivo

Implementation

```

1  ```ruby
2
3  has_many :related_links
4  belongs_to :user
5  ```
6  https://github.com/coopera/that-s-the-way-you-do-it/blob/master/app/models/doc.rb#L5

```

Fonte: O autor (2015)

Figura 16: Importação de *Commit***Implementation**

```

1  ```diff
2  class DocsController < ApplicationController
3  def index
4  | @q = Doc.ransack(params[:q])
5  | @docs = @q.result(distinct: true)
6  +
7  +   if params[:tag]
8  +     @docs = @docs.tagged_with(ActsAsTaggableOn::Tag.find(params[:tag]))
9  +   end
10 +
11 | @tags = ActsAsTaggableOn::Tag.all
12 | end
13 ```
14 ```diff
15 ```diff
16
17 -<%= link_to "/" do %>
18 +<%= link_to docs_path(tag: tag.id) do %>
19   <div class="small-tag">
20     <%= tag.name %> | <%= tag.taggings_count %>
21   </div>
22 ```
23 https://github.com/coopera/that-s-the-way-you-do-it/commit/d7f365db9777b4cb6e1c5799c

```

Fonte: O autor (2015)

Figura 17: Importação de *Pull Request***Implementation**

```

1  ```diff
2      langs = {
3          'py': 'python',
4          'c' : 'c',
5          'css' : 'css',
6          - 'html' : 'html'
7          + 'html' : 'html',
8          + 'php' : 'php',
9          + 'cpp' : 'c++',
10         + 'rs' : 'rust',
11         + 'exs' : 'elixir',
12         + 'scss' : 'sass',
13         + 'less' : 'less',
14         + 'ts' : 'typescript',
15         + 'go' : 'go',
16         + 'scala' : 'scala'
17     }
18  ```
19  https://github.com/coopera/that-s-the-way-you-do-it/pull/7

```

Fonte: O autor (2015)

Todo código importado para aplicação já vem formatado como código na linguagem de marcação utilizada. Caso o arquivo importado seja de uma extensão conhecida, é adicionada a marcação para código da linguagem referente àquela extensão. Isso diminui o trabalho do autor para a formatação do texto, permitindo que foque em outros aspectos do documento, como seu texto.

O código importado pode ser livremente modificado, tendo em vista que é texto-puro, permitindo que o autor faça modificações caso ache conveniente.

Além disso, o *link* original colado é mantido para que futuros usuários que consultem o documento possam acessar o código-fonte original disponível no seu respectivo repositório no GitHub.

Sobre a linguagem de marcação utilizada, a próxima subseção a descreve, bem como suas vantagens para o contexto.

4.2.3 Suporte a *Markdown*

Uma linguagem de marcação permite que os autores dos documentos possam formatar seus textos com cabeçalhos, listas, tabelas, entre outros elementos.

Markdown³ é uma linguagem de marcação amplamente utilizada na web atualmente. Possui sintaxe simples e é utilizada em campos textuais de sites e serviços amplamente

³<https://daringfireball.net/projects/markdown/>

utilizados por desenvolvedores de software como os já citados GitHub e Stack Overflow.

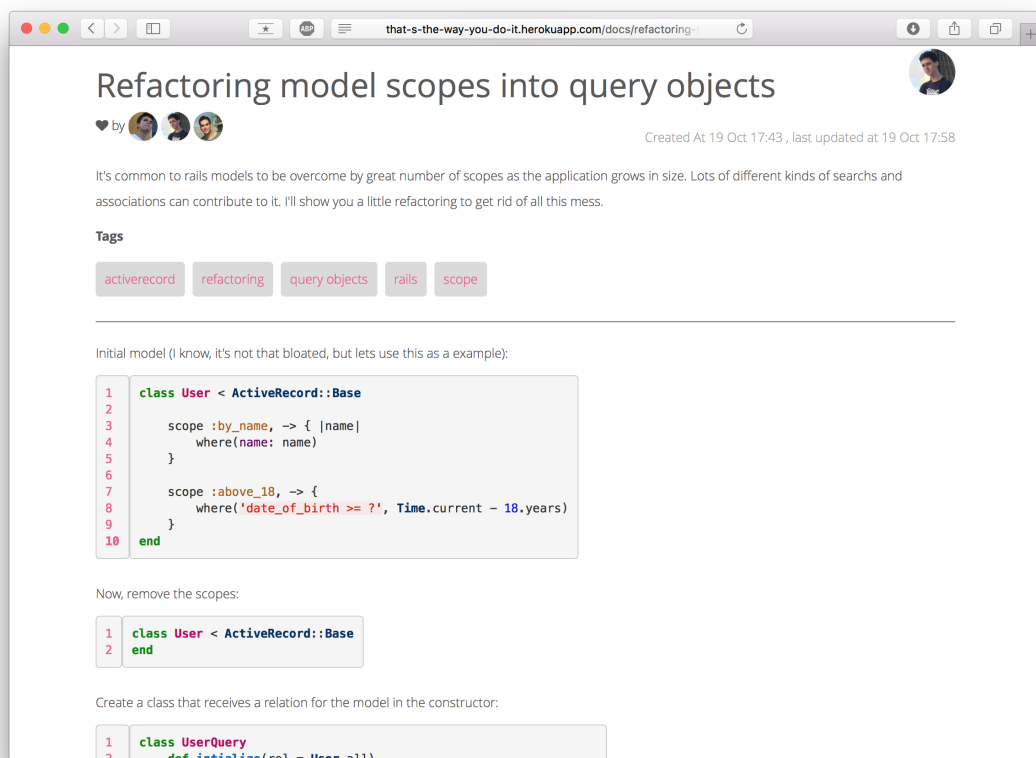
Assim, foi optado por adicionar suporte a tal linguagem devido a familiaridade dos desenvolvedores modernos com a mesma.

Dessa forma é permitido aos autores maior clareza e poder de expressão com relação aos documentos criados, além de garantir uma padronização em sua formatação (como de tipografia, por exemplo).

4.2.4 Exibição do Documento

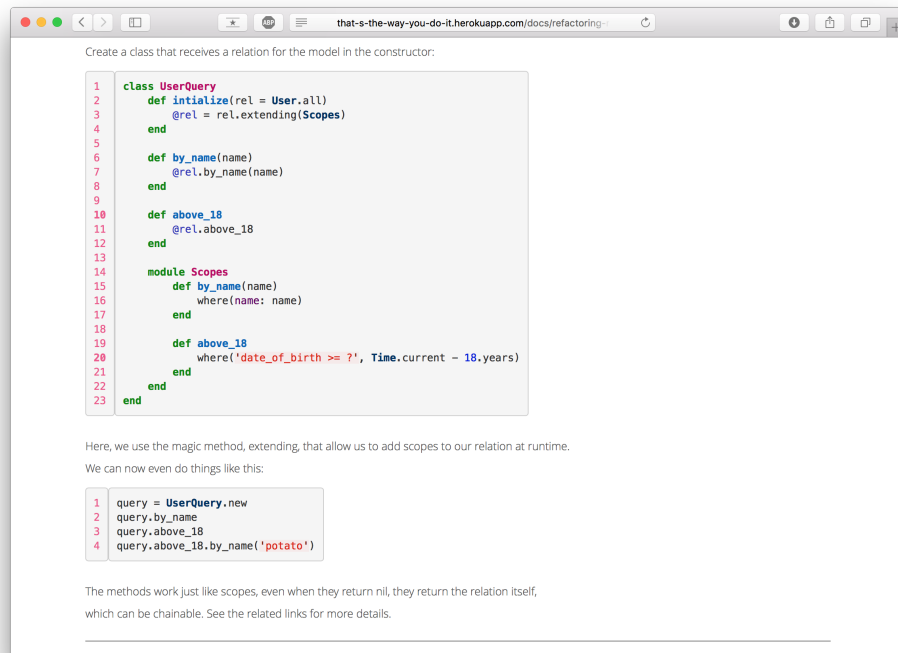
Uma vez que o documento é criado, sua visualização fica disponível para todos os usuários. As figuras 18, 19 e 20 mostram como um documento fica exibido dentro da aplicação.

Figura 18: Exibição de documento (início)



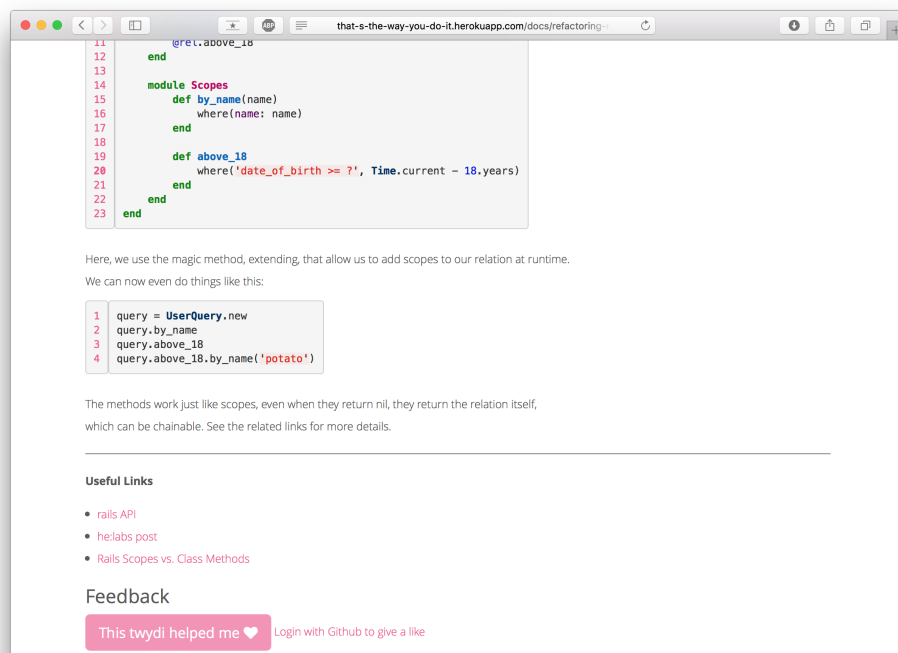
Fonte: O autor (2015)

Figura 19: Exibição de documento (meio)



Fonte: O autor (2015)

Figura 20: Exibição de documento (fim)



Fonte: O autor (2015)

Enfatiza-se que o elemento de implementação é propriamente renderizado de acordo com a marcação utilizada pelo autor.

4.3 Recuperação de Documentos

Todos os documentos da aplicação são acessíveis por qualquer desenvolvedor. Existem duas formas de fazer esta busca: via *tags* ou via busca textual por algum de seus elementos.

4.3.1 Através de *Tags*

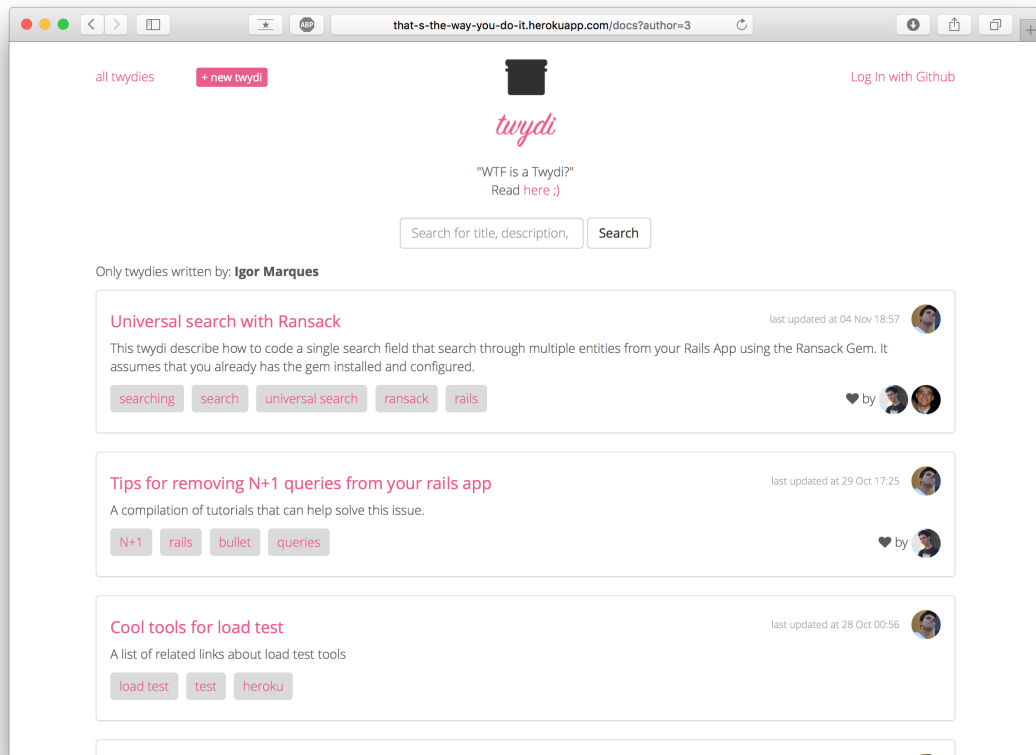
Ao se clicar em uma tag de um documento ou em uma tag presente na lista de tags que se encontra ao fim da lista de documentos, é possível se recuperar todos os documentos que apresentam aquela *tag*.

4.3.2 Através de Conteúdo dos Atributos

Utilizando a barra de busca que se encontra na página que lista todos os documentos da aplicação, é possível realizar uma busca textual que recupera documentos que apresentem o texto informado nos seguintes elementos: Título, Descrição, Implementação ou Links Relacionados.

Além disso, é possível visualizar todos os documentos criados por um usuário ao se clicar no avatar do mesmo em qualquer tela do sistema. Um exemplo da filtragem de documentos por usuário pode ser visualizado na figura 21.

Figura 21: Lista de documentos de um usuário



Fonte: O autor (2015)

4.4 Autenticação com GitHub

Para que desenvolvedores sejam identificados como autores dos documentos que criarem, é necessário que se autenticuem na aplicação com sua conta do GitHub. Isso também os permite importar código de repositórios privados para dentro da aplicação, tendo em vista que o GitHub exige autorização do próprio usuário para fornecer código privado.

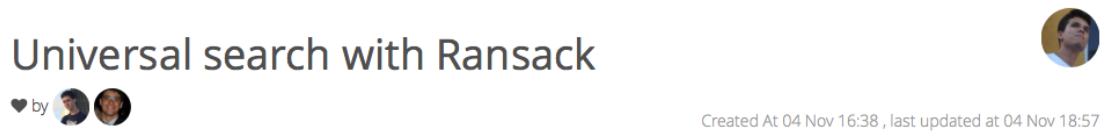
Além disso, usuários logados podem utilizar a funcionalidade de “Curtir” documentos, descrita na próxima seção.

4.5 “Curtir” Documentos

Usuários autenticados podem utilizar esta funcionalidade, que permite adicionar uma “curtida” a um documento. Ao fazer isso, todos os outros usuários da aplicação podem ver que aquele desenvolvedor “curtiu” aquele documento tanto na página do documento em si

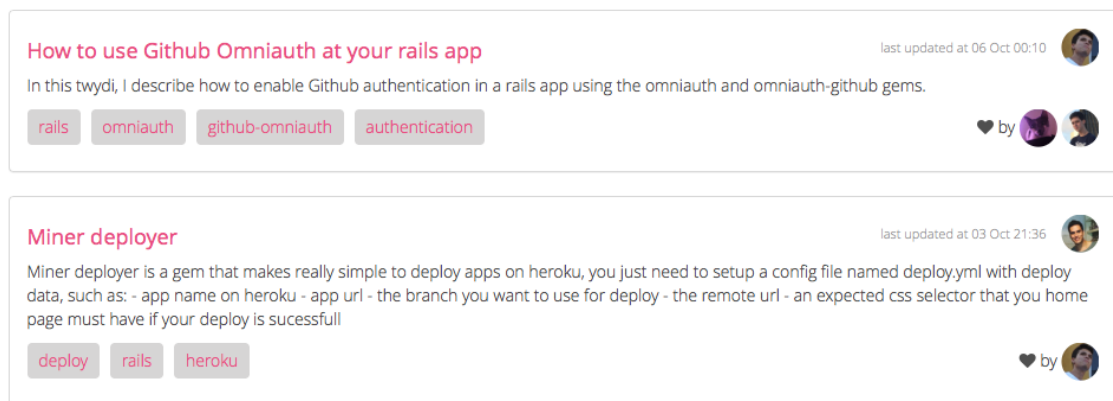
quando na página de listagem de documentos. Figuras 22 e 23 ilustram a funcionalidade.

Figura 22: Visualização dos “Curtir” em um documento



Fonte: O autor (2015)

Figura 23: Visualização dos “Curtir” na lista de documentos



Fonte: O autor (2015)

Esta funcionalidade permite ao usuário informar que o documento curtido foi útil para ele de alguma forma ou que pode ser útil para outros usuários, estimulando sua consulta por estes.

4.6 Limitações da Ferramenta

A aplicação ainda possui algumas limitações em suas funcionalidades. A respeito da importação, a ferramenta não suporta no momento a importação via *links* de código que se encontram em outros ramos (*branches*) de desenvolvimento do repositório.

Além disso, a ferramenta não trata mudanças que possam ter ocorrido na fonte do código importado. Ou seja, uma vez importado, mudanças ocorridas no código disponível no GitHub não se refletem na aplicação.

5 Análise e Discussão dos Resultados

Após o fim do período de uso pelos participantes da pesquisa, o autor iniciou a análise das respostas obtidas no questionário aplicado. Todas as perguntas eram opcionais pelo intuito de garantir que apenas respostas de qualidade fossem informadas.

5.1 Avaliação da ferramenta

Seguem as análises sobre cada uma das perguntas do questionário (com exceção da primeira, que era de identificação do participante). Cada participante foi anonimizado e toda referência a um participante será feita na forma de P# onde # se refere ao identificador do participante.

5.1.1 Sobre o que era seu twydi?

Os assuntos dos documentos foram variados. Alguns mais simples descreviam como clonar um repositório utilizando *git*¹ (criado por P4). Já P3, por exemplo, redigiu um documento detalhado de acesso à sua API. Outro documento que vale ser mencionado foi o criado por P8, sobre como utilizar uma biblioteca específica na linguagem *Ruby*.

Percebe-se que a ferramenta suporta a criação de documentos com diferentes tipos de complexidade e assunto.

5.1.2 No seu dia-a-dia de desenvolvimento, como você resolve os problemas pelos quais você passa ou como descobre como implementar algo novo pela primeira vez?

Todos os participantes com exceção de P4 afirmaram utilizar ferramentas de busca como Google, sites de perguntas e respostas como Stack Overflow e contato com outros

¹<https://git-scm.com>

desenvolvedores como meio de obter auxílio. P4 relatou que apenas utiliza ferramentas de pesquisa *online*.

Nota-se então nas equipes das quais os participantes fazem parte não existem práticas de gestão de conhecimento, tendo em vista de que não houve nenhuma resposta afirmando que havia algum tipo de consulta a algum repositório da equipe.

5.1.3 Com relação aos Twydis, qual a utilidade que você vê nos *links* relacionados?

Com exceção do participante 6 (que não respondeu a questão), todos afirmaram que utilizariam o campo de *links* relacionados como meio de permitir ao leitor aprofundamento sobre o assunto abordado no documento. De acordo com P7, por exemplo: *“Obter mais informações ou detalhes caso seja necessário.”*.

P2 afirmou: *“Embasamento pro que foi escrito, e ainda a possibilidade de aprofundar o assunto.”*. Assim, imagina que a função deste campo seja, além da já mencionada, oferecer maior credibilidade sobre o que está sendo escrito.

Assim, verifica-se que a ferramenta pode também servir de ponto de partida para o aprendizado de outros assuntos além do abordado nos seus documentos presentes, servindo de referencial para outras fontes de conhecimento para a equipe.

5.1.4 O que você mudaria na forma como os links relacionados estão implementados atualmente?

Sobre esta pergunta, P6 e P7 não responderam. Os demais participantes afirmaram que não sentem necessidade de alterações neste campo.

5.1.5 E qual a utilidade que você vê nas tags?

Todos os participantes com exceção de P6 (que não respondeu) afirmaram que a utilidade das *tags* seria de facilitar a busca por outros usuários da aplicação.

5.1.6 O que você mudaria na forma como as tags estão implementados atualmente?

Sobre esta pergunta, P6 e P7 não responderam. P1 sugeriu: “Faz igual do stackoverflow”. O mecanismo de inserção de *tags* em perguntas do site Stack Overflow exhibe sugestões de *tags* baseado no conteúdo escrito em suas perguntas, além de sugerir a grafia das *tags* conforme o usuário as digita (garantindo que são utilizadas sempre *tags* padronizadas pela aplicação). Sem dúvida, uma sugestão interessante que enriqueceria a experiência do usuário da ferramenta.

Os demais participantes afirmaram que não sentem necessidade de alterações neste campo.

5.1.7 Que importações de código você utilizou?

Para a criação do documento, P1 e P4 utilizaram a importação de linhas de arquivo. P5 e P6 utilizaram a importação de Múltiplas Linhas de Arquivo. P8 utilizou a importação de Arquivo Inteiro. Os demais participantes não responderam à questão. Destes, P2 afirmou como resposta da pergunta seguinte não ter utilizado o mecanismo de importação. Importações de *commit* e *pull-request* não foram utilizadas.

5.1.8 Que importação de código você achou mais útil? Por quê?

Todos os participantes informaram respostas bastante variadas. P1 afirmou não ver utilidade na ferramenta de importação. P2 afirmou: “*Não utilizei nenhuma, mas talvez a mais útil seria as relacionadas com linhas de arquivo, principalmente se puderem ser gists.*”. Gists² funcionam como repositórios de um arquivo só hospedados no GitHub. São comumente utilizados por desenvolvedores para armazenar *scripts* de código, arquivos de configuração ou notas pessoais. O suporte da aplicação desenvolvida para esse tipo de armazenamento de código pode ser uma funcionalidade útil para outros desenvolvedores. P5 afirmou que importação de um *commit* é a mais útil em sua opinião. P8 respondeu: “*Considero o modo de múltiplas linhas porque permite uma maior variedade de combinações para a resposta.*”.

P4 afirmou não ter conseguido utilizar a importação. P3 e P7 não responderam à pergunta.

²<https://gist.github.com>

5.1.9 Algum resultado da importação de código ficou ruim/ confuso? Por quê?

P4 e P6 afirmaram terem encontrado problemas com a importação. P4 informou: “*Não consegui utilizar importação, pois não entendi bem como funciona.*”, apesar das instruções presentes na tela de criação de documento. P6 afirmou teve problemas para importar um arquivo “*Leia-me*” em formato *markdown*, havendo a má detecção de sua linguagem. De fato a ferramenta não foi pensada com este tipo de arquivo em mente, o que justifica os problemas encontrados. Futuras versões da mesma podem tratar melhor deste caso.

Os demais participantes não responderam ou afirmaram não terem encontrado problemas com a importação.

5.1.10 No caso da importação de *commits* e *pull requests*, é mais importante:

Metade dos participantes afirmaram que é mais importante visualizar o código adicionado e removido pela ação. Esta forma é a já feita pela aplicação atualmente e, graças a estas respostas, deve se manter assim. Os demais participantes não responderam ou afirmaram não saber o que é mais importante.

5.1.11 Como você utilizaria a ferramenta no seu dia-a-dia?

Esta pergunta também obteve respostas variadas. P1, P2 e P8 afirmaram que utilizariam como forma de registrar como estes resolveram algum problema já enfrentado. P2 afirmou: “*Relembrar alguma implementação que eu fiz, ou alguma boa prática que achei interessante.*”. P3, P6 e P8 afirmaram que utilizariam para a divulgação de tutoriais ou soluções para suas equipes. P7 afirmou que utilizaria a aplicação para gerar arquivos de “*Leia-me*”, enquanto P5 afirmou que não utilizaria com frequência. P4 afirmou que utilizaria, segundo o próprio “*Em projetos de disciplinas.*”.

Assim, os casos de uso mais mencionados pelos participantes estão relacionados intimamente com políticas de gestão do conhecimento, área na qual se percebe deficiência de acordo com o notado através da pergunta 5.1.2.

5.1.12 O que pode ser melhorado na ferramenta para facilitar a escrita de documentos?

P3, P5, P6 e P7 afirmaram que funcionalidades relacionadas ao auxílio na formatação do documento como ferramentas visuais para formatação de texto ou pré-visualização do documento gerado. P1 e P8 afirmaram que não acham que haja nada a ser melhorado. P2 fez uma sugestão semelhante a de P1 na pergunta 5.1.6: *“Sugestões de tags baseados no conteúdo do twydi.”*.

Dessa forma, para futuras versões da aplicação, essas melhorias no formulário de implementação do documento devem ser implementadas.

5.1.13 Além dos campos já presentes em um Twydi, que outras informações você acrescentaria?

A maioria dos participantes afirmou não ter nenhum campo a mais em mente. P8 afirmou: *“Talvez a possibilidade de adicionar imagens”*.

5.1.14 Houve alguma dificuldade para o preenchimento deste questionário ou para o entendimento das atividades solicitadas?

Todos os participantes afirmaram não terem encontrado dificuldades para o preenchimento do questionário.

5.2 Conclusões Obtidas e Respostas das Perguntas de Pesquisa

Com base nas respostas obtidas, foi possível obter respostas para as perguntas de pesquisa inicialmente propostas.

5.2.1 1. A ferramenta proposta oferece suporte à maneira como conhecimento é trocado em equipes de software?

A ferramenta, de acordo com os participantes não substitui totalmente a forma como conhecimento é trocado em suas respectivas equipes, porém, atua como um complemento a esta forma. Foi afirmado que a ferramenta pode atuar como agregador e organizador

de informações já conhecidas pela equipe. Desta forma, é possível afirmar que **sim, a ferramenta oferece suporte à troca de conhecimento entre desenvolvedores.**

5.2.2 2. Que outras funcionalidades podem ser agregadas a ferramenta para melhorar este suporte, caso exista?

De acordo com as respostas obtidas e tendo em vista de que o suporte existe, as funcionalidades mais requisitadas são de **melhor suporte a pré-visualização e formatação e dos documentos**, mais especificamente como botões de adição de formatação ao texto inserido. Melhorias com relação ao mecanismo de *tags* também foram solicitados, porém, em escala reduzida.

5.2.3 3. Quais são os casos potenciais de uso da ferramenta segundo os desenvolvedores?

Dois casos de uso foram mencionados por múltiplos participantes, sendo o primeiro **divulgação de tutoriais ou soluções para suas equipes** e o segundo **registro pessoal sobre como o próprio desenvolvedor solucionou determinado problema**. Assim, a ferramenta possui utilidade tanto relacionada a difusão de informação para a equipe quanto relacionada ao registro e organização de conhecimento pessoal.

5.3 Limitações do Estudo

Este estudo possui possíveis limitações. Uma delas sendo o curto período de tempo no qual a ferramenta esteve disponível (três semanas). Com mais tempo, certamente haveria a oportunidade de se estabelecer uma cultura de uso da aplicação dentro dos contextos analisados e quesitos relacionados a recuperação de informação poderiam ser avaliados.

Além disso, houve uma participação limitada de voluntários e de apenas dois escopos distintos (alunos e membros da empresa júnior). Um escopo de participantes de empresas de desenvolvimento de software já estabelecidas certamente poderia enriquecer ainda mais os resultados da pesquisa e agregar mais melhorias para futuras versões da ferramenta.

Uma problemática do uso da ferramenta pode ser, com o passar do tempo, o imenso volume de artefatos criados pela equipe e a possível obsolescência destes. Futuras versões da ferramenta e estudos podem trabalhar em cima de mecanismos de “esquecimento” de artefatos.

6 Considerações Finais

Ao longo do desenvolvimento desse estudo, foi proposta a elaboração de uma ferramenta que oferecesse suporte a gestão de informação dentro do contexto de equipes de desenvolvimento de software. Visou-se observar se a ferramenta de fato oferecia tal suporte, o que poderia ser melhorado nela e potenciais casos de uso segundo os próprios desenvolvedores de software.

Rotatividade de pessoal e falta de registro de conhecimento obtido, aliado a alta necessidade de reuso na área da engenharia de software e a precariedade das mídias atuais deixam clara a demanda por uma ferramenta como a proposta.

Estudo de ferramentas semelhantes e inquéritos contextuais realizados por desenvolvedores ajudaram na elicitação e validação de requisitos para a ferramenta.

A ferramenta atua como um catálogo de implementações, descrições textuais passo-a-passo, código e referências externas na forma de *links*, permitindo a desenvolvedores criem e recuperem artefatos produzidos pela sua equipe.

Após sua implementação, um questionário de avaliação foi aplicado com desenvolvedores e foi constatado que a ferramenta elaborada oferece suporte à troca de conhecimento entre desenvolvedores. Porém um melhor suporte a pré-visualização e formatação e dos documentos podem aprimorar sua eficiência no dia-a-dia. Sobre sua utilidade foi constatado que a divulgação de tutoriais ou soluções para suas equipes e registro pessoal sobre como o próprio desenvolvedor solucionou determinado problema seriam casos de uso nos quais utilizariam a ferramenta em sua rotina.

Futuros estudos podem validar a aplicabilidade da ferramenta em outros contextos e por períodos maiores de tempo, com foco maior na avaliação da eficiência de recuperação de informações armazenadas na ferramenta quando comparada a outras mídias utilizadas pelos desenvolvedores de software da atualidade.

Referências

- BJØRNSON, F. O.; DINGSØYR, T. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology*, v. 50, n. 11, p. 1055 – 1068, 2008. ISSN 0950-5849. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0950584908000487>>.
- CLARK, H.; BRENNAN, S. Grounding in communication', 127-149 in resnick lb, levine jm and teasley sd. In: RESNICK, L. et al. (Ed.). *Perspectives on Socially Shared Cognition*. [S.l.]: American Psychological Association, 1991. p. 259–292.
- CUBRANIĆ, D. et al. Learning from project history: A case study for software development. In: *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*. New York, NY, USA: ACM, 2004. (CSCW '04), p. 82–91. ISBN 1-58113-810-5.
- DRUCKER, P. F. D. P. F. *Post-Capitalism Society*. 1st. ed. UK, Oxford: Butherworth-Heinemann, 1993.
- FIGUEIRA FILHO, F. et al. A study on the geographical distribution of brazil's prestigious software developers. *Journal of Internet Services and Applications*, Springer London, v. 6, n. 1, p. 1–12, 2015.
- FREEMAN, P. Reusable software engineering: Concepts and research directions. *ITT Proceedings of the Workshop on Reusability in Programming*, v. 129, p. 137, 1983.
- GOUSIOS, G.; SPINELLIS, D. Ghtorrent: Github's data from a firehose. In: *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. Piscataway, NJ, USA: IEEE Press, 2012. (MSR '12), p. 12–21. ISBN 978-1-4673-1761-0. Disponível em: <<http://dl.acm.org/citation.cfm?id=2664446.2664449>>.
- HALL, T. et al. The impact of staff turnover on software projects: The importance of understanding what makes software practitioners tick. In: *Proceedings of the 2008 ACM SIGMIS CPR Conference on Computer Personnel Doctoral Consortium and Research*. New York, NY, USA: ACM, 2008. (SIGMIS CPR '08), p. 30–39. ISBN 978-1-60558-069-2.
- KAVITHA, R.; AHMED, M. I. A knowledge management framework for agile software development teams. In: IEEE. *Process Automation, Control and Computing (PACC), 2011 International Conference on*. [S.l.], 2011. p. 1–5.
- KRUEGER, C. W. Software reuse. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 24, n. 2, p. 131–183, jun. 1992. ISSN 0360-0300. Disponível em: <<http://doi.acm.org/10.1145/130844.130856>>.

LEVY, M.; HAZZAN, O. Knowledge management in practice: The case of agile software development. In: *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2009. (CHASE '09), p. 60–65. ISBN 978-1-4244-3712-2.

OLSON, G. M.; OLSON, J. S. Distance matters. *Hum.-Comput. Interact.*, v. 15, n. 2, p. 139–178, 2000.

RABELO, J. et al. Knowledge management and organizational culture in a software organization: a case study. In: IEEE PRESS. *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*. [S.l.], 2015. p. 89–92.

SANGMOK, H. *Improved source code editing for effective ad-hoc code reuse*. Tese (Doutorado) — Massachusetts Institute of Technology, 2011.

SENGE, P. M. *The fifth discipline fieldbook: Strategies and tools for building a learning organization*. [S.l.]: Crown Business, 2014.

SINGER, L.; FILHO, F. F.; STOREY, M.-A. Software engineering at the speed of light: How developers stay current using twitter. In: ACM. *Proceedings of the 36th International Conference on Software Engineering*. [S.l.], 2014. p. 211–221.

STEINMACHER, I. et al. Social barriers faced by newcomers placing their first contribution in open source software projects. In: ACM. *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*. [S.l.], 2015. p. 1379–1392.

STOREY, M.-A. et al. The (r) evolution of social media in software engineering. In: *Proceedings of the on Future of Software Engineering*. New York, NY, USA: ACM, 2014. (FOSE 2014), p. 100–116. ISBN 978-1-4503-2865-4.

TREUDE, C.; BARZILAY, O.; STOREY, M.-A. How do programmers ask and answer questions on the web?: Nier track. In: ACM. *Proceeding of the 33rd international conference on Software engineering*. [S.l.], 2011. p. 804–807.

TREUDE, C. et al. Programming in a socially networked world: the evolution of the social programmer. *The Future of Collaborative Software Development*, p. 1–3, 2012.

WEINBERG, G. M. *The Psychology of Computer Programming (Silver Anniversary Ed.)*. New York, NY, USA: Dorset House Publishing Co., Inc., 1998. ISBN 0-932633-42-0.

WIIG, K. M.; JOOSTE, A. Exploiting knowledge for productivity gains. In: _____. [S.l.]: Springer, 2004. cap. Handbook on Knowledge Management, p. 289–308.

APÊNDICE A – Questionário

1. Nome ou maneira como deseja ser identificado
2. Sobre o que era seu twydi?
3. No seu dia-a-dia de desenvolvimento, como você resolve os problemas pelos quais você passa ou como descobre como implementar algo novo pela primeira vez?
4. Com relação aos Twydis, qual a utilidade que você vê nos *links relacionados*?
5. O que você mudaria na forma como os links relacionados estão implementados atualmente?
6. E qual a utilidade que você vê nas tags?
7. O que você mudaria na forma como as tags estão implementados atualmente?
8. Que importações de código você utilizou? (pode ser marcada mais de uma)
 - (a) Linha de arquivo
 - (b) Múltiplas linhas de arquivo
 - (c) Arquivo Inteiro
 - (d) *Commit*
 - (e) *Pull request*
9. Que importação de código você achou mais útil? Por quê?
10. Algum resultado da importação de código ficou ruim/ confuso? Por quê?
11. No caso da importação de *commits* e *pull requests*, é mais importante:
 - (a) Visualizar somente o código adicionado
 - (b) Visualizar o código adicionado e removido



(c) Não sei

12. Como você utilizaria a ferramenta no seu dia-a-dia?
13. O que pode ser melhorado na ferramenta para facilitar a escrita de documentos?
14. Além dos campos já presentes em um Twydi, que outras informações você acrescentaria?
15. Houve alguma dificuldade para o preenchimento deste questionário ou para o entendimento das atividades solicitadas?



ANEXO A – Outros exemplos de Dúvidas das Equipes

Figura 24: Dúvida do tipo 1

November 3rd



  2:51 PM
@channel: alguém tem alguma sugestão de como implementar isso:



- 1) tenho 3 ou mais entidades diferentes
- 2) quero uma barra de busca só que busque dados das 3 entidades
- 3) a página resultante deve ter resultados das 3 entidades (edited)

  2:51 PM
Como assim?



quero uma barra de busca só que busque dados das 3 entidades



Exemplo

  2:52 PM
entidade 1 tem nome e cpf
entidade 2 tem cep
se eu botar um cpf, vai retornar coisas da entidade 1
que tenham aquele cpf
vulgo uma "barra de busca universal"
sacou?

  2:53 PM
Elastic

Deve ser mais simples

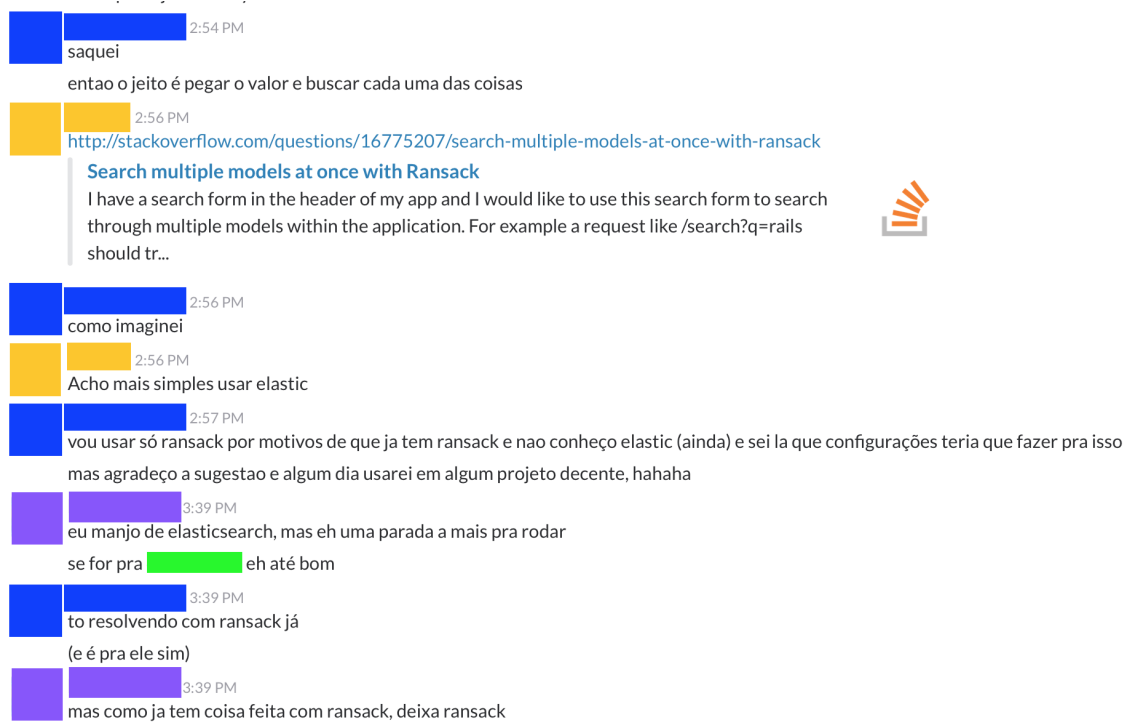
  2:53 PM
ransack puro nao rola nao?

  2:53 PM
Rolar rola

Mas vai requerer que você faça uma entidade agregadora
Nem que seja um array com todo mundo

Fonte: Adaptado de Slack (2015)

Figura 25: Dúvida do tipo 1 (continuação)



Fonte: Adaptado de Slack (2015)

Figura 26: Dúvida do tipo 2

August 23rd

  12:44 AM
@channel: alguém tem alguma recomendação de como implementar um relacionamento com "tags"?

  12:45 AM
 Se quer algo simples

  12:45 AM
 vai ser um simples has_many :tags mesmo?

  12:45 AM
 Usa array

  12:45 AM
 vou precisar recuperar pelas tags dps

  12:45 AM
 Beleza

<http://blog.arkency.com/2014/10/how-to-start-using-arrays-in-rails-with-postgresql/>

Arkency Blog
How to start using Arrays in Rails with PostgreSQL
 So far we covered a lot of PostgreSQL goodness. We've already talked about using uuid or storing hashes in our Rails applications with PostgreSQL database. Now is the time to do something in the middle of these topics (more complex than uuid, but easier than hstore) - we want to store list of simple values under one attribute. How can we do that? You may think "arrays" right now and you are correct. Let's see how we can achieve that.

<http://www.postgresql.org/docs/9.4/static/functions-array.html>

Acho mais trabalhoso que ter uma tabela para tags
 Mas sei lá
 Se quiser mais flexibilidade

  12:49 AM
 vou fazer a tabela, mesmo

  12:49 AM
<https://github.com/mbleigh/acts-as-taggable-on>

GitHub
mbleigh/acts-as-taggable-on
 acts-as-taggable-on - A tagging plugin for Rails applications that allows for custom tagging along dynamic contexts.

Rails é cheio dessas gems para tags
 Sugiro ver como eles fazem e copiar
 Vale mais a pena que instalar a gem

Fonte: Adaptado de Slack (2015)

Figura 27: Dúvida do tipo 5

August 14th

12:56 AM

@channel: no app.js:

```
var doc_implementation = document.getElementById('doc_implementation');
doc_implementation.onpaste = function(e) {
  window.alert('colou');
};
```

alguém sabe me dizer pq ele dizer que doc_implementation é null?

sendo que se eu vou no console

e dou

```
document.getElementById('doc_implementation');
```

ele acha a parada que eu quero

ah, descobri

ele tava carregando antes do elemento existir

resolvi

1:00 AM

added a JavaScript/JSON snippet ▾

```
1 document.addEventListener("DOMContentLoaded", function () {
2   var doc_implementation =
document.getElementById('doc_implementation');
3
4   doc_implementation.onpaste = function(e) {
5
6     window.alert('colou');
7   };
8 });
```

9:03 PM

deixa tudo dentro de um document.ready

9:09 PM

mesma coisa que isso, nao?

só que sem jquery

Fonte: Adaptado de Slack (2015)