

COPYCATCH – SISTEMA DE DETECÇÃO DE PLÁGIO

RESUMO: O objetivo deste projeto é apresentar uma ferramenta desenvolvida para detectar plágio em trabalhos acadêmicos automaticamente, com uma lógica similar ao raciocínio humano. Com o aumento de pessoas ingressando no mundo acadêmico, a detecção de plágio vem se tornando algo muito trabalhoso de se realizar e até mesmo ignorada em alguns casos. Diante disso, foi desenvolvido um sistema que possui verificação online, entre arquivos físicos e por SaaS. O projeto utiliza a lógica fuzzy para efetuar suas comparações, de forma que quando se comparam dois textos, não tem como resultado um simples “sim” ou “não”, ela pode responder “talvez este texto seja igual ao primeiro”, ou seja, o texto não precisa ser exatamente igual ao original para ser considerado plágio, possuindo assim uma taxa considerável de detecção. O desenvolvimento do projeto possibilitou uma detecção de plágio com maior eficiência, custo benefício e facilidade de integração com sistemas legados, como, por exemplo, portais de aluno.

1 INTRODUÇÃO

Um trabalho acadêmico tem como algumas de suas finalidades demonstrar, difundir, recuperar ou contestar o conhecimento produzido, acumulado ou transmitido. A demonstração do conhecimento é uma necessidade na comunidade acadêmica, em que esse conhecimento é um critério de mérito e acesso.

Cada vez mais trabalhos acadêmicos possuem algum plágio, principalmente devido à facilidade de se encontrar algo pronto online ou até mesmo pela possibilidade de transferência de arquivos pelas redes sociais. Devido a este fato percebe-se um aumento na procura de soluções de detecção de plágio entre profissionais do meio acadêmico.

Austin e Brown (1999) chamam a atenção para uma série de sítios na Internet que vendem trabalhos acadêmicos prontos, ou sob encomenda, com nomes sugestivos como CheatHouse. No Brasil, é famoso o site Zé Moleza (<http://www.zemoleza.com.br>) entre alunos de graduação.

Com o aumento de pessoas ingressando no mundo acadêmico, a detecção de plágio vem se tornando algo muito trabalhoso de se realizar e até mesmo ignorada em alguns casos. Sendo que, ao efetuar uso do plágio, os alunos, além de prejudicarem sua própria aprendizagem, estão cometendo um crime.

Segundo o código penal, se a violação consistir em reprodução total ou parcial, com intuito de lucro direto ou indireto, por qualquer meio ou processo, de obra intelectual, interpretação, execução ou fonograma, sem autorização expressa do autor, do artista intérprete ou executante, do produtor, conforme o caso, ou de quem os represente: Pena reclusão, de 2 (dois) a 4 (quatro) anos, e multa.(CÓDIGO PENAL, ARTIGO 184, 1º)

Neste contexto, este projeto descreve o desenvolvimento de um sistema para detecção de plágio por meio de serviços de verificação entre arquivos e utilizando os principais sites de pesquisas. Esses serviços podem ser utilizados por qualquer instituição que aderir ao uso do sistema. As instituições pagam somente pelos serviços que utilizarem, de forma que qualquer instituição possa aderir ao sistema, independentemente de seu porte.

2 REFERENCIAL TEÓRICO

Nesta seção os conceitos referentes ao processo de desenvolvimento do trabalho serão apresentados: manifesto ágil, scrum, software as a service, integração contínua, bancos NoSQL e lógica fuzzy.

2.1 MANIFESTO ÁGIL

Segundo Rodrigues o desenvolvimento ágil é qualquer processo de desenvolvimento criado com base nos conceitos do Manifesto Ágil. Tal manifesto, assinado em 2001, foi elaborado por profissionais experientes e veteranos da indústria de software. Embora cada um tivesse a sua forma de pensar e agir, todos estavam de acordo que os projetos de sucesso tinham certos princípios em comum. Esses princípios servem como diretrizes para equipes que procuram uma forma ágil de administrar seus processos de desenvolvimento.

Segundo Beck, Cunningham e Hunt o Manifesto Ágil apresenta doze princípios para serem seguidos durante o desenvolvimento:

- Satisfazer o cliente entregando o produto no prazo e com qualidade.
- Estar aberto às mudanças constantes de requisitos, adequando o processo as novas necessidades.
- Entregas constantes em um período curto.
- Todas as partes interessadas no processo devem trabalhar juntas durante todo o processo.
- Proporcionar um ambiente adequado para o trabalho e suporte aos desenvolvedores confiando em suas capacidades de trabalho.
- Comunicação eficaz entre os membros da equipe.
- Software sempre funcional.

- Manter sempre o mesmo ritmo de desenvolvimento.
- Atenção contínua a excelência técnica e bom design visando aumentar a agilidade.
- Simplicidade.
- Equipes auto-organizacionais promovem arquitetura, requisitos e designs de melhor qualidade.
- Discussões em intervalos regulares para aperfeiçoar o processo de desenvolvimento.

Segundo Rocha (2015) atualmente agilidade tornou-se a palavra da moda quando se descreve um moderno processo de software. Todo mundo é ágil. Uma equipe ágil é aquela rápida e capaz de responder apropriadamente a mudanças. Mudanças têm muito a ver com desenvolvimento de software. Mudanças no software que está sendo criado, mudanças nos membros da equipe, mudanças devido a novas tecnologias, mudanças de todos os tipos que poderão ter um impacto no produto que está em construção ou no projeto que cria o produto. Suporte para mudanças deve ser incorporado em tudo o que fazemos em software, algo que abraçamos porque é o coração e a alma do software. Uma equipe ágil reconhece que o software é desenvolvido por indivíduos trabalhando em equipes e que as habilidades dessas pessoas, suas capacidades em colaborar, estão no cerne do sucesso do projeto.

2.2 SCRUM

Segundo Sabbagh (2013), Scrum é um *framework* ágil, simples e leve, utilizado para a gestão do desenvolvimento de produtos complexos imersos em ambientes complexos. O Scrum é embasado no empirismo e utiliza uma abordagem iterativa e incremental para entregar valor com frequência e, assim, reduzir os riscos do projeto.

De acordo com Alliance, Scrum é um processo de equipe. O Time Scrum inclui três papéis: o *Product Owner*, o *ScrumMaster* e os membros do Time de Desenvolvimento. O *Product Owner* tem a responsabilidade de decidir que trabalho será realizado. O *ScrumMaster* atua como um líder/servidor, auxiliando o time e a organização a fazer o melhor uso do Scrum. O Time de Desenvolvimento constrói o produto incrementalmente, em uma série de períodos curtos de tempo, chamados de *Sprints*. Uma *Sprint* é um período de tempo fixo, de uma a quatro semanas, com uma preferência por intervalos mais curtos. Em cada *Sprint*, o Time

Scrum irá construir e entregar um Incremento do Produto. Cada incremento é um subconjunto operacional, com melhorias evidentes e reconhecíveis do produto, atendendo a critérios de aceitação compreendidos e é construído com um nível de qualidade chamado Definição de Pronto.

O Scrum também possui um importante artefato, o *product backlog*. O *Product Backlog* é uma lista ordenada de ideias para o produto. É a única fonte da qual todos os requisitos fluem. Isso significa que todo o trabalho que o Time de Desenvolvimento faz vem do *Product Backlog*. Cada ideia de funcionalidade, melhoria, correção de problemas, documentação necessária - qualquer trabalho que eles façam - é derivado de um item do *Product Backlog*. Cada item no *Product Backlog* inclui uma descrição e uma estimativa.

2.3 SOFTWARE AS A SERVICE

Segundo Chong e Carraro (2015), *SaaS* (software as a service) pode ser definido como um software implementado como um serviço hospedado e acessado pela Internet.

Por exemplo, considerando um serviço de email baseado na Web como o Microsoft® Hotmail®. Ele atende todos os critérios básicos: um fornecedor hospeda todos os dados e a lógica do programa e fornece acesso a esses dados ao usuário final através da Internet pública por meio de uma interface do usuário baseada na Web.

2.4 INTEGRAÇÃO CONTÍNUA

Segundo Fowler(2006), a integração contínua é uma prática de desenvolvimento de software, onde os membros de uma equipe integram o seu trabalho com frequência, levando a múltiplas integrações por dia. Cada integração é verificada por meio de uma compilação automatizada (incluindo testes) para detectar erros de integração tão rapidamente quanto possível. Esta abordagem leva a uma redução significativamente dos problemas de integração e permite que a equipe possa desenvolver softwares coesos rapidamente.

Segundo Humble e Farley(2010) em muitas organizações, o tempo de ciclo é medido em semanas ou meses, e o processo de implantação certamente não é repetível ou confiável. É manual e muitas vezes requer uma equipe de pessoas para implantar o software, mesmo em um ambiente de teste ou produção. Automação é a chave. Ela permite que todas as tarefas comuns envolvidas na criação e implantação de software que seriam realizadas

pelos desenvolvedores, testadores e pessoal de operações, sejam executadas pressionando um botão.

2.5 BANCOS NOSQL

De acordo com o artigo da empresa MongoDB (2015), NoSQL abrange uma grande variedade de diferentes tecnologias de banco de dados, que foram desenvolvidas em resposta a diversos fatores como:

- Aumento do volume de dados armazenados sobre usuários, objetos e produtos.
- Desempenho.
- Processamento.
- Necessidades.

Existem diferentes tipos de bancos NoSQL, estes são alguns exemplos (MONGODB, 2015):

- **Documental:** emparelhar cada tecla com uma complexa estrutura de dados conhecida como um documento. Os documentos podem conter muitos diferentes pares de valores-chave, ou pares de chave-matriz, ou mesmo documentos aninhados. Exemplo MongoDB.
- **Gráficos:** são usados para armazenar informações sobre redes, como conexões sociais. Exemplos: Neo4J e HyperGraphDB.
- **Chave-Valor:** são as mais simples bancos de dados NoSQL. Cada item único no banco de dados é armazenado como um nome de atributo (ou "key"), juntamente com o seu valor. Exemplo: Redis.
- **Famílias de colunas:** como Cassandra e HBase são otimizados para consultas sobre grandes conjuntos de dados e armazena colunas de dados em conjunto, em vez de linhas.

Segundo Fowler e Sadalage (2013) os bancos de dados relacionais estão sendo vistos como uma opção para o armazenamento de dados. Esse ponto de vista é, muitas vezes, chamado de persistência poliglota – utilizar diferentes armazenamentos de dados em diferentes circunstâncias. Em vez de escolher o banco de dados relacional mais utilizado por todos, precisamos entender a natureza dos dados que estamos armazenando e como

queremos manipulá-los. O resultado é que a maioria das organizações terá uma mistura de tecnologias de armazenamento de dados para diferentes circunstâncias.

2.6 LÓGICA FUZZY

Segundo Castillo, Melin(2008) a lógica fuzzy é a forma de lógica multivalorada na qual os valores lógicos das variáveis podem ser qualquer número real entre 0 (FALSO) e 1 (VERDADEIRO) ao contrário da lógica convencional que pode ser apenas 0 ou 1.

A lógica fuzzy foi criada para lidar com o conceito de verdade parcial, onde o valor verdade(1) pode compreender entre completamente verdade e completamente falso.

3 METODOLOGIA

A metodologia utilizada no desenvolvimento do projeto foi Scrum, que prega a metodologia ágil durante seu processo de desenvolvimento de software, sendo iterativo e incremental, extremamente focado nas pessoas e sem burocracia.

Para a parte de desenvolvimento foram utilizadas as seguintes ferramentas:

- Docker 1.5: utilizado para criação do ambiente de produção e desenvolvimento.
- Git 2.1: utilizado para gerenciamento de código fonte.
- MongoDB 2.7: utilizado para armazenar os dados das instituições que aderirem ao sistema.
- NodeJS 10.35: utilizada para desenvolvimento do *módulo web* do sistema.
- Python 2.7: linguagem utilizada para desenvolvimento do algoritmo de comparação e *api's* do sistema.
- SublimeText 3: utilizado como *IDE* para desenvolvimento do *módulo web* do sistema.
- PyCharm 4.5: utilizado como *IDE* para desenvolvimento do algoritmo de comparação e *api's* do sistema.
- CodeShip: utilizado para automação do processo de *deploy* do sistema.
- OpenShift: utilizado como *hospedagem* para o sistema.

- FuzzyWuzzy: módulo que implementa lógica fuzzy, utilizado nas comparações entre trechos de texto do sistema

Por ser um processo em ciclos, o Scrum passa por diversas etapas até a conclusão do produto. O ponto de partida é a construção do *product backlog*, em que o *product owner* constrói as *user stories*. Nesse projeto foi definida como ferramenta de criação das histórias uma planilha do Google Docs, que possibilita aos participantes do processo o acesso do documento de forma rápida e em qualquer lugar. A Tabela 1 mostra as *user stories* referentes ao projeto. Cada *user story* representa o que foi desenvolvido ao decorrer do projeto.

ID	Story	Pontos	Sprint
1	Como desenvolvedor eu quero usufruir do mesmo ambiente de trabalho na produção e no <i>staging</i> para que não ocorra problemas de compatibilidade.	5	0
2	Como desenvolvedor eu quero que o processo de <i>deploy</i> do sistema seja automatizado para que todas as tarefas sejam executadas automaticamente com verificação de erros	5	0
3	Como usuário eu quero poder efetuar a comparação entre diversos arquivos para verificar se houve plágio	8	1
4	Como usuário eu quero poder efetuar a comparação entre diversos arquivos através do módulo web para que eu possa efetuar uma verificação online	5	2
5	Como usuário eu quero acessar o sistema por meio de login para que meus dados estejam seguros	5	2
6	Como admin do sistema eu quero poder efetuar o cadastro de instituições para que somente instituições autorizadas possam utilizar o sistema	5	3
7	Como usuário eu quero poder efetuar a comparação de um trecho de texto online, utilizando os principais sites de pesquisa para verificar se houve plágio	8	4
8	Como usuário eu quero poder efetuar a comparação entre diversos arquivos como também utilizando os principais sites de pesquisas para verificar se houve plágio	13	5

9	Como usuário eu quero poder utilizar a verificação entre arquivos através de uma API para que eu possa integrar esta função ao meu sistema	5	6
10	Como usuário eu quero poder utilizar a verificação online através de uma API para que eu possa integrar esta função ao meu sistema	5	6
11	Como usuário eu quero poder utilizar a verificação múltipla através de uma API para que eu possa integrar esta função ao meu sistema	5	7
12	Como admin eu quero poder ter controle sobre o uso dos serviços por usuário para poder calcular o preço a ser cobrado.	8	7

Fonte: Dados do trabalho

Tabela 1 – User Stories

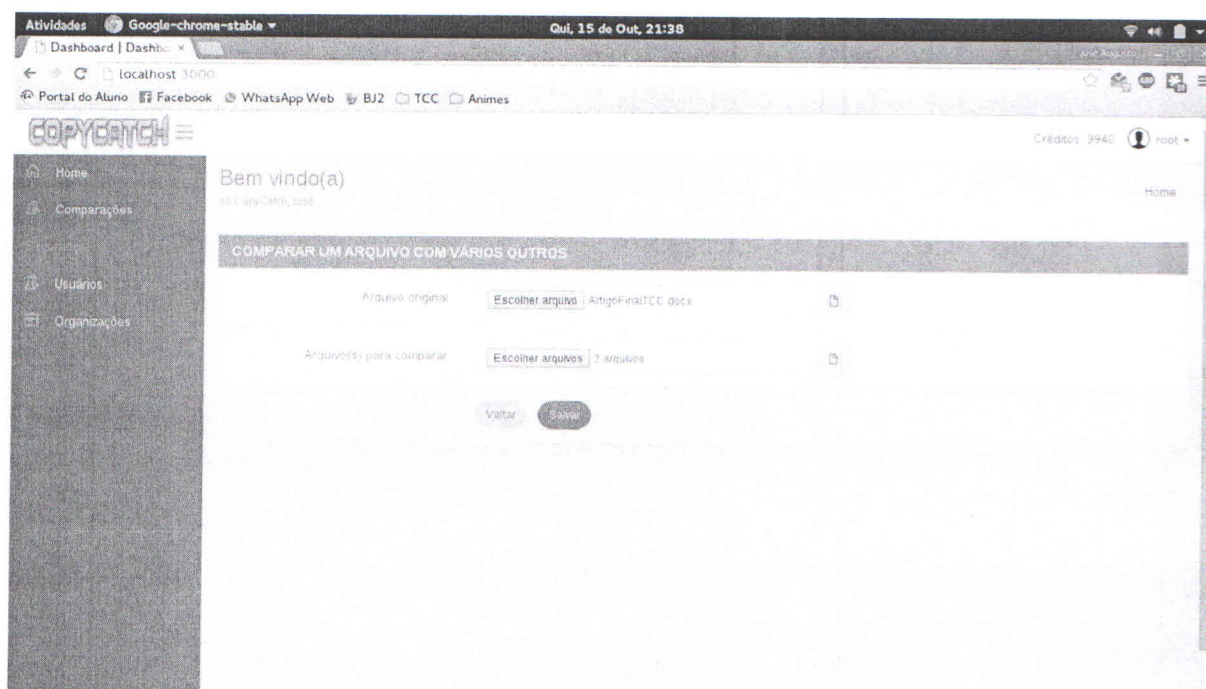
Neste projeto foram utilizados os números da sequência de *Fibonacci* para estimativa, facilitando a escolha das *stories* que entrarão em cada *sprint*. Como exemplo, foi dado a *user story* #8 uma pontuação superior a #1 devido a sua complexidade, seguindo assim o padrão para cada pontuação das *user stories*. Exemplo sequência *Fibonacci*: 0, 1, 1, 2, 3, 5, 8, 13, 21.

No final de cada Sprint foi realizada uma nova reunião de revisão, onde houve a inspeção dos itens desenvolvidos e possíveis mudanças no projeto. Esse processo foi repetido até que o produto estivesse concluído, devido à natureza iterativa e incremental do Scrum.

4 DESENVOLVIMENTO

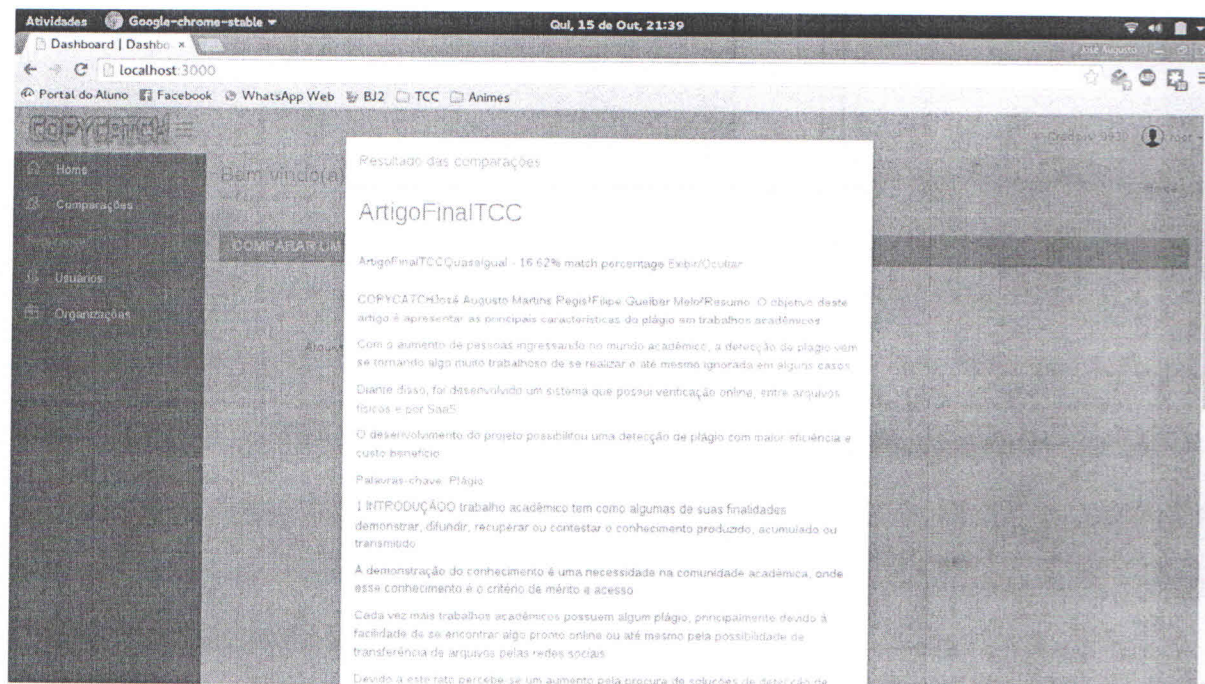
As principais funcionalidades do sistema são:

1 - Comparação de um arquivo com vários outros representada na figura 1, função na qual é informado um arquivo para se comparar com vários outros arquivos também informados, tendo como resultado a porcentagem de plágio para cada arquivo, como também os trechos considerados como plágio representado na figura 2.



Fonte: Dados do trabalho

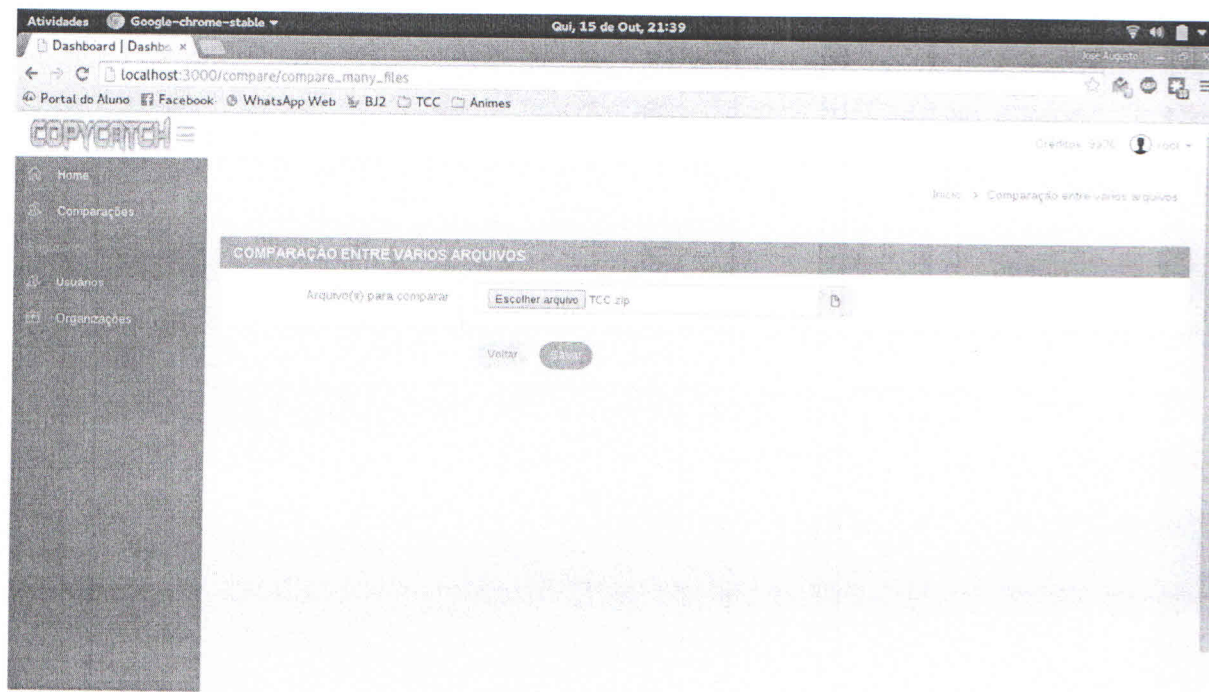
Figura 1 - Tela da funcionalidade de comparar um arquivo com vários outros



Fonte: Dados do trabalho

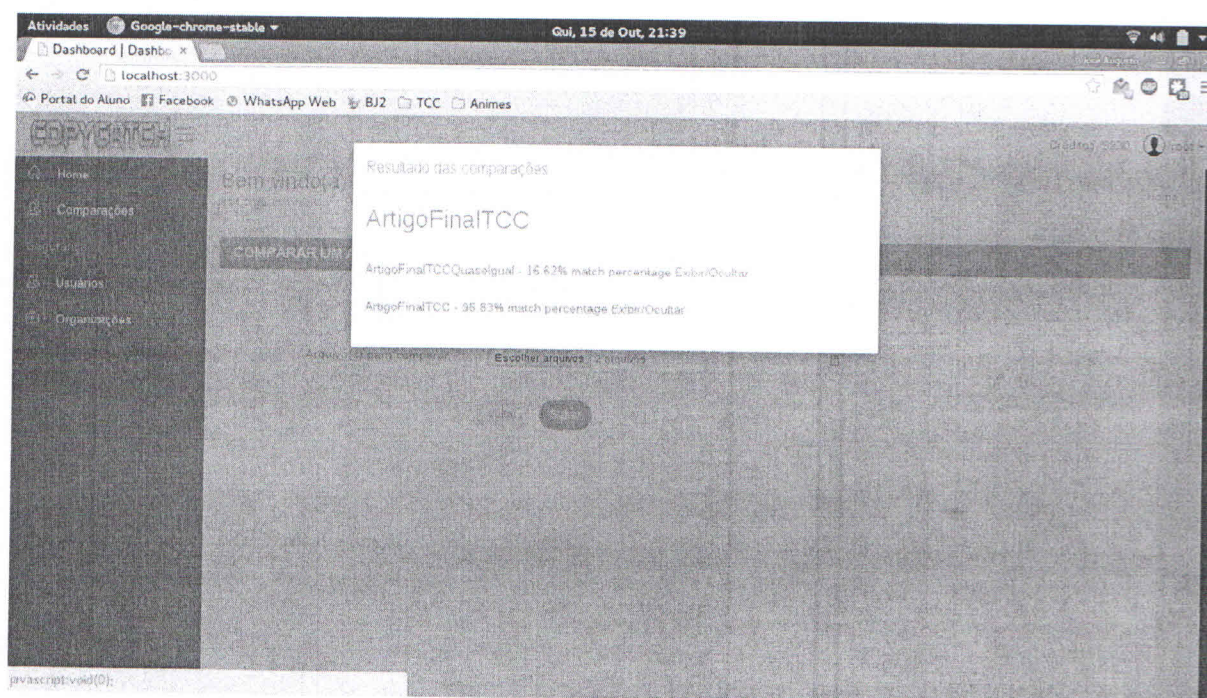
Figura 2 – Tela com trechos considerados como plágio

2 - Comparação entre vários arquivos representada na figura 3, função na qual é informado um arquivo compactado com vários arquivos para comparar entre si, resultado representado na figura 4.



Fonte: Dados do trabalho

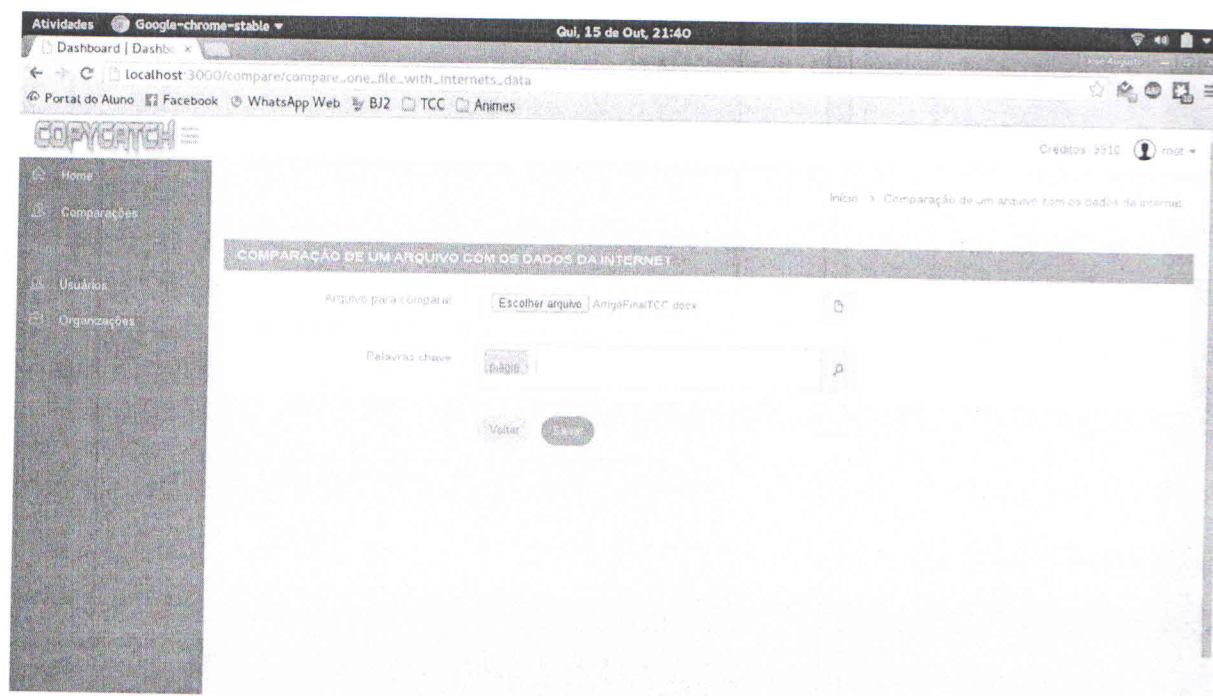
Figura 3 – Tela da funcionalidade de comparação entre vários arquivos



Fonte : Dados do trabalho

Figura 4 – Tela com resultados da funcionalidade de comparação entre vários arquivos

3 - Comparação de um arquivo com diversos sites de pesquisas representada na figura 5, função na qual é informado um arquivo para se comparar e as palavras chaves que definem o assunto do trabalho, as quais são utilizadas para encontrar sites que descrevem algo sobre o assunto, tendo além dos resultados anteriores, o link de onde foi retirado o texto considerado como plágio.

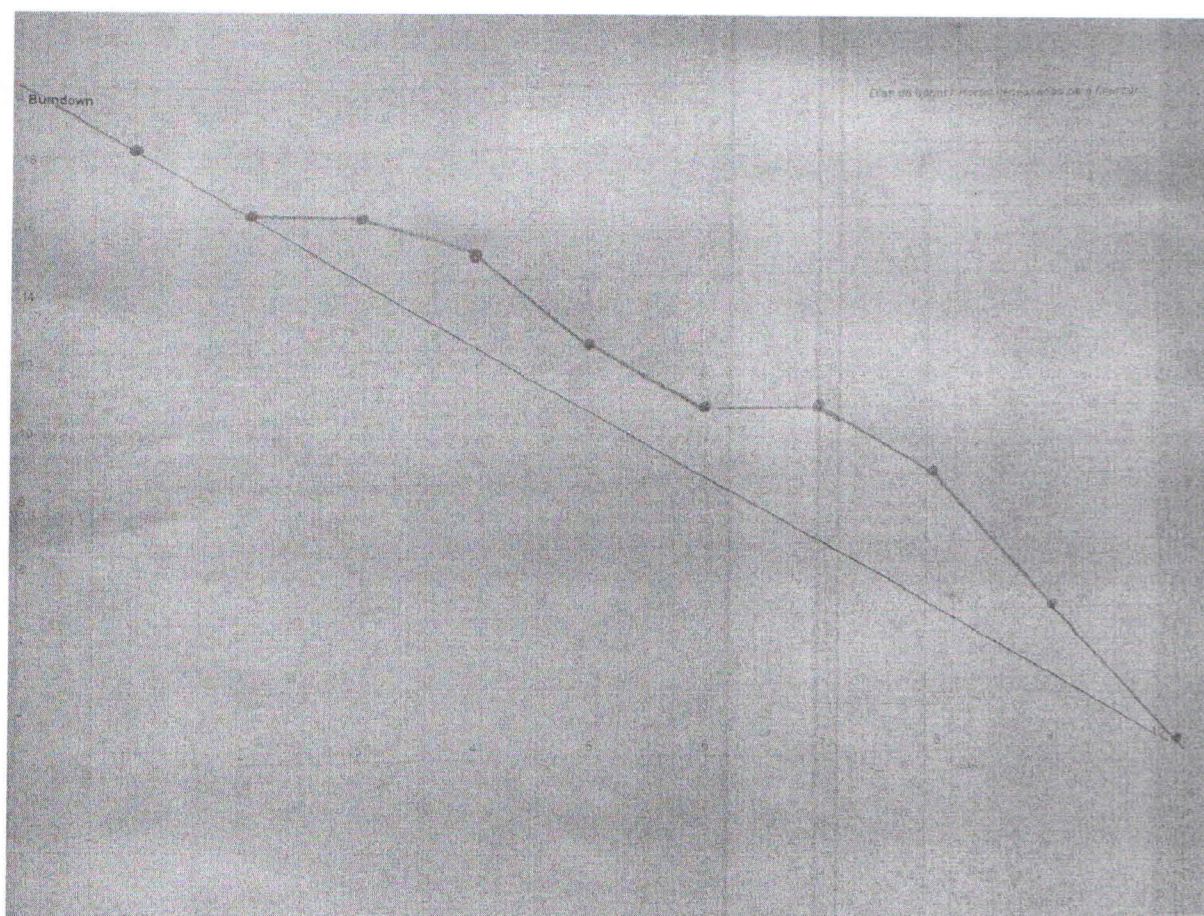


Fonte: Dados do trabalho

Figura 5 – Tela da funcionalidade de comparar um arquivo com dados retirados da internet

4 - Controle de usuários, função na qual é possível gerenciar os usuários do sistema, definindo ainda um papel com permissões para cada usuário.

5 - Controle de organizações, função na qual é possível gerenciar as organizações (clientes) que possuem acesso ao sistema, visualizando crédito restante e podendo ainda desativar o acesso para todos os usuários relacionados com aquela organização.



Fonte : Dados do trabalho
 Figura 9 – Burndown da sprint 7

5 RESULTADO

Foi desenvolvido um método de detecção de plágio, que efetua a comparação entre trabalhos, utilizando também os principais sites de pesquisas em conjunto com a lógica fuzzy, para determinar a porcentagem de plágio.

Segundo Castillo, Melin(2008) a lógica fuzzy coloca-se como o principal instrumento para uma representação mais adequada do conhecimento (e do próprio raciocínio), isso se devendo à sua capacidade de lidar com incertezas, raciocínio aproximado, termos vagos e ambíguos, o que as pessoas pensam, isso tudo indo além do escopo das lógicas clássicas. Dessa forma, a lógica fuzzy permite aos sistemas computacionais inteligentes “raciocinar” considerando aspectos inerentes à incerteza e aos processos realísticos e torná-los mais “humanos”.

Decidiu-se pelo uso da lógica fuzzy quando foi descoberto que somente a lógica normal não seria o suficiente para detectar o plágio, pois ela retorna uma simples resposta, “sim” ou “não”, ou seja, este texto é exatamente igual a este outro texto, o que faria com que

o sistema não tivesse uma precisão desejável ao efetuar as comparações. Utilizando a lógica fuzzy é possível obter outras respostas, como por exemplo, "talvez", "quase", deixando a comparação mais precisa. O texto comparado não precisa ser exatamente o mesmo para ser considerado plágio, o que torna útil essa forma de comparação.

Foi utilizado a ferramenta FuzzyWuzzy, que é um módulo que encapsula várias funções de comparações utilizando a lógica fuzzy. Utilizando esta ferramenta foi possível efetuar a comparação automática entre frases com um raciocínio semelhante ao humano. Por exemplo, comparando a frase "João atrasou para a aula porque perdeu o ônibus" com "Maria perdeu o ônibus e atrasou para a aula" é encontrado um resultado de 81% de compatibilidade. Assim, mesmo se o aluno alterar um pouco o texto plagiado, ele ainda será identificado pelo sistema.

Todas as funcionalidades do sistema foram disponibilizadas em webservices, facilitando a integração com outros sistemas.

6 CONCLUSÃO

O CopyCatch tem como propósito melhorar a verificação de plágio nos trabalhos acadêmicos, possuindo ainda custo sob demanda, sendo que cada serviço que o sistema disponibiliza possui um preço, assim o cliente paga por uso dos serviços, reduzindo assim o custo, pois somente será cobrado o que foi realmente utilizado.

O sistema permite realizar três tipos de comparações como, um trabalho com diversos outros, entre vários arquivos e por fim a comparação utilizando os principais sites de pesquisa. Cada um desses serviços está disponível a partir de uma chamada *rest*, facilitando assim a integração com diversos serviços.

O sistema aceita 4 formatos de arquivos: txt, pdf, docx, zip(contendo os formatos anteriores).

O Sistema faz uso da programação paralela, disponibilizando assim os resultados das comparações em um curto espaço de tempo.

O acesso ao sistema é gerenciado por organizações, ou seja, um cliente pode ser uma organização e teria assim inúmeros usuários para esta organização, o que gera um controle de acesso ao sistema e também facilita o próprio uso.

Além de retornar quais os trechos considerados como plágio, o sistema também disponibiliza uma porcentagem de plágio, como por exemplo uma atividade definida com limite de plágio de 80% pode automaticamente rejeitar trabalhos entregues, cuja porcentagem de plágio seja superior à definida, ou seja, utilizando-se as chamadas *rest* do sistema é possível automatizar decisões que normalmente tomariam muito tempo.

Assim, o sistema CopyCatch é um serviço oferecido sob demanda que agiliza o processo de avaliação de trabalhos acadêmicos possuindo disponibilidade para várias empresas, custo flexível e variável e facilidade de integração com sistemas legados, como por exemplo, portais de alunos.

Como trabalhos futuros pretende-se desenvolver um aplicativo que processe trechos de textos visualizados pela câmera do smartphone e execute uma detecção de plágio, retornando os resultados em tempo real.

REFERÊNCIAS

ALLIANCE, Scrum. **Scrum Is an Innovative Approach to Getting Work Done**. Disponível em: <<https://www.scrumalliance.org/why-scrum/core-scrum-values-roles>>

AUSTIN, M.J.; BROWN, L.D. Internet Plagiarism: Developing Strategies to Curb Student Academic Dishonesty. **Internet and Higher Education**. Vol. 2, n.1, p. 21-33, 1999.

BECK, Kent; CUNNINGHAM, Ward; HUNT, Andrew; et al. **Manifesto para o desenvolvimento ágil de software**. Disponível em: <<http://www.manifestoagil.com.br>>

BRASIL. Código Penal. Decreto-Lei nº 2.848/40, de 7 de dezembro de 1940. **Vade mecum**. São Paulo: Saraiva, 2008.

CASTILLO, Oscar; MELIN, Patricia. **TYPE-2 FUZZY LOGIC: THEORY AND APPLICATIONS**, 2008.

CHONG, Frederick; CARRARO, Gianpaolo. **Estratégias de Arquitetura para Cauda Longa (Long Tail)**. Disponível em: <<https://msdn.microsoft.com/ptbr/library/aa479069.aspx>>

FOWLER, Martin. **Continuous Integration**. Disponível em: <<http://www.martinfowler.com/articles/continuousIntegration.html>>

FOWLER, Martin; SADALAGE, Pramod J. **NoSQL: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. Edição 1, p. 37, 2013.

HUMBLE, Jez; FARLEY, David. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. p. 24, 2010.

MONGODB. **NOSQL DATABASES EXPLAINED**. Disponível em:
< <http://www.mongodb.com/nosql-explained> >

ROCHA, Fábio Gomes. **Uma visão geral sobre Metodologia Ágil**. Disponível em:
<<http://www.devmedia.com.br/uma-visao-geral-sobre-metodologia-agil/27944>>

RODRIGUES, Luiza. **O que é desenvolvimento ágil**. Disponível em:
< <http://blog.myscrumhalf.com/2011/05/faq-scrum-o-que-e-desenvolvimento-agil> >

SABBAGH, Rafael. **SCRUM - GESTAO AGIL PARA PROJETOS DE SUCESSO**.
Edição 1, p. 17, 2013.