UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE COMPUTAÇÃO E TECNOLOGIA DA INFORMACÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MATEUS PASTORI

FERRAMENTA PARA DIGITALIZAÇÃO DE ACERVOS BASEADA EM *CROWDSOURCING*

CAXIAS DO SUL, RS

2015

UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE COMPUTAÇÃO E TECNOLOGIA DA INFORMACÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MATEUS PASTORI

FERRAMENTA PARA DIGITALIZAÇÃO DE ACERVOS BASEADA EM *CROWDSOURCING*

Projeto de Diplomação submetido ao curso de Bacharelado em Sistemas de Informação do Centro de Computação e Tecnologia da Informação da Universidade de Caxias do Sul, como requisito obrigatório para graduação.

Orientadora: Maria de Fátima Webber do Prado Lima.

CAXIAS DO SUL, RS

2015

MATEUS PASTORI

FERRAMENTA PARA DIGITALIZAÇÃO DE ACERVOS BASEADA EM *CROWDSOURCING*

Projeto de Diplomação submetido ao curso de Bacharelado em Sistemas de Informação do Centro de Computação e Tecnologia da Informação da Universidade de Caxias do Sul, como requisito obrigatório para graduação.

Orientadora: Maria de Fátima Webber do Prado Lima.

Aprovado em: ____/____/_____

Banca Examinadora

Profa. Dr. Maria de Fátima Webber do Prado Lima
Universidade de Caxias do Sul - UCS

Prof. Me. Giovanni Ely Rocco
Universidade de Caxias do Sul - UCS

Prof. Dr. Ricardo Vargas Dorneles
Universidade de Caxias do Sul - UCS

Dedico este trabalho à minha família, à minha namorada e aos meus amigos, em especial a minha mãe Neusa, ao meu pai Adriano e a minha irmã Tamara, por sempre me apoiarem, por estarem ao meu lado nos momentos difíceis, por estarem sempre dispostos a me ajudar, por sempre acreditarem no meu potencial, e por se constituírem diferentemente enquanto pessoas, igualmente belas e admiráveis em essência, estímulos que me impulsionaram a buscar vida nova a cada dia, meus agradecimentos por terem aceito se privar de minha companhia pelos estudos, concedendo a mim a oportunidade de me realizar ainda mais.

RESUMO

Apesar dos sistemas de reconhecimento ótico de caracteres (OCR) terem evoluído consideravelmente nas últimas décadas, eles ainda apresentam algumas falhas, principalmente tratando-se da digitalização de documentos antigos. No entanto, com a rápida expansão da internet nos últimos anos e os bilhões de usuários espalhados pelo planeta, novos paradigmas de interação humano-computador vêm ganhando força, como é o caso do *crowdsourcing*. Esse paradigma baseia-se na colaboração *on-line* em escala massiva, ou seja, o seu principal objetivo é utilizar a capacidade intelectual humana, de uma multidão de usuários, para resolver algum problema computacional aberto. Um desses problemas é o reconhecimento ótico de caracteres, que pode ser facilmente resolvido utilizando uma ferramenta baseada em *crowdsourcing*. Baseando-se nisso, esse trabalho teve como principal objetivo desenvolver um protótipo para a digitalização de acervos, que utilize um *software* de OCR, e que faça o uso do *crowdsourcing*, buscando corrigir as eventuais falhas resultantes do mesmo. O estudo de algumas ferramentas revelou que isso pode ser feito utilizando mecanismos de segurança para *websites*, denominados CAPTCHAs. A principal função de um CAPTCHA é distinguir usuários humanos de máquinas. O CAPTCHA pode ser utilizado em diversos tipos de *websites*, como *sites* de enquetes *on-line* e de contas de *e-mail*. Esse fato torna o CAPTCHA um mecanismo de *crowdsourcing* extremamente poderoso, por oferecer a possibilidade e facilidade de ser utilizado em larga escala. Definidas a arquitetura e as ferramentas a serem utilizadas, foi desenvolvido um protótipo composto de uma aplicação *web* e uma API CAPTCHA. A aplicação *web* possibilita a digitalização de documentos, para isso ela faz uso de um *software* OCR. A API CAPTCHA atua como uma ferramenta de *crowdsourcing*, cuja a função é resolver as falhas resultantes do processamento do OCR.

Palavras-chave: Reconhecimento ótico de caracteres. OCR. *Crowdsourcing*. CAPTCHA. Colaboração *on-line*.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AJAX	*Asynchronous Javascript and XML*
API	*Application Programming Interface*
CAPTCHA	*Completely Automated Public Turing test to tell Computers and Humans Apart*
CC	*Componente Conexo*
EER	Entidade Relacionamento Estendido
GWIDO	*Games With Interaction Design Objective*
GWAPs	*Games With a Purpose*
HTML	*HyperText Markup Language*
IUT	*International Telecommunication Union*
JSON	*JavaScript Object Notation*
MVC	*Model-view-controller*
OCR	*Optical Character Recognition*
PDF	*Portable Document Format*
PHP	*PHP: Hypertext Preprocessor*
REST	*Representational State Transfer*
UML	*Unified Modeling Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Pesquisas realizadas recentemente pela ITU (*International Telecommunication Union*) estimam que pelo menos 2.7 bilhões de pessoas possuem acesso à internet, ou seja, cerca de 39% da população mundial. Com a crescente e acelerada evolução tecnológica dos meios de comunicação, e a rápida expansão da Internet pelo planeta, esse número vem aumentando consideravelmente a cada ano, principalmente pela ascensão das redes sociais, sites de compras *on-line*, jogos virtuais, portais de notícias e conteúdos multimídia (DONELAN, 2010).

O imenso volume de pessoas conectadas em rede possibilitou o surgimento de um novo paradigma denominado *crowdsourcing*, que é um modelo de criação e/ou produção, que conta com a mão-de-obra e conhecimento coletivo, para desenvolver soluções e criar produtos. O termo *crowdsourcing* é utilizado principalmente para caracterizar sistemas de colaboração *on-line* em escala massiva ou de forma mais genérica, sistemas de colaboração *on-line*. A ideia de sistemas de colaboração *on-line*, em sua grande maioria, é utilizar o poder de processamento humano para resolver problemas que os computadores ainda não podem resolver sozinhos (BRABHAM, 2008).

Há vários exemplos de *softwares* que utilizam o *crowdsourcing*, desde jogos para *smartphones*, que auxiliam na busca da cura do câncer, até aplicativos de ensino de idiomas, que traduzem páginas da *web* enquanto os usuários aprendem uma nova língua. Esses *softwares* tem a capacidade de aproveitar o tempo e a inteligência intelectual humana para auxiliar na resolução de algum problema, sem que o usuário, muitas vezes, se dê conta disso.

Um dos grandes desafios computacionais é o reconhecimento de caracteres em arquivos de imagem. A digitalização de acervos impressos em papel, vem se tornando cada vez mais necessária por diversos fatores. E por mais modernos que sejam os *softwares* de reconhecimento de caracteres, também chamados de *softwares* de OCR (*Optical Character Recognition*), eles geralmente não conseguem reconhecer todas as palavras presentes em um documento digitalizado. Esse é um dos problemas que pode ser resolvido utilizando o poder da multidão de usuários espalhados pela internet. De fato, as palavras não reconhecidas pelo OCR podem ser enviadas para usuários comuns da internet decifrarem.

A digitalização de acervos é uma das ferramentas essenciais ao acesso e à difusão dos acervos arquivísticos constituindo-se como instrumento capaz de dar acesso simultâneo local ou remoto aos documentos digitais (documentos textuais, cartográficos e iconográficos em suportes convencionais). Além disso, essa contribui para a preservação do acervo, uma vez que restringe o manuseio dos originais (CONARQ, 2010).

Atualmente, as empresas estão adotando processos que permitam digitalizar seus documentos, sejam eles contratos, notas fiscais, processos jurídicos, entre outros documentos importantes. Segundo o CONARQ, documentos em formato digital fornecem diversas vantagens, entre as quais pode-se destacar (CONARQ, 2010):

- Contribuir para o amplo acesso e disseminação dos documentos arquivísticos por meio da Tecnologia da Informação e Comunicação;
- Permitir o intercâmbio de acervos documentais e de seus instrumentos de pesquisa por meio de redes informatizadas;
- Promover a difusão e a reprodução dos acervos arquivísticos não digitais, em formatos e apresentações diferenciados do formato original;
- Incrementar a preservação e segurança dos documentos arquivísticos originais que estão em outros suportes não digitais, por restringir seu manuseio;
- O acesso rápido à informação e à preservação de documentos históricos, evitando o manuseio do documento no formato papel;
- Possibilidade de acesso rápido contribuindo para decisões rápidas e aumento da produtividade;
- Segurança, portabilidade e conectividade;
- Economia do espaço físico, com a eliminação de documentos no formato papel.

No entanto o custo e o tempo consumido para a digitalização de grandes acervos ainda é muito elevado. Há algumas empresas, no mercado, que fornecem esse tipo de serviço, mas a maioria disponibiliza apenas o serviço de digitalização do documento em formato de imagem. Juntamente com esse serviço é disponibilizado o serviço de indexação de arquivos. O processo de indexação de arquivos consiste no armazenamento de algumas informações, referentes a esses arquivos, em uma base de dados. Essa indexação tem como objetivo facilitar futuras consultas. No entanto, poucas empresas fornecem o serviço que torna possível a edição de documentos digitalizados, através de um editor de textos, e isso apenas é possível através do uso de *softwares* de OCR.

Os algoritmos de reconhecimento ótico de caracteres (OCR) apresentam limitações, sendo que para garantir uma tradução correta de palavras que o algoritmo não conseguiu identificar, é possível utilizar um sistema que utilize o poder da colaboração *on-line* em escala massiva, onde seres humanos farão o reconhecimento dessas palavras.

O projeto denominado reCAPTCHA já vem utilizando, há algum tempo, esse conceito de *crowdsourcing* em conjunto com sistemas de OCR para auxiliar na digitalização de livros. A ideia do reCAPTCHA consiste em utilizar os seres humanos para identificar as palavras que os algoritmos de OCR não conseguiram traduzir corretamente. A nível de curiosidade, os algoritmos de OCR não conseguem reconhecer cerca de 30% das palavras de um livro antigo (AHN, 2011). No entanto, esse projeto é fechado, ou seja, não é possível que usuários comuns enviem arquivos para serem digitalizados. A fim de disponibilizar esse serviço de forma livre, o principal objetivo deste trabalho consiste no desenvolvimento de uma API (*Application Programming Interface*) baseada no mesmo conceito utilizado pelo reCAPTCHA. Posteriormente, será desenvolvido um *website* simples que permita validar a API desenvolvida.

1.1 OBJETIVOS

O objetivo desse trabalho consistiu no desenvolvimento de uma aplicação *web* que dê suporte a digitalização de acervos. Para tanto, a aplicação faz o uso de um *software* OCR, juntamente com uma ferramenta de colaboração *on-line* baseada no mesmo conceito utilizado pelo reCAPTCHA. Destaca-se que o aplicativo não faz a digitalização e o reconhecimento dos caracteres de forma instantânea, ou seja, para a utilização da mesma devem ser utilizados arquivos de imagens previamente digitalizadas.

1.1.1 Objetivos específicos

De forma a atingir o objetivo geral apresentado, o trabalho foi orientado pelos seguintes objetivos específicos:

- Desenvolvimento de um aplicativo para reconhecimento de caracteres, utilizando uma biblioteca OCR;
- Desenvolvimento de API baseada na ideia do reCAPTCHA, para que *websites* possam utilizá-la como ferramenta de segurança;
- Desenvolvimento de um *website* simples que permita validar a API desenvolvida.

1.2 ESTRUTURA DO TRABALHO

O segundo capítulo aborda os conceitos básicos de *crowdsourcing*, como surgiu, o que significa e como ele pode ajudar a humanidade. Nesse capítulo também são abordados os conceitos de computação humana e como ela pode trabalhar de forma conjunta com sistemas baseados no *crowdsourcing*. Além disso, cita exemplos de sistemas que utilizam testes de *turing* automatizados, como é o caso do CAPTCHA, e como ele pode ser usado para digitalizar livros em larga escala.

No terceiro capítulo são abordados os principais componentes de um sistema OCR, seu funcionamento, seus pontos críticos, falhas e soluções. Nesse capítulo também são estudados alguns *softwares* OCR *open source*, para a definição de qual seria utilizado na elaboração da proposta de solução.

No quarto capitulo é exposta a proposta de solução onde são descritas as ferramentas, os padrões de projeto, as linguagens de programação e banco de dados que serão utilizados para o desenvolvimento do protótipo. Nesse capitulo também são abordados os requisitos funcionais a serem atendidos, e também a modelagem das principais classes e tabelas envolvidas no desenvolvimento da aplicação.

No quinto capítulo são expostos os detalhes sobre a solução desenvolvida. Nele são destacadas a arquitetura, as ferramentas e tecnologias utilizadas durante o desenvolvimento do protótipo.

No sexto capítulos são descritos os testes realizados e os resultados obtidos para a validação da aplicação desenvolvida.

No último capítulo do trabalho são apresentadas as considerações finais que sintetizam os resultados encontrados, além de abrir a possibilidade de futuros projetos.

2 COLABORAÇÃO *ON-LINE* EM ESCALA MASSIVA (*CROWDSOURCING*)

O conceito de *crowdsourcing* surgiu em 2005 e foi concebido por Jeff Howe e Mark Robinson, depois de conversas sobre empresas que estavam usando a força de trabalho de uma multidão de usuários, espalhados pela internet, para terceirizar os seus serviços. Jeff Howe e Mark Robinson chegaram à conclusão de que o que estava acontecendo era uma "terceirização para a multidão". Juntando os termos, multidão e terceirização, em inglês respectivamente, *crowd* e *outsourcing*, foi criado um novo termo denominado *crowdsourcing*.

Após estudar mais de quarenta definições de *crowdsourcing* na literatura científica e popular, Estellés e González (2012) desenvolveram uma nova definição para o termo:

> *Crowdsourcing* é um tipo de atividade *on-line* participativa na qual um indivíduo, uma instituição, uma organização sem fins lucrativos, ou companhia propõe a um grupo de indivíduos de conhecimentos variados, heterogêneos, e número, o empreendimento voluntário de uma tarefa por meio de um chamado aberto e flexível. O empreendimento da tarefa, de complexidade variável e modulável, e no qual o grupo deve participar contribuindo com seu trabalho, dinheiro, conhecimento e/ou experiência, sempre implica em benefícios mútuos. O usuário receberá em troca benefícios que podem ser de origem econômica, reconhecimento social, autoestima, ou o desenvolvimento de habilidades individuais, enquanto o *crowdsourcer* receberá as contribuições oferecidas pelo usuário ao empreendimento, na qual o formato dependerá do tipo de atividade empreendida (ESTELLÉS; GONZÁLEZ, 2012).

Segundo Howe (2008), o conceito de *crowdsourcing* teve início em benefício dos *softwares open source* (livres), ou de código aberto. O sistema operacional Linux, o *software* para servidores Apache e o navegador Firefox, são alguns exemplos de como uma comunidade de pessoas, que pensam de forma similar, é capaz de criar um produto de alta qualidade, e que compete com *softwares* de grandes corporações como, por exemplo, a Microsoft. Em suma, a motivação dessa comunidade de pessoas, é criar algo que trará benefícios a todos, e é nisso que o *crowdsourcing* se sustenta. A constante evolução dos *softwares* de código aberto se deve a cooperação ou colaboração de desenvolvedores espalhados pela rede, que ao encontrarem uma falha ou uma necessidade de melhoria, estão prontamente dispostos a corrigi-lo, ou melhorá-lo. Essa intensa interação acaba tornando o produto cada vez mais seguro, interativo e de fácil usabilidade (HOWE, 2008).

Howe (2008), destaca o potencial que a internet tem de interligar a humanidade bem como de fazer disso um organismo próspero e infinitamente poderoso. O constante crescimento da internet, permite explorar uma forma de trabalho humana muito primitiva, a divisão de tarefas. Dividir uma tarefa, como escrever uma exaustiva enciclopédia, em pequenos pedaços, torna o processo muito mais rápido e gera um resultado final de alta qualidade (HOWE, 2008). Para Estellés e González (2012), a enciclopédia *on-line*, Wikipédia, é um exemplo de que a cooperação ou colaboração, que vai além do desenvolvimento de *softwares* de código aberto.

A Wikipédia é uma enciclopédia livre, em que qualquer pessoa, de qualquer parte do mundo pode colaborar com a criação ou edição de artigos. Atualmente, ela conta com aproximadamente 14 milhões de artigos em centenas de línguas e dialetos (ESTELLÉS; GONZÁLEZ, 2012).

Howe (2008), define ainda que o termo *crowdsourcing* refere-se a criação, execução, solução, inovação, e que pode vir de qualquer lugar, não importando a origem, raça, sexo, idade e qualificação do colaborador. A promessa do *crowdsourcing* é libertar o potencial criativo de um indivíduo para se destacar em mais de uma área de interesse. Grandes empresas vêm adotando o *crowdsourcing* pois este utiliza a inteligência coletiva, favorece constante inovação, possibilita melhores resultados, e ainda reduz os custos (HOWE, 2008).

Para Howe (2008), o *crowdsourcing* impulsiona a globalização da mão-de-obra e o deslocamento econômico. A exemplo da internet, seu meio de operação, o *crowdsourcing* não tem fronteiras. Ou seja, a rede não se importa se o usuário está na mesma rua, mesmo estado, ou do outro lado do mundo, esse é o grande benefício do uso de *crowdsourcing* (HOWE, 2008). Para Quinn e Bederson (2011), o *crowdsourcing* é um paradigma de aquisição da informação emergente e poderosa que apareceu sob muitos nomes, incluindo computação social, inteligência coletiva e computação humana (QUINN; BEDERSON, 2011).

2.1 CLASSIFICAÇÃO DE SISTEMAS *CROWDSOURCING*

Segundo Doan (2011), sistemas de *crowdsourcing* podem ser classificados em várias dimensões. A Tabela 1 apresenta as diversas classificações possíveis para sistemas *crowdsourcing*, bem como, alguns exemplos de sua aplicação. Da esquerda para a direita, a tabela é organizada pela natureza da colaboração, arquitetura, necessidade de recrutar os usuários e ações que os usuários podem tomar.

2.2 COMPUTAÇÃO HUMANA

Na ciência da computação, computação humana, é um paradigma no qual um computador terceiriza algumas etapas do processo para serem computadas por seres humanos. Computadores foram criados, inicialmente, para auxiliar os seres humanos na resolução de problemas. Nesse caso, humanos interagem com um computador fornecendo descrições detalhadas de um problema, para que este seja processado e forneça um resultado para ser interpretado. Na computação humana frequentemente os papéis são invertidos, onde o computador utiliza um grupo de pessoas para resolver um problema, então coleta, processa e integra os resultados. Dessa forma, parte do processamento é realizado por computadores e parte por humanos (QUINN; BEDERSON, 2011).

TABELA 1 - Divisão de sistemas *Crowdsourcing*

Natureza da colaboração	Arquitetura	Recruta usuários?	O que os usuários fazem?	Exemplos	Problemas alvo	Comentários
Explicitas	*Standalone*	Sim	Avaliação • Revisão, voto, rotulação.	• Revisar e votar na Amazon.	Avaliação de uma coleção de itens (por exemplo, produtos, usuários)	Seres humanos como os prestadores de perspectiva. Ausência ou combinação flexível das contribuições.
			Compartilhamento • Itens; • Conhecimento textual; • Conhecimento estruturado.	• Napster, YouTube, FlickrMailing lists, Yahoo! Answers, QUIQ, ehow.com; • Swivel, Many Eyes, Google Fusion Tables, Google Base, bmrb.wisc.edu.	Construção de uma coleção de itens (distribuídos ou centrais) que podem ser compartilhados entre os usuários.	Seres humanos como os provedores de conteúdo. Ausência ou combinação frouxa das contribuições.
			Networking	• Linkedin, MySpace, Facebook.	Construção de redes sociais	Seres humanos como os provedores de componentes. Ausência ou combinação flexível das contribuições.
			Construindo artefatos • Software; • Bases de conhecimento textual; • Bases de conhecimento estruturadas; • Sistemas; • Outros.	• Linux, Apache, Hadoop; • Wikipedia, openmind, Intellipedia, ecolicommunity; • Wikipedia infoboxes/DBpedia, IWP, Google Fusion Tables; • Wikia Search, mahalo, Freebase, Eurekster; • Newspaper at Digg.com, Second Life.	Construção de artefatos físicos	Os seres humanos podem desempenhar todas as funções. Normalmente uma combinação firme das contribuições. Alguns sistemas pedem para seres humanos e máquinas para contribuir.
			Execução de tarefas	• Encontrar extraterrestres, eleições, encontrar pessoas, criação de conteúdo (por exemplo a Demand Media, Associated Content).	Possivelmente qualquer problema	

(Continua)

(Conclusão)

Natureza da colaboração	Arquitetura	Recruta usuários?	O que os usuários fazem?	Exemplos	Problemas alvo	Comentários
Implícitas	*Standalone*	Sim	• Jogar jogos com um objetivo; • Apostar em mercados de previsão; • Uso de contas privadas; • Resolver CAPTCHAs; • Comprar/vende/leiloar, jogar jogos com *multiplayers* maciços.	• ESP; • intrade.com, Iowa Electronic Markets; • IMDB private accounts; • recaptcha.net; • eBay, World of Warcraft.	• Rotulação; • Previsão de eventos; • Classificando filmes; • Digitalizando texto escrito; • Construção de uma comunidade de usuários (para fins de cobrança de taxas, publicidade).	Os seres humanos podem desempenhar todas as funções. Combinação flexível ou firme das contribuições.
	Piggyback em outro sistema	Não	• Busca por palavra-chave; • Comprar produtos; • Procurar websites.	• Google, Microsoft, Yahoo; • O recurso de recomendação da Amazon; • Web sites adaptativos (por exemplo, a primeira página do Yahoo!).	• Correção ortográfica, previsão de epidemias; • Recomendação de produtos; • Reorganizar um site para um melhor acesso.	Os seres humanos podem desempenhar todas as funções. Combinação flexível ou firme das contribuições.

Fonte: Adaptada de DOAN, 2011.

A partir da união dos dois paradigmas, *crowdsourcing* e computação humana, podem-se criar sistemas de colaboração *on-line* em escala massiva ou de forma mais genérica, sistemas de colaboração *on-line*. Ou seja, sistemas que utilizam o poder de processamento humano de uma multidão de usuários, espalhados pela internet, para realizar algumas tarefas que envolvam a computação humana (QUINN; BEDERSON, 2011).

Sistemas de colaboração *on-line* se concentram em aproveitar o tempo e a energia humana para resolver tais problemas. Tarefas como reconhecimento de imagens são triviais para seres humanos, mas continuam desafiando até mesmo os programas de computadores mais sofisticados. Segundo Luis Von Ahn (2005), embora os computadores tenham avançado significativamente em muitos aspectos, eles ainda não possuem a capacidade intelectual ou a percepção conceitual básica dos seres humanos (ANH, 2005).

Ainda segundo Luis Von Ahn (2005), neste paradigma de colaboração *on-line*, os cérebros humanos são tratados como processadores em um sistema distribuído, onde cada um executa uma parte de uma computação massiva. Ao contrário dos processadores, os seres humanos necessitam de um incentivo para tornar-se parte de um processo coletivo. Jogos *on-line* são uma forma de incentivar a participação nesse processo. Assim, os jogos constituem basicamente um mecanismo geral que visa usar o poder do cérebro humano para resolver problemas computacionais abertos. Dessa forma, cada problema exige uma concepção cuidadosa para que o jogo desenvolvido seja agradável e, ao mesmo tempo, garanta a resolução correta das instâncias do problema. A concepção de tais jogos é similar ao projeto de algoritmos de computador, mas ao invés de usar um processador de silício, esses "algoritmos" executam em um sistema composto por seres humanos que interagem com computadores através da Internet (ANH, 2005).

2.3 CAPTCHA

Há alguns anos Luis Von Ahn introduziu uma nova ferramenta de segurança para *websites* denominada **CAPTCHA** (***C****ompletely* ***A****utomated* ***P****ublic* ***T****uring test to tell* ***C****omputers and* ***H****umans* ***A****part*). Essa ferramenta impede que "robôs", ou seja, *scripts* que simulam as ações humanas inúmeras vezes e de maneira padrão, criem, por exemplo, milhares de contas de *e-mail* em um *website* que disponibiliza esse serviço, ou ainda, para evitar que *websites* que necessitem de *login*, sofram ataques de invasão por força bruta. Os CAPTCHAs nada mais são do que testes para verificar se o usuário em questão é um humano. Esses testes consistem em exibir uma imagem com algumas letras e números, embaralhados e distorcidos, que somente um humano pode distinguir (ANH, 2005).

Os CAPTCHAs são basicamente testes de *Turing* automatizados. No teste de *Turing* original, um juiz humano foi autorizado a fazer uma série de perguntas para dois jogadores, um deles era um computador e outro um ser humano. Ambos os jogadores fingiam ser um humano, e o juiz teve que distinguir qual deles era realmente um ser humano. Os CAPTCHAs são semelhantes aos testes de *Turing*, onde o objetivo é distinguir os humanos de computadores, mas a diferença é que o juiz, nesse caso, é um computador (ANH, 2005).

Anh (2005), destaca que atualmente CAPTCHAs são úteis para muitas aplicações, sendo que entre essas se destacam (ANH, 2005):

- Enquetes *on-line*: na maioria das enquetes *on-line*, os endereços IP dos eleitores são registrados para evitar que os usuários votem mais de uma vez. No entanto, já existem formas de burlar esse sistema de registros de IP, e isso acaba tornando as enquetes *on-line* pouco confiáveis. O uso de um registro de IP combinado com o uso de um CAPTCHA, para validar se o eleitor em questão é um ser humano, é o ideal;
- Serviço gratuito de *e-mail*: várias empresas como por exemplo, Google, Yahoo! e Microsoft, oferecem serviços de *e-mail* gratuito. A maioria delas sofria com algum tipo de ataque realizado por "robôs", que criavam milhares de contas de *e-mail* a cada minuto. Esse problema foi solucionado exigindo que os usuários provem que são humanos, antes que eles possam criar a conta de *e-mail* gratuita. A maioria dessas empresas adota algum tipo de CAPTCHA para impedir que "robôs" criem contas falsas;
- Robôs de motores de busca: alguns sites não querem ser indexados por motores de busca. Existe uma *tag* HTML que evita que os "robôs" de motores de busca, indexem o conteúdo das páginas do site, mas isso não garante que esses "robôs" não leiam o conteúdo das páginas. A fim de garantir que esses "robôs" não leiam o conteúdo do *site*, o uso de algum tipo de CAPTCHA pode ser necessário;
- Prevenção de ataques de dicionário (força bruta): os CAPTCHAs também podem ser usados para evitar ataques de dicionário em sistemas que requerem autenticação através de usuário e senha. De fato, o uso de um CAPTCHA evita que um computador seja capaz para percorrer todo o espaço de senhas.

2.3.1 Exemplos de CAPTCHAs

Pode-se considerar um CAPTCHA qualquer tipo de teste que utilize alguma capacidade sensorial humana e que possa interagir com o usuário através de um computador. Alguns exemplos de CAPTCHAs são:

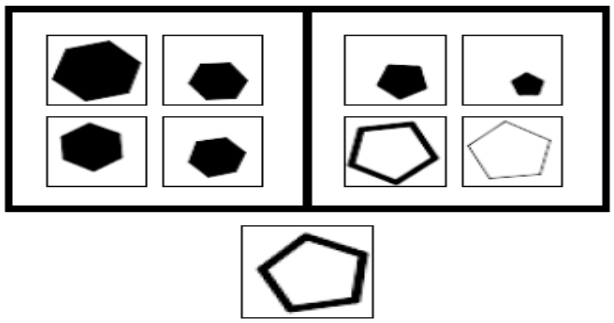
- CAPTCHAs baseados em OCR (*Optical Character Recognition*): esse é um dos CAPTCHAS mais usados atualmente, e está baseado na dificuldade de leitura de um texto distorcido. Esses baseiam-se na escolha de algumas palavras de um dicionário que são processadas em uma imagem distorcida (Figura 1). O usuário então deve identificar e digitar corretamente as palavras distorcidas. A maioria dos seres humanos conseguem ler as palavras dessa imagem distorcida, mas *softwares* atuais não possuem essa capacidade. Dessa forma, os CAPTCHAs baseados em OCR se baseiam na dificuldade de reconhecimento óptico de caracteres, ou seja, a dificuldade de ler um texto distorcido;

FIGURA 1 – Exemplo de CAPTCHA baseado em OCR

Fonte: FACEBOOK.

- CAPTCHAs baseados no reconhecimento de padrões visuais: Anh (2005) cita o Bongo com um exemplo de CAPTCHA baseado no reconhecimento de padrões visuais. Esse CAPTCHA exibe dois conjuntos de blocos, um a esquerda e outro à direita (Figura 2). Os blocos do conjunto da esquerda diferem daqueles do conjunto direita, e o usuário deve encontrar a característica que os diferencia. Depois de ver os dois conjuntos de blocos, é apresentado ao usuário um único bloco, e então ele deve determinar se este bloco pertence ao conjunto da direita ou ao conjunto da

FIGURA 2 – Exemplo do CAPTCHA Bongo

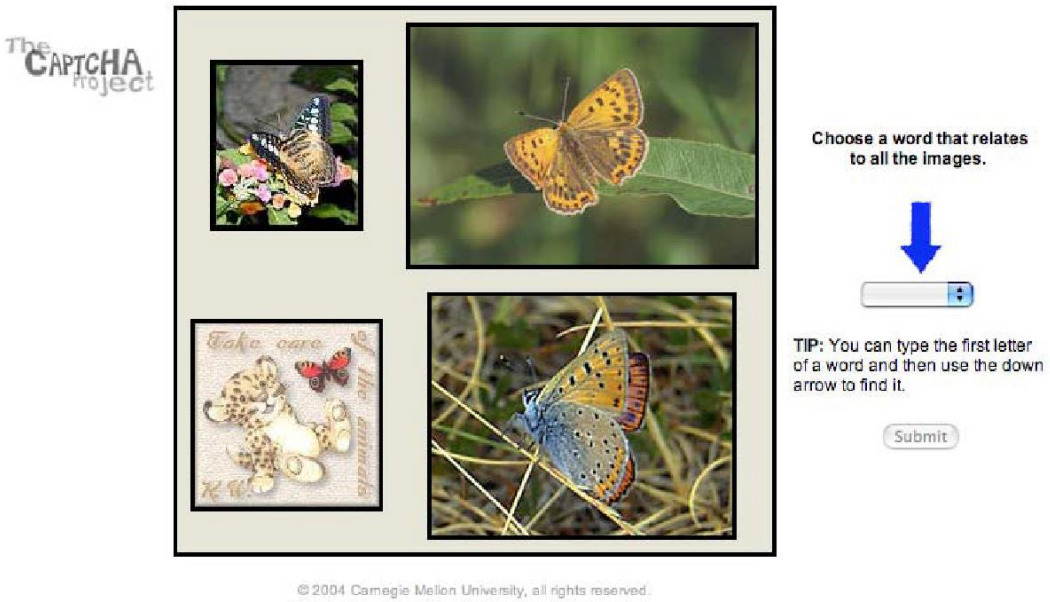


Fonte: (ANH, 2005).

esquerda (Figura 2). Assim, o usuário passa no teste se ele determinar corretamente a qual lado pertence o bloco (ANH, 2005);

- CAPTCHAs baseado no reconhecimento de objetos em imagens: o Pix é outro CAPTCHA citado por Anh (2005), sendo que esse baseia-se no reconhecimento de objetos em imagens. Esse tipo de CAPTCHA possui um banco de dados com um grande acervo de imagens, sendo que essas imagens estão indexadas conforme os objetos que aparecem nelas. Seu funcionamento baseia-se na escolha de um objeto qualquer, e em seguida são procuradas, em seu banco de dados, quatro imagens que possuem esse objeto. Posteriormente, essas imagens são apresentadas para o usuário que deve responder a seguinte pergunta "O que há nessas imagens?" (Figura 3). O Pix pode ser considerado um CAPTCHA, uma vez que os s*oftwares* atuais ainda não são capazes de fazer o reconhecimento de objetos em imagens (ANH, 2005);

FIGURA 3 – Exemplo do CAPTCHA Pix



Fonte: (ANH, 2005).

- CAPTCHAs baseados em sons: o último CAPTCHA a ser citado é baseado em sons. Neste tipo de CAPTCHA, o programa escolhe uma palavra ou uma sequência de números aleatoriamente, e converte-os para um formato sonoro e distorcido. Em seguida, o usuário deverá digitar o conteúdo existente nesse som, para passar no teste. Este CAPTCHA baseia-se na diferença de capacidade entre os seres humanos e os computadores no reconhecimento da linguagem falada.

Os CAPTCHAs tiram proveito da capacidade humana, a fim de diferenciar os seres humanos de computadores, e fazer isso tem importantes aplicações práticas. A além disso, há muitas tarefas que os humanos podem facilmente fazer que os computadores ainda não podem resolver. Na próxima seção, veremos como um CAPTCHA pode ser usado a fim de resolver um problema computacional aberto.

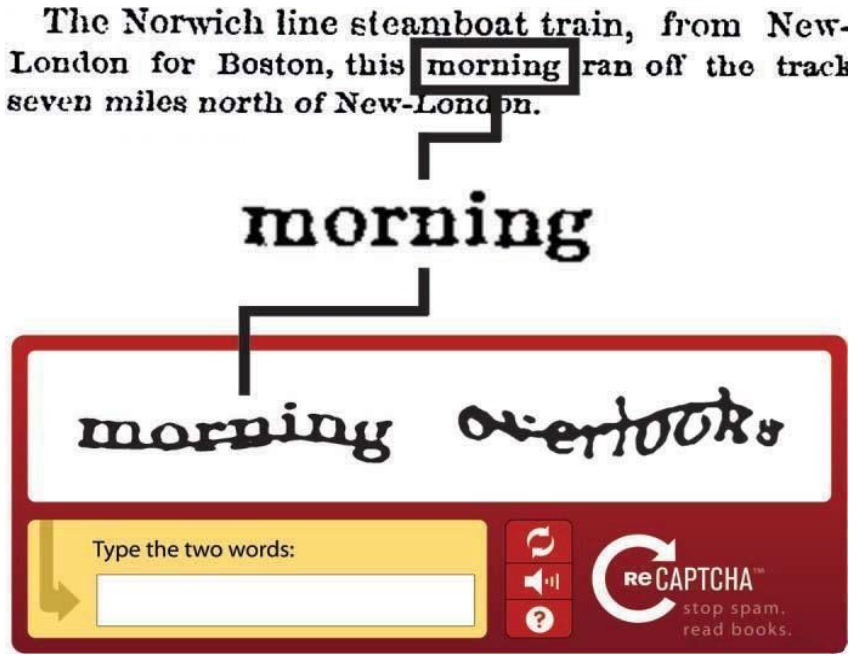
2.4 RECAPTCHA

Com milhares de *websites* adotando o CAPTCHA baseado em OCR como ferramenta de segurança, e cerca de 200 milhões de CAPTCHAs digitados diariamente, Luis Von Ahn percebeu que esse sistema poderia ser utilizado para algo útil, foi então que surgiu o reCAPTCHA. O funcionamento do reCAPTCHA é basicamente o mesmo do CAPTCHA, mas ele implementa um outro conceito, isso é, além de testar se o usuário é um humano, ele também ajuda a digitalizar livros.

O reCAPTCHA utiliza duas palavras para testar se o usuário realmente é um humano. O motivo pelo qual ele utiliza duas palavras ao invés de uma, consiste no fato de que uma das palavras foi digitalizada, mas não foi identificada através do algoritmo de OCR (Figura 4). Dessa forma, o sistema não sabe qual é a resposta para essa palavra, então ele não pode avaliar se o usuário é um humano. O que vai testar isso é uma segunda palavra, que possui o resultado conhecido pelo sistema. Para garantir o reconhecimento correto para a palavra desconhecida, que foi retirada do livro, o sistema envia a mesma palavra para inúmeros usuários, se todos, ou a maior parte deles, concordarem com o mesmo resultado, a nova palavra é digitalizada (AHN, 2011). Em média, para digitar um CAPTCHA, cada usuário demora 10 segundos, multiplicando esse número por 200 milhões, temos 550 mil horas de esforço humano em um único dia. Dessa forma, apenas com o uso do reCAPTCHA é possível digitalizar cerca 2,5 milhões de livros anualmente (AHN, 2011). Esse número é significativo, se for considerado o tempo e os custos que uma empresa especializada levaria para executar a mesma tarefa. Com isso é possível perceber que o reCAPTCHA é uma poderosa ferramenta de *crowdsourcing*, em que é

aproveitado um curto espaço de tempo do usuário, mas que multiplicado pelos milhões de usuários espalhados pela internet, é possível realizar a digitalização de um grande acervo de livros ou documentos.

FIGURA 4 – reCAPTCHA



Fonte: (ANH, 2008).

2.5 TRABALHOS RELACIONADOS

Sistemas baseados em c*rowdsourcing* permitem a invocação flexível e a utilização em larga escala da contribuição humana para a coleta e análise de dados, que introduz um novo paradigma de processo de mineração de dados. Os métodos tradicionais de mineração de dados, muitas vezes necessitam de especialistas em domínios analíticos para anotar os dados. No entanto esse processo é dispendioso e, normalmente, demora um longo tempo. O *crowdsourcing* permite o uso do conhecimento heterogêneo de voluntários e distribui o processo de anotação em pequenas partes (XINTONG, 2014).

Em alguns tipos de cenários, o *crowdsourcing* pode ajudar as pessoas a resolver problemas de uma forma mais eficiente. Um exemplo de problema enfrentado na mineração de dados é o de classificação dos dados. Os dados, geralmente, são rotulados e usados para treinar um classificador que posteriormente deverá classificar novos dados (XINTONG, 2014).

Além da classificação, vários outros tipos de funções de mineração de dados podem ser realizados por meio de *crowdsourcing*, como, por exemplo, *clustering*, aprendizagem semi-supervisionada, e regras de associação. Algoritmos tradicionais têm dificuldades em lidar com esses problemas, pela falta de conhecimento. Nessas situações, o poder das multidões pode

realizar essas tarefas de forma mais precisa, flexível e eficiente do que os algoritmos existentes (XINTONG, 2014).

Outra aplicação potencial para o *crowdsourcing* é o apoio a tomada de decisões. Um problema complicado que é difícil para uma pessoa, pode ser resolvido pela multidão. Multidões podem fornecer ideias de forma colaborativa ou em um modo competitivo. No entanto, o papel da multidão pode ser diferente em diferentes fases do processo de tomada de decisão. Pode-se utilizar a multidão para fornecer informações sobre um problema complexo ou usar a multidão para ajudar a decidir se um projeto é útil. Esse processo de tomada de decisão geralmente é dividido em três fases principais (CHIU; LIANG; TURBAN, 2014):

- Inteligência: Nessa fase as informações são coletadas e compartilhadas com a finalidade de resolução de problemas ou a exploração de oportunidades, identificação de problemas, bem como a determinação da importância do problema;
- *Design*: Nessa fase as ideias e/ou soluções alternativas são geradas pela multidão;
- Escolha: Finalmente, nessa fase, ocorre a avaliação das alternativas geradas e, em seguida, é selecionado o melhor curso de ação.

Crowdsourcing ainda pode ser usado como ferramenta em atividades de marketing. Segundo Whitla (2009), há três áreas de aplicação *crowdsourcing* nessa área (GATAUTIS; VITKAUSKAITE, 2013, apud WHITLA, 2009):

- Desenvolvimento de produtos;
- Promoção e publicidade;
- Pesquisa de marketing.

Para o desenvolvimento de novos produtos algumas empresas utilizam o potencial das tecnologias de comunicação e informação para a obtenção de entrada e/ou conselhos dos seus consumidores existentes ou potenciais. Em outros casos, as empresas buscam especialistas que identifiquem problemas que podem ser resolvidos através de determinados produtos. Em alguns casos as empresas desafiam os consumidores a fornecer um projeto para produto, seleciona o melhor e desenvolve em parceria com o autor (GATAUTIS; VITKAUSKAITE, 2013).

Nas atividades de promoção e publicidade Whitla (2009) distingue dois casos de implantação de *crowdsourcing*. Um deles é a busca de especialistas que podem realizar tarefas (criar *design*, criar *flyer*, criar *banner*, etc.), outro caso é o uso de *crowdsourcers* para tarefas

bastante trabalhosas, no caso da empresa não ter tempo nem recursos humanos para concluí-las (GATAUTIS; VITKAUSKAITE, 2013, apud WHITLA, 2009).

Nas pesquisas de *marketing* a perspectiva *crowdsourcing* dá oportunidade de alcançar um grande grupo de potenciais consumidores. Naturalmente, elementos motivacionais são necessários para manter o interesse e o envolvimento, mas em muitos casos o *crowdsourcing* oferece oportunidades mais baratas e mais rápidas para a coleta de informações de mercado (GATAUTIS; VITKAUSKAITE, 2013).

2.6 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o conceito de *crowdsourcing*, e como sistemas baseados nesse paradigma auxiliam na resolução de problemas computacionais abertos. Esse capítulo também abordou a utilização da computação humana para a resolução de problemas que os computadores e *softwares* atuais não podem resolver sozinhos, como por exemplo, o reconhecimento de caracteres em documentos digitalizados. Os sistemas de reconhecimento ótico de caracteres atuais não são capazes de reconhecer, em muitos casos, todas as palavras de um documento digitalizado. Para isso, a computação humana é utilizada, mas como, geralmente, esse esforço requer muito tempo, de acordo com o tamanho do acervo a ser digitalizado, é necessário distribuir essa tarefa em pequenas partes.

Essa distribuição de tarefas pode ser feita utilizando um sistema semelhante ao reCAPTCHA. O reCAPTCHA é um projeto fechado, ou seja, não é possível enviar documentos digitalizados para que ele possa interpretar. Para isso se faz necessário o uso de um sistema de OCR que interprete o máximo de palavras possíveis e o restante, das palavras não reconhecidas, sejam encaminhadas para um sistema que o utilize o poder da computação humana distribuída.

No próximo capítulo são descritos os principais componentes de um sistema de reconhecimento ótico de caracteres (OCR), onde será possível conhecer as suas falhas e as técnicas utilizadas para a identificação de caracteres e palavras. Essa descrição será feita para que se tenha uma visão geral do funcionamento de um sistema de OCR, buscando compreender também o comportamento de cada componente, desse sistema, no processo de digitalização. Com isso será possível identificar quais etapas estão envolvidas no reconhecimento das palavras digitalizadas, quais são as suas principais falhas, como elas ocorrem e suas possíveis soluções.

3 RECONHECIMENTO ÓTICO DE CARACTERES (OCR)

O reconhecimento ótico de caracteres é feito, geralmente, utilizando algoritmos de OCR. Estes algoritmos são responsáveis pelo reconhecimento de caracteres a partir de arquivos de imagens, sejam eles escaneados, escritos a mão, datilografados ou impressos. Assim, com o uso destes algoritmos, é possível obter arquivos de texto editáveis.

A digitalização de um documento tem início quando esse passa por um equipamento de *scanner*, sendo que a imagem gerada a partir desse equipamento é processada para que todas as regiões que contenham algum tipo de texto sejam localizadas. Após, cada uma dessas regiões é segmentada e os símbolos (caracteres) presentes nela são extraídos.

Posteriormente, esses símbolos passam por um pré-processamento, onde algumas imperfeições são removidas. A remoção das imperfeições garante que haja um melhor reconhecimento das características de cada símbolo. As características então, são comparadas com as descrições das classes de símbolos, que se originam através da fase de aprendizado do sistema de OCR. Finalmente, os caracteres reconhecidos são agrupados para a formação do texto original.

Nas seções a seguir são apresentadas as etapas do processo de digitalização e as bibliotecas OCR disponíveis.

3.1 PROCESSO DE DIGITALIZAÇÃO

Para Eikvil (1993), esse processo de digitalização pode ser dividido, basicamente, em seis etapas (EIKVIL, 1993):

- Escaneamento ótico;
- Localização e segmentação;
- Pré-processamento;
- Extração de características;
- Classificação;
- Pós-processamento.

3.1.1 Escaneamento Ótico

Através do escaneamento óptico é possível obter uma imagem digitalizada a partir de um documento. Geralmente documentos são impressos em preto-e-branco, por isso é uma boa prática converter uma imagem colorida, em uma imagem de níveis de cinza ou em preto-e-branco. Este processo, de conversão de tons de cinza, é chamado de "*thresholding*", sendo que esse é frequentemente utilizado para otimizar o uso de memória e reduzir o processamento computacional (EIKVIL, 1993).

O processo de *thresholding* possui um grande impacto no processo de reconhecimento de caracteres. Esse processo é relativamente simples, nele um valor limite, entre o que pode ser considerado preto ou branco, é definido. Os níveis de cinza abaixo deste valor são tratados como preto e níveis de cinza acima deste valor são tratados como branco. Em documentos com alto contraste entre o preto e o branco, apenas um nível pode ser o suficiente, mas a maioria dos documentos não possuem essa característica. Sendo assim, esses valores limites são modificados dinamicamente até que se obtenha um resultado satisfatório. (EIKVIL, 1993).

3.1.2 Localização e segmentação

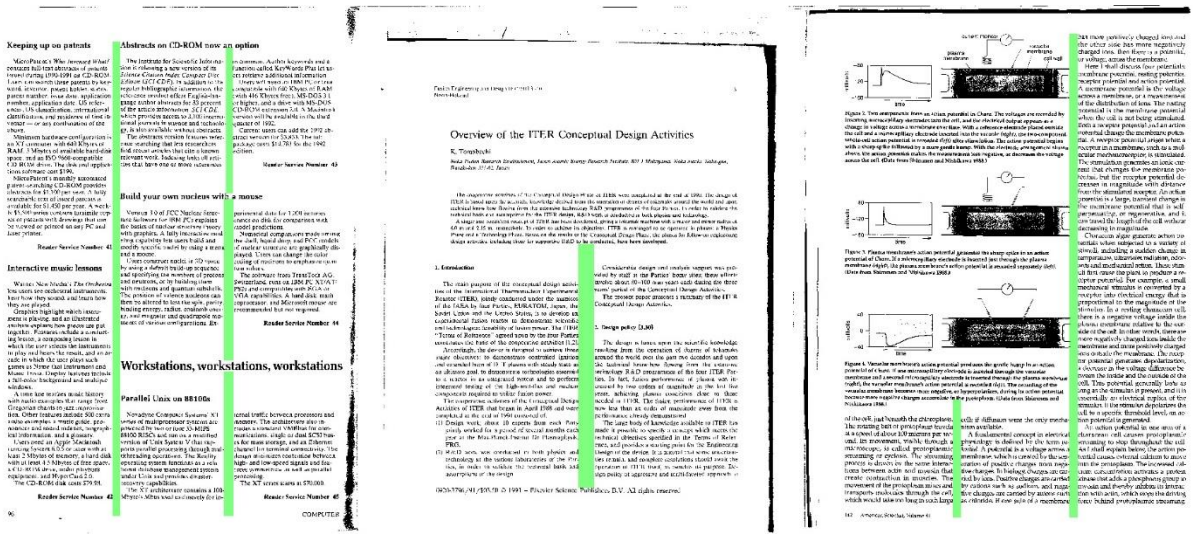
A localização e segmentação é uma das etapas presentes em praticamente todos os sistemas de processamento de imagens. Esse componente tem o objetivo de localizar e segmentar, ou seja, delimitar as imagens, parágrafos, linhas de texto, caracteres, palavras, blocos de texto e outros itens estruturais que compõem o documento digitalizado, de acordo com a necessidade do sistema. O funcionamento e desempenho do sistema como um todo depende de uma correta localização e segmentação. Tratando-se de um sistema de reconhecimento ótico de caracteres (OCR), a localização e segmentação ocorre em componentes de texto de uma imagem (BREUEL, 2002).

Uma das primeiras fases na segmentação de um documento, é a detecção de componentes conexos. Entende-se como um CC (Componente Conexo) um conjunto de *pixels* dispostos de tal maneira que pode-se chegar a qualquer outro *pixel* presente no componente a partir de outro *pixel* presente nesse mesmo componente. Ou seja, um componente conexo possui todos os seus *pixels* conectados de alguma maneira (BREUEL, 2002).

Os dois métodos para a solução de problemas relativos a problemas de segmentação de documentos propostos por Breuel (2002), consistem em identificar espaços retangulares em branco e identificar linhas levando-se em consideração um conjunto de espaços em branco em uma página. Pode-se utilizar o primeiro método para detectar colunas em um documento, desde que este encontre um retângulo vazio, ou seja, sem texto, entre duas colunas de texto. Esse método identifica espaços em branco de acordo com a área de cada retângulo, isso é feito através de uma fila de prioridades que é dada de acordo com o tamanho de cada retângulo, geralmente feita em ordem crescente (Figura 5). A cada iteração desse método um retângulo é retirado da fila e dividido em quatro partes que são reinseridas na fila. Quando um retângulo vazio é encontrado ele é classificado como espaço em branco e inserido na lista de CCs, isso evita que outros retângulos que possuam partes em comum a este sejam processados. Essa iteração continua até que todos os retângulos vazios sejam encontrados. O segundo método é utilizado

para encontrar linhas de texto. A Figura 5 exibe alguns exemplos de resultados da avaliação de espaços em brancos para a detecção de limites de colunas. As linhas encontradas por esse método não podem exceder os limites gerados pela detecção de espaços vazios. Esses dois métodos unidos formam uma ferramenta ideal para a segmentação de documentos (BREUEL, 2002).

FIGURA 5 – Exemplos de detecção de limites de colunas



Fonte: (BREUEL, 2002).

Segundo Wang, Lu e Tan (2003) é possível segmentar palavras em uma imagem utilizando diagramas de Voronoi. Primeiramente, os CCs são detectados e os que representam algum tipo de ruído ou caracteres especiais são descartados. O diagrama de Voronoi forma bordas que separam dois CCs. Cada uma dessas bordas é avaliada segundo um conjunto de restrições que levam em conta a distância da borda aos CCs que ela separa, essa distância é relacionada com as distâncias das outras bordas que cercam esses componentes. Caso a borda se enquadrar nas restrições ela é removida, e isso resulta na união de dois CCs. Uma das vantagens desse método é que ele dispensa a detecção de linhas e segmenta as palavras diretamente. Por outro lado, as operações realizadas pelo cálculo do diagrama de Voronoi são complexas e isso resulta em um processamento relativamente custoso (WANG; LU; TAN, 2003).

3.1.3 Pré-processamento

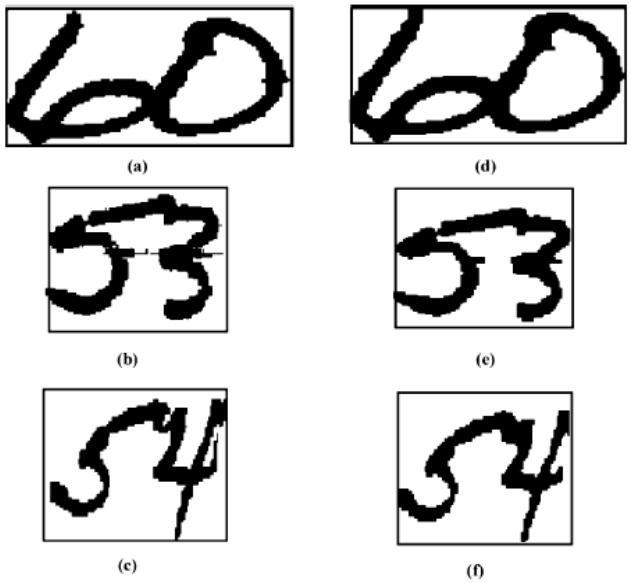
Dependendo do estado físico do documento digitalizado, da qualidade do equipamento de *scanner* utilizado na digitalização, ou ainda da técnica de *thresholding* escolhida, a imagem pode conter uma quantidade razoável de imperfeições que podem afetar o reconhecimento dos caracteres. A fim de diminuir, ou até eliminar essas imperfeições, a etapa de pré-processamento

é necessária (EIKVIL, 1993), sendo que Cheriet (2007) divide o pré-processamento nas seguintes etapas:

- Suavização e remoção de ruídos;
- Análise do grau de enviesamento e correção;
- Inclinação;
- Análise/definição de contorno;
- Afinamento.

A suavização dos níveis de cinza de uma imagem é utilizada para redução de ruído e *blurring*. O efeito de *blurring* é usado na etapa de pré-processamento para a remoção de pequenas imperfeições. Além disso, utiliza-se a suavização, geralmente, para reduzir o ruído, ou ainda, para diminuir a espessura da borda dos caracteres.

FIGURA 6 – Suavização e remoção de ruídos



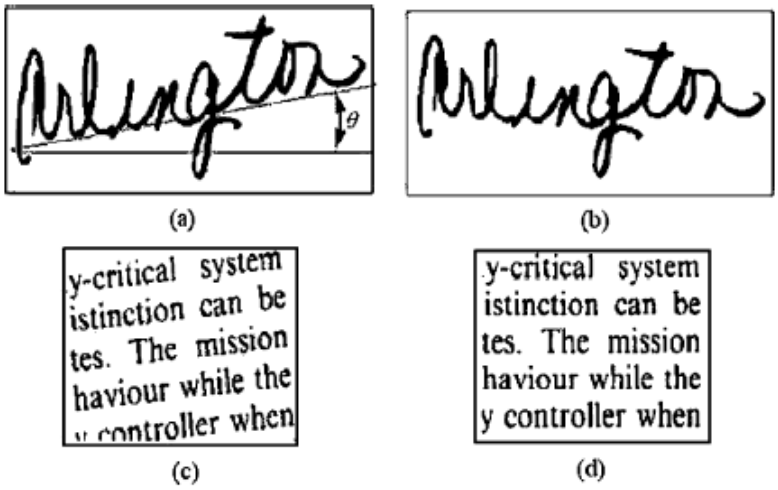
Fonte: (CHERIET, 2007)

Esse processo de suavização e remoção de ruídos pode ser feito utilizando técnicas de filtragem, sendo que essas técnicas de filtragem baseiam-se na análise dos *pixels* que estão nas proximidades (CHERIET, 2007). A Figura 6 retrata exemplos da aplicação dessa técnica, onde (a), (b) e (c) são as imagens originais; (d), (e) e (f) são respectivamente as imagens com essa técnica aplicada.

O grau de enviesamento é o valor de inclinação do texto escaneado em relação a uma linha horizontal. A Figura 7 retrata dois tipos de enviesamento. No primeiro exemplo tem-se uma palavra manuscrita enviesada (a) e, posteriormente, a palavra manuscrita corrigida (b). Já no segundo exemplo tem-se um trecho datilografado enviesado (c) e após, o mesmo trecho

datilografado corrigido (d). Isso pode ocorrer quando o documento, a ser escaneado, acaba ficando inclinando em relação ao sensor do equipamento de *scanner*. Nessa etapa são utilizados alguns métodos para determinar o grau de enviesamento, e posteriormente a imagem é ajustada, até que o texto apresente um grau de enviesamento igual, ou próximo a zero (CHERIET, 2007).

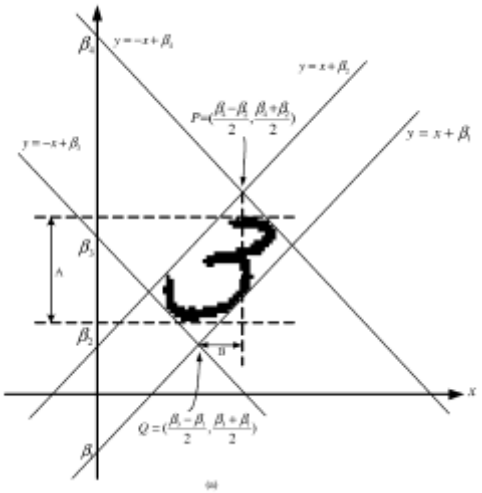
FIGURA 7 – Grau de enviesamento e correção



Fonte: (CHERIET, 2007)

Diferentemente do grau de enviesamento a inclinação de caracteres é encontrada principalmente em textos escritos à mão. O objetivo dessa etapa é melhorar a taxa de precisão do reconhecimento de palavras e numerais. Assim como o cálculo do grau de enviesamento, a análise da inclinação, leva em conta o grau de inclinação, mas dessa vez cada caractere é analisado individualmente (Figura 8) (CHERIET, 2007).

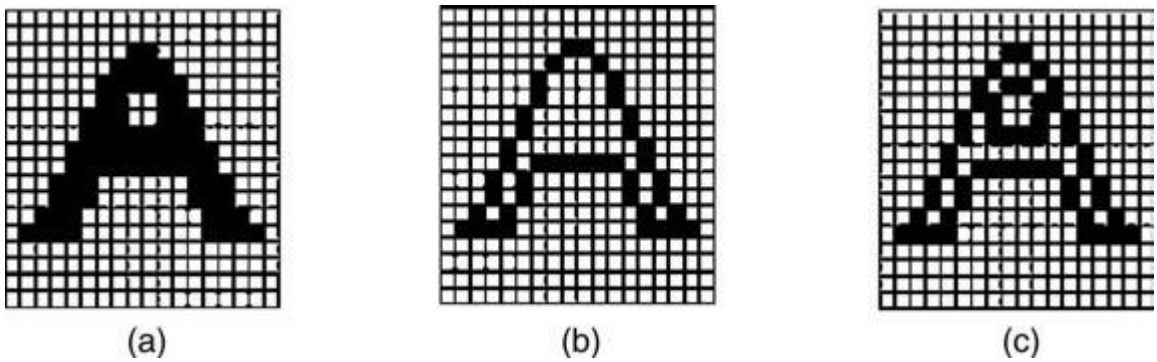
FIGURA 8 – Grau de inclinação do caractere



Fonte: (CHERIET, 2007)

A determinação de contorno é uma técnica aplicada a um caractere para extrair o seu contorno externo e interno. Para definir os *pixels* onde estão as bordas de cada caractere, é feita uma comparação entre o *pixel* central e seus vizinhos. Com a determinação da localização dos *pixels* de borda pode-se definir os contornos do caractere. Definido o contorno, suas diferentes características podem ser analisadas e, posteriormente, utilizadas na classificação (CHERIET, 2007). A Figura 9 retrata um exemplo dessa técnica aplicada. No primeiro exemplo tem-se o caractere original (a), já no segundo exemplo tem-se essa técnica aplicada, mas apenas para extração do contorno externo (b), e finalmente no último exemplo, tem-se essa técnica aplicada, tanto para contornos externos quando para contornos internos (c).

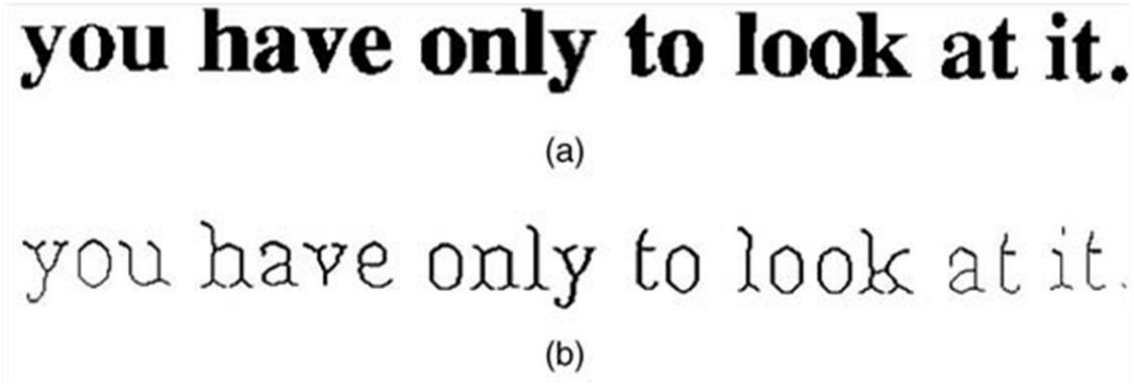
FIGURA 9 – Análise/definição de contorno



Fonte: (CHERIET, 2007)

O processo de afinamento consiste em remover o máximo possível de *pixels* dos caracteres sem afetar o aspecto geral do caractere. Ou seja, mesmo com a remoção de *pixels*, o caractere ainda é reconhecível. Esse processo concentra-se em tornar o caractere o mais fino possível, sendo que os *pixels* devem estar todos conectados e centralizados. A partir disso, algumas características importantes podem ser extraídas como, por exemplo, número de ramificações, intersecções e posição relativa (CHERIET, 2007). Na Figura 10 é possível ver o resultado (b) dessa técnica aplicada a um trecho de texto (a).

FIGURA 10 – Afinamento



Fonte: (CHERIET, 2007)

3.1.4 Extração de Características

Para Eikvil (1993) um dos grandes desafios de sistemas de OCR é extrair as características principais de cada caractere. Eikvil (1993) considera essa, uma das etapas mais difíceis realizadas pelo sistema de OCR, pois baseia-se no reconhecimento de padrões. A extração de características é uma das formas mais utilizadas para o reconhecimento de caracteres, é através dessas características que o reconhecimento dos caracteres torna-se possível. As técnicas para obtenção dessas características podem ser divididas em três grupos (EIKVIL, 1993), que são:

- Transformações e expansões em séries: essas técnicas ajudam a reduzir as dimensões do vetor de características, por se basearem somente nas curvas externas dos caracteres. Essas técnicas possuem tolerância satisfatória quando se trata de rotação e translação, no entanto possui pouca tolerância a imperfeições externas;
- Distribuição dos pixels: essa categoria abrange as técnicas que extraiam características baseadas na distribuição estatística dos *pixels*. As características obtidas são geralmente tolerantes a distorções e variações de estilo;
- Análise estrutural: as características obtidas descrevem de forma geométrica e topológica os caracteres. Com essas informações pode-se descrever a parte física do caractere como, por exemplo, ranhuras, voltas, pontos de término e intersecções de linhas. Comparada às outras técnicas, a análise estrutural fornece características com alta tolerância a ruídos e variações de estilo. No entanto, não são tão satisfatórias quando se trata de rotação e translação.

3.1.5 Classificação

Cheriet (2007) afirma que a obtenção das classes de características que compõem os padrões dos caracteres é o objetivo final do reconhecimento de caracteres. O trabalho de reconhecimento consiste em ligar um conjunto pré-definido de classes a cada padrão de caracteres ou de palavras, que foram segmentados a partir de imagens de documentos. Nessa etapa então, ocorre a classificação dos segmentos extraídos, de acordo com as suas classes de características (CHERIET, 2007).

A classificação de padrões tem sido o principal tema abordado dentro da área de reconhecimento de padrões. Muitos métodos eficientes têm se originado a partir da intensa fundamentação teórica feita com relação ao reconhecimento estatístico de padrões, esses

métodos utilizam, geralmente, algoritmos baseados na teoria Bayesiana, ou ainda algoritmos baseados em redes neurais (CHERIET, 2007).

As redes neurais vêm sendo utilizadas desde o final dos anos 1980 para o reconhecimento de padrões devido ao surgimento de algoritmos compostos por redes de retro propagação. Estas redes são compostas por várias camadas de elementos interconectados. Um vetor de características entra na rede pela camada de entrada, onde cada elemento da camada calcula uma soma ponderada de sua entrada e a transforma em uma saída. Durante o treinamento, os pesos de cada conexão são ajustados até que a saída desejada seja obtida. Um problema de redes neurais em OCR pode ser a sua previsibilidade e generalidade limitada, enquanto que uma vantagem é a sua natureza adaptativa (CHERIET, 2007).

3.1.6 Pós-processamento

O simples reconhecimento de cada símbolo em um documento resulta em um conjunto individual de símbolos. Mas esses símbolos, geralmente, não representam informações isoladas. Por isso, é desejável que se associe cada símbolo, com aqueles que pertencem a mesma cadeia, criando assim, palavras e números. Esse processo de associação pode ser chamado de agrupamento, e baseia-se basicamente, na posição de cada símbolo dentro do documento. Ou seja, são agrupados os símbolos que estão próximos o suficiente uns dos outros.

A etapa de agrupamento pode apresentar alguns problemas com fontes que não tenham um tamanho padrão, uma vez que é mais difícil determinar o espaço que cada caractere ocupa. Isso pode ocorrer principalmente para documentos escritos à mão ou que possuem o texto inclinado. Geralmente, para caracteres que possuem distância variável entre eles, o agrupamento é possível, pois a distância entre uma palavra e outra é relativamente maior.

Até os sistemas de OCR mais modernos não possuem cem por cento de precisão na identificação de caracteres, mas com o uso do contexto, alguns erros podem ser detectados e até mesmo corrigidos. Por exemplo, existem mecanismos que utilizam regras de sintaxe ou ainda, se a palavra possui erros ortográficos. Um exemplo disso consiste, em que uma frase deve começar com letra maiúscula após o ponto final. Ou ainda, a língua portuguesa não possui palavras que iniciem com a letra “ç” ou que contenham três letras “r” juntas, se isso ocorrer, houve alguma falha no reconhecimento.

Uma segunda abordagem para a detecção de erros é o uso de dicionários. Essa é considerada a melhor forma de detectar e corrigir erros. Toda a palavra que possui alguma inconformidade é verificada no dicionário. Caso a palavra não exista no dicionário, ela é

alterada para uma palavra similar, mas essa abordagem também não está livre de erros. Ou seja, o OCR pode ter transformado uma palavra em outra que também exista no dicionário e, dessa forma, o erro não é detectado. Além disso, o uso de dicionários na detecção de falhas ocupa um grande tempo e esforço de processamento, sendo essa a sua principal desvantagem.

Por fim, outra abordagem, que está proposta nesse trabalho, é a utilização de uma ferramenta de *crowdsourcing*, para fazer a identificação das palavras e/ou caracteres não reconhecidos pelo OCR. De fato, fica claro que o componente ideal para a implantação dessa ferramenta, é o pós-processamento. É no pós-processamento que os caracteres são agrupados, as palavras formadas e reconhecidas. Desta forma, o uso de *crowdsourcing* é uma alternativa para garantir uma digitalização livre de erros.

3.2 BIBLIOTECAS OCR

Há uma série de soluções de OCR disponíveis para converter imagem em texto. O *site www.toptenreviews.com* lista os nove melhores *softwares* comerciais de OCR disponíveis atualmente[1]. Os *softwares* OCR foram avaliadas seguindo os seguintes critérios:

- Desempenho: em um *software* OCR, o desempenho é fundamental. Um programa pode ter uma série de características, mas deve converter imagens em texto, pesquisável, preciso, e em um formato que seja útil;
- Conjunto de recursos: as características que cada *software* de OCR fornece devem ser analisadas de acordo com as necessidades de cada projeto. Quanto mais recursos um *software* possuir, maiores são as chances de ele se adequar ao projeto;
- Facilidade de uso: os *softwares* de OCR foram desenvolvidos para acelerar o processo de digitalização, isso significa que o *software* tem que levar menos tempo e ser mais fácil de usar do que digitar o documento digitalizado manualmente;
- Ajuda e suporte: um dos grandes benefícios de *softwares* comerciais é o suporte ao usuário. Isso vai desde dúvidas, tutoriais e até suporte técnico.

O *site http://www.linux-mag.com* lista as opções *open source* de bibliotecas OCR disponíveis para o sistema operacional Linux[2]. Apesar da extensa lista de *softwares* OCR *open source* disponíveis, este *site* destaca as seguintes ferramentas:

- Tesseract;
- OCROpus;

[1] http://ocr-software-review.toptenreviews.com/
[2] http://www.linux-mag.com/id/5320/

- GNU Ocrad;
- GOCR.

Para fazer um comparativo entre esses *softwares open source*, realizou-se uma extensa pesquisa em diversos artigos, revistas, trabalhos acadêmicos e periódicos *on-line*, no entanto a comparação mais satisfatória encontrada foi feita por Peter Selinger em 2007. Selinger (2007) fez uma comparação entre os softwares Tesseract, Ocrad e GOCR.

Essa comparação foi feita utilizando dois documentos digitalizados, sendo que, cada um originou uma imagem que posteriormente sofreu a aplicação de dois tipos diferentes de conversão em tons de cinza (*thresholding*), resultando em duas imagens bitonais[3]. A intenção da aplicação do método de *thresholding*, foi produzir uma imagem mais clara, possibilitando um melhor reconhecimento dos caracteres.

Os testes foram realizados utilizando basicamente três critérios: percentual de palavras não reconhecidas ou traduzidas incorretamente, percentual de caracteres errados e o tempo de execução que cada *software* levou para concluir a digitalização. A Tabela 2 lista o percentual de palavras não reconhecidas ou traduzidas incorretamente em cada um dos *software*s avaliados.

TABELA 2 – Percentual de palavras erradas

Imagens	Ocrad	GOCR	Tesseract
Imagem 1 (original)	20.00%	29.84%	93.33%
Imagem 1 (naive bitonal)	22.85%	28.25%	6.66%
Imagem 1 (custom bitonal)	26.98%	26.98%	6.34%
Imagem 2 (original)	37.97%	55.27%	89.72%
Imagem 2 (naive bitonal)	30.00%	49.45%	0.67%
Imagem 2 (custom bitonal)	22.70%	47.29%	0.40%

Fonte: Adaptada de SELINGER, 2007.

Na Tabela 3 é possível visualizar o percentual de caracteres que não foram reconhecidos, ou foram traduzidos incorretamente, pelos *softwares* de OCR.

TABELA 3 – Percentual de caracteres errados

Imagens	Ocrad	GOCR	Tesseract
Imagem 1 (original)	5.32%	6.96%	75.14%
Imagem 1 (naive bitonal)	5.49%	6.73%	1.75%
Imagem 1 (custom bitonal)	5.71%	8.15%	1.47%
Imagem 2 (original)	12.57%	26.62%	73.57%
Imagem 2 (naive bitonal)	8.37%	22.58%	0.22%
Imagem 2 (custom bitonal)	5.64%	19.99%	0.14%

Fonte: Adaptada de SELINGER, 2007.

[3] Imagens bitonais são imagens possuem apenas uma cor de primeiro plano e uma cor de fundo.

Por fim na Tabela 4 é possível observar o tempo de execução, em segundos, que cada *software* levou para concluir a digitalização.

TABELA 4 – Tempo de execução

Imagens	Ocrad	GOCR	Tesseract
Imagem 1 (original)	17.36	66.78	21.00
Imagem 1 (naive bitonal)	4.76	46.68	18.11
Imagem 1 (custom bitonal)	4.14	61.80	14.37
Imagem 2 (original)	15.69	149.73	44.15
Imagem 2 (naive bitonal)	5.67	126.35	27.34
Imagem 2 (custom bitonal)	4.77	224.52	25.42

Fonte: Adaptada de SELINGER, 2007.

Selinger (2007) chegou à conclusão de que em termos de precisão, nos testes em que foram utilizadas imagens bitonais, o Tesseract supera, com uma larga diferença, tanto o Ocrad quanto o GOCR. No entanto, para imagens não bitonais, o uso do Tesseract se mostrou inadequado. O Ocrad e o GOCR tiveram resultados semelhantes, incluindo a dificuldade para reconhecer caracteres em itálico. Já em termos de tempo de execução Peter identificou que o Ocrad obteve o melhor desempenho, seguido pelo Tesseract, e por último o GOCR.

Selinger (2007) ressalta que os resultados dos testes são relativos, e podem mudar de acordo com o documento digitalizado. Por fim ele conclui que o Tesseract se saiu melhor nos testes e foi eleito o vencedor, onde a única ressalva é que deve-se utilizar imagens bitonais para um resultado satisfatório. O Ocrad pode ser útil para aplicações onde a velocidade é mais importante que o resultado. Já os testes realizados com o GOCR, resultaram em uma precisão pobre, além de um alto tempo de execução, tornado o uso desse software inviável se comparado aos outros dois.

A fim de comprovar a viabilidade e a funcionalidade dos *softwares* OCR listados, foram realizados alguns testes apresentados nas próximas seções. Todos os testes foram realizados utilizando os mesmos fragmentos de um documento a fim de comparar qual o *software* mais adequado para a aplicação proposta nesse trabalho.

3.2.1 Testes realizados no software Tesseract-OCR

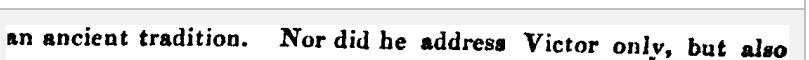
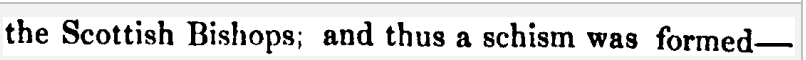
O Tesseract é uma biblioteca OCR codificada em linguagem de programação *C/C++* e desenvolvido pela HP (*Hewlett-Packard*) entre 1984 e 1994. Esse foi desenvolvido com um projeto de pesquisa nos laboratórios da HP e ganhou importância como um possível *software* ou *hardware* adicional para a linha de *scanners* de mesa da HP. Sua principal motivação era dada pelo fato de que na época os sistemas de OCR ainda eram imprecisos e falhavam quando

o documento de origem não apresentava uma boa qualidade de impressão. Apesar de apresentar uma significante liderança na precisão com relação a outros produtos comerciais, o Tesseract não tornou-se um produto, sendo que em 1994, o projeto foi cancelado. Em 2005, a HP liberou Tesseract como um *software open source* e desde então esse projeto vem sendo mantido pela Google (SMITH, 2007).

Atualmente o Tesseract-OCR está licenciado através da *Apache Licence 2.0* e está disponível para sistemas operacionais *Windows*, *Linux* e *MacOS X*. A biblioteca Tesseract e sua documentação estão disponíveis através do endereço http://code.google.com/p/tesseract-ocr/. O idioma padrão do Tesseract-OCR é o inglês, mas ele também conta com suporte a diversas línguas, bastando baixar e instalar o pacote com a linguagem desejada. Isso possibilita a utilização de dicionários para auxiliar no reconhecimento das palavras, correspondentes a cada idioma. Não foi encontrada nenhuma informação sobre o uso da gramatica pelo Tesseract, para fazer o reconhecimento das palavras.

A Tabela 5 lista os testes realizados com o intuito de demonstrar a funcionalidade do sistema apresentado.

TABELA 5 – Testes *software* Tesseract

Palavras	**Certas**	**Erradas**	**Imagem/Resultado**
11	11	0	
			an ancient tradition. Nor did he address Victor only, but also
13	12	1	
			the rest of the Bishop: on the same side, as Victor himself had
9	9	0	
			the Scottish Bishops; and thus a schism was formed—
9	9	0	
			the same doctrines being professed, and the same forms
9	9	0	
			used in the Scotch Episcopal Chapels, and in those

(Continua)

(Conclusão)

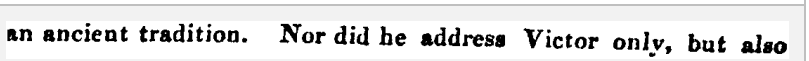
Palavras	Certas	Erradas	Imagem/Resultado
8	7	1	
			legally qualified chapels which were under no episcopal

3.2.2 Testes realizados no software OCROpus

O OCROpus é uma biblioteca OCR codificada em linguagem de programação *Python*, e que utiliza duas bibliotecas auxiliares que são o *NumPy* e *SciPy*. Esse foi desenvolvido a partir de dois projetos de pesquisa, sendo o primeiro deles um sistema de reconhecimento de escrita à mão de alto desempenho, e o segundo uma ferramenta de alto desempenho de métodos de análise. O projeto atualmente é patrocinado pela Google e chefiado por Thomas Breuel do Centro de Pesquisa Alemão para Inteligência Artificial em Kaisersleutern na Alemanha. Atualmente o OCROpus está licenciado através da *Apache Licence 2.0* e está disponível para o sistema operacional *Linux*. A biblioteca OCRopus, e a documentação correspondente, estão disponíveis através do endereço https://code.google.com/p/ocropus (OCROPUS, 2012).

A Tabela 6 lista os testes realizados com o intuito de demonstrar a funcionalidade do sistema apresentado.1

TABELA 6 – Testes *software* OCROpus

Palavras	Certas	Erradas	Imagem/Resultado
11	7	4	
			an &ncient tradition. Nor did ne addres8 ViEor only, but also
13	9	4	
			the rest of tbe Bishop: on tbe same side, as ViEor himself had
9	6	3	
			the Scottish Bis}nops;; aod thus a 8chism was formed—
9	9	0	
			the same doctrines being professed, and the same forms

(Continua)

(Conclusão)

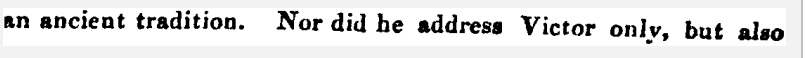
Palavras	Certas	Erradas	Imagem/Resultado
9	8	1	
			used in the Scotch Episcopal Chapels, and in fhose
8	5	3	
			legally qualihed chapels which were onder no episcopal

3.2.3 Testes realizados no software GOCR

O GOCR (ou JOCR) é uma biblioteca OCR codificada em linguagem de programação *C++* que surgiu no final dos anos 1990. O nome GOCR significa *GNU Optical Character Recognition*. GOCR é licenciado através da *GNU General Public License* (GPL) e está disponível para os sistemas operacionais *Windows, OS/2* e *Linux*. A biblioteca GOCR e sua documentação estão disponíveis através do endereço http://sourceforge.net/projects/jocr/ (GOCR, 2012).

A Tabela 7 lista os testes realizados com o intuito de demonstrar a funcionalidade do sistema apresentado.

TABELA 7 – Testes *software* GOCR

Palavras	Certas	Erradas	Imagem/Resultado
11	2	9	
			n &ncieat tr8ditioa, Nor di be _ddre8g YicEor onl_v,ii_yt__o
13	0	13	
			tbe r__t of tbe _i8_op_ o0 tbe ___e 8id_, _ _i_tor _i__eIf b8d
9	4	5	
			the Scottish Bis}_o_s; 8Dd thus _ 8chism _8_ Formed-
9	1	8	
			the s_me d_ctrine& b_in_ pro Fesscd, _nd the __me Fo_g

(Continua)

(Conclusão)

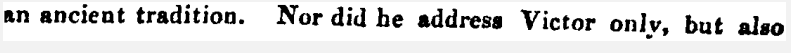
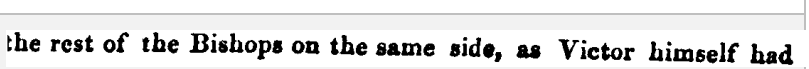
Palavras	Certas	Erradas	Imagem/Resultado
9	5	4	
			used in the Scotch F__isro__l Ch_pel9, 8nd jn fhoge
8	4	4	
			le,oaIly quali_ed chapela which _ere under no eFjscopgl

3.2.4 Testes realizados no software GNU Ocrad

O GNU Ocrad é uma biblioteca OCR, que faz parte do Projeto GNU, e também está licenciada pela GNU GPL. Esse projeto teve início em 2003, e foi desenvolvido em linguagem de programação *C++*. Poucas informações estão disponíveis sobre o Ocrad em sua página oficial, mas sabe-se que ele se baseia no método de extração de características para o reconhecimento de caracteres. Essa biblioteca também possui um analisador de *layout*, que é capaz de separar os blocos e colunas de texto encontrados em diferentes documentos impressos. A biblioteca GNU Ocrad e sua documentação estão disponíveis através do endereço http://www.gnu.org/software/ocrad/ocrad.html.

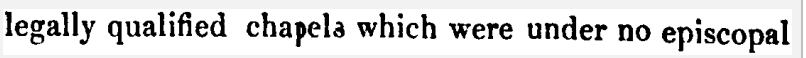
A Tabela 8 lista os testes realizados com o intuito de demonstrar a funcionalidade do sistema apresentado.

TABELA 8 – Testes *software* Ocrad

Palavras	Certas	Erradas	Imagem/Resultado
11	2	9	
			n &ncieat tr8ditioa, Nor di be _ddre8g YicEor onl_v,ii_yt__o
13	0	13	
			tbe r__t of tbe _i8_op_ o0 tbe ___e 8id_, _ _i_tor _i__eIf b8d
9	4	5	
			the Scottish Bis}_o_s; 8Dd thus _ 8chism _8_ Formed-

(Continua)

(Conclusão)

Palavras	Certas	Erradas	Imagem/Resultado
9	2	7	
			the s_me d_ctrine& b_in_ pro Fesscd, _nd the __me Fo_g
9	4	5	
			used in the Scotch F__isro__l Ch_pel9, 8nd jn fhoge
8	4	4	
			le,oaIly quali_ed chapela which _ere under no eFjscopgl

3.3 CONSIDERAÇÕES FINAIS

Nesse capitulo foram apresentados os principais componentes de um *software* de reconhecimento ótico de caracteres, bem como, o funcionamento de cada um. Com isso pode-se entender melhor cada etapa do processo de digitalização e identificar onde ocorrem as falhas de reconhecimento dos caracteres e por consequência, a formação de palavras incorretas. Também foram apresentados alguns *softwares* de OCR *open source*, os quais foram submetidos a testes e comparativos para avaliar qual é o mais adequado para a proposta do presente trabalho.

Após a realização de vários testes, utilizando os mesmos fragmentos de um documento, é possível visualizar na Tabela 9 os resultados obtidos. Com isso percebe-se que o *software* Tesseract apresentou melhores resultados, em relação aos outros *softwares*. Conforme os testes realizados por Selinger (2007), os *softwares* GOCR e Ocrad ficaram com o desempenho abaixo do esperado, e foram descartados. Já o *software* OCROpus apresentou resultados satisfatórios, no entanto houveram algumas dificuldades para a realização dos testes devido a documentação ser insuficiente e pouco clara.

TABELA 9 – Comparativo dos testes realizados entre *softwares* OCRs

Softwares	Tesseract	OCROpus	GOCR	Ocrad
Acertos	96,6%	74,6%	27,1%	27,1%
Erros	3,4%	25,4%	72,9%	72,9%

No próximo capítulo é apresentada a proposta de solução para a resolução do problema destacado no objetivo deste trabalho. O estudo da proposta de solução apresenta a arquitetura a ser utilizada, os seus principais componentes, as tecnologias utilizadas, e modelagem de requisitos e classes, para o desenvolvimento da aplicação proposta.

4 PROPOSTA DE SOLUÇÃO

Conforme destacado no objetivo desse trabalho, o propósito foi desenvolver uma aplicação que dê suporte a digitalização de acervos. Para que isso seja possível, essa aplicação deve contar com o auxílio de uma biblioteca OCR, e também oferecer suporte a uma API CAPTCHA, para que *websites* externos possam utilizá-la como ferramenta de segurança. Além de fornecer segurança aos *websites*, a API CAPTCHA é responsável por auxiliar na digitalização das palavras não reconhecidas pelo OCR. Nessa etapa foram aplicados os conceitos de *crowdsourcing*, para que as palavras sejam distribuídas em larga escala pela internet e a multidão de usuários possa colaborar com a digitalização, reconhecendo essas palavras.

De acordo com as informações obtidas nos trabalhos relacionados, o uso de *crowdsourcing* é capaz de solucionar diversos problemas associados a falta de tempo, de mão-de-obra e de qualidade. Com isso é possível comprovar a viabilidade da aplicação do *crowdsourcing* para a solução do problema apresentado nesse trabalho.

Conforme abordado nas seções anteriores, o reCAPTCHA é uma ferramenta de *crowdsourcing* que age de forma implícita, ou seja, o usuário não possui o consentimento de estar auxiliando na digitalização de um documento. O principal conceito envolvido na utilização de um CAPTCHA como ferramenta de *crowdsourcing*, é aproveitar, de forma útil, o tempo que o usuário leva para fazer a sua autenticação sem agregar algum custo de tempo adicional.

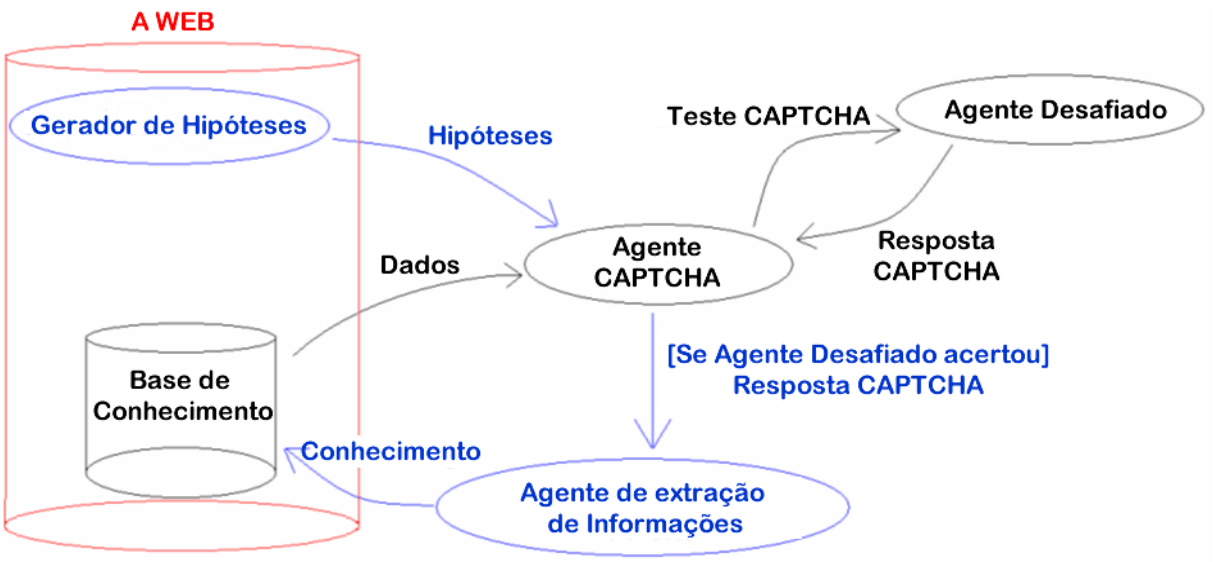
4.1 EXEMPLOS DE ARQUITETURAS

Silva (2007) elaborou uma arquitetura (Figura 11) com a finalidade de criar uma base de conhecimento utilizando um mecanismo de CAPTCHA e o poder da colaboração *on-line*. O modelo, denominado KA-CAPTCHA, é um mecanismo que é capaz de extrair conhecimento a partir de usuários da *web* durante a sua interação com um CAPTCHA (SILVA, 2007).

Em um mecanismo CAPTCHA, um agente gera um teste de recuperação de dados a partir de uma base de conhecimento pública. Nessa extensão, Silva (2007) tenta coletar novos conhecimentos dos usuários, combinando o que foi coletado a partir da base de conhecimento CAPTCHA, com os dados recolhidos a partir de outras fontes da *web*, representadas por um gerador de hipóteses. Os dados que são recuperados a partir do gerador de hipóteses representam o conteúdo da *web* que ainda é pouco comentada e cuja semântica correta do

CAPTCHA pode ser fornecida por um usuário legítimo que está resolvendo o teste (SILVA, 2007).

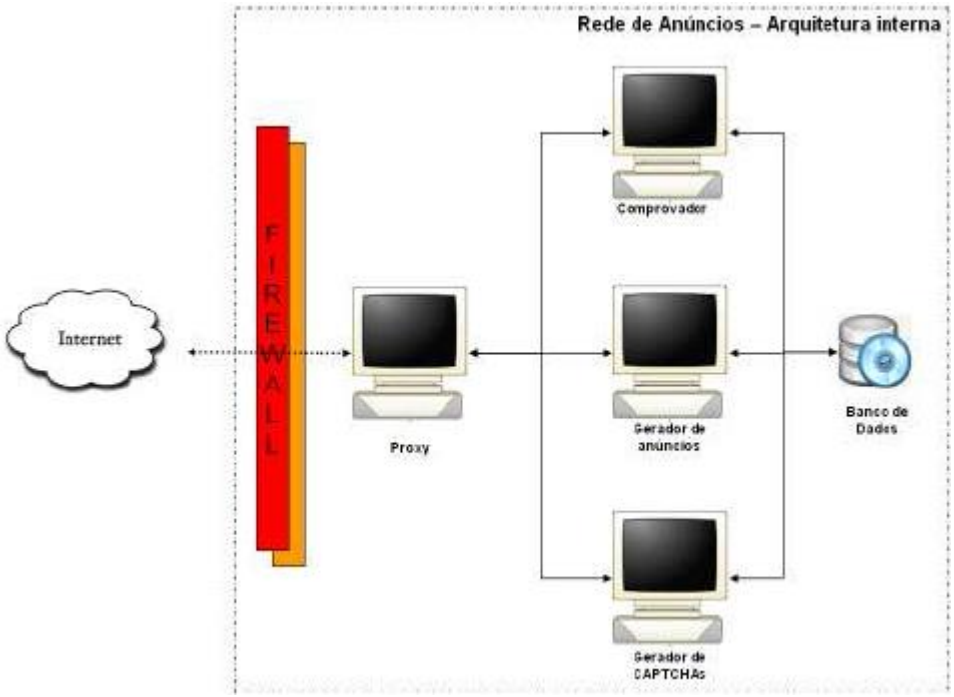
FIGURA 11 – Arquitetura KA-CAPTCHA



Fonte: Adaptado de SILVA, 2007.

Costa (2010) elaborou uma arquitetura com a finalidade de utilizar um mecanismo CAPTCHA para combater e prevenir fraudes em anúncios *on-line*. Além disso, ele propõe uma nova abordagem para combater o problema da *click fraud* em sistemas de pagamento por clique,

FIGURA 12 – Arquitetura proposta por Costa (2010)

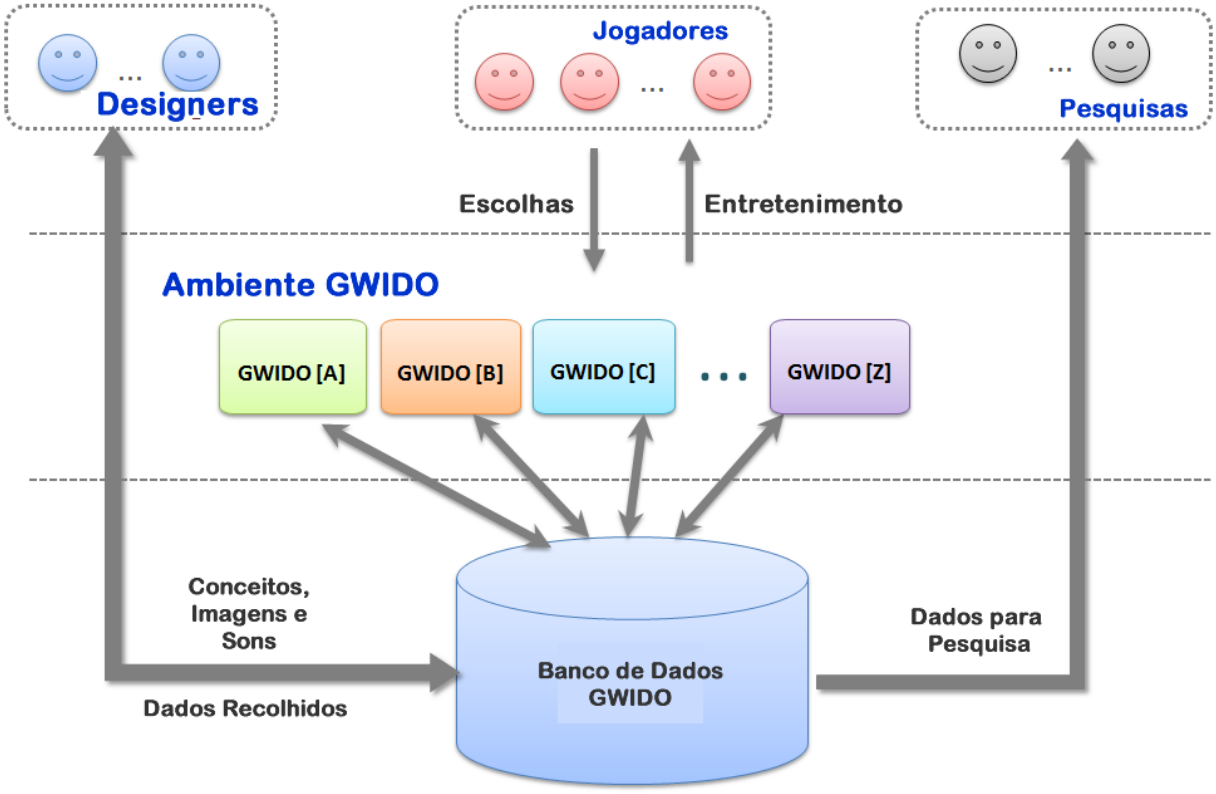


Fonte: (Costa, 2010).

utilizados em anúncios publicados em ferramentas de busca (COSTA, 2010). Conforme a arquitetura proposta por Costa (2010) (Figura 12), os anúncios são acessados pelo usuário e ao efetuar esse acesso, o usuário passa por um Comprovador, a fim de comprovar a sua identidade. Essa comprovação ocorre através de um *token* armazenado no navegador do usuário, caso esse *token* esteja expirado ou não existir, o usuário passará por um teste CAPTCHA e assim um novo *token* é gerado. Após passar por essa autenticação o usuário tem acesso ao anúncio.

Romani (2013), propõe uma abordagem GWIDO (*Games With Interaction Design Objective*), beneficiando-se de recursos oferecidos pela *web* contemporânea, bem como do interesse das pessoas por jogos na internet. A ideia central consiste no uso de GWAPs (*Games With a Purpose*) para apoiar um *designer* na escolha de elementos de *design*, envolvendo nesse processo um grande número de usuários. O GWIDO é uma forma de GWAP colaborativa e síncrona entre dois jogadores. O ambiente GWIDO é uma aplicação *web* que pode conter diversos jogos com propósitos diferentes, esses jogos geralmente alimentam o mesmo banco de dados (Figura 13). O *designer* utiliza uma interface para cadastrar imagens, sons e conceitos de interface, esses elementos então, são usados nos jogos GWIDO. Depois de algumas rodadas do

FIGURA 13 – Arquitetura do Ambiente GWIDO



Fonte: Adaptado de Romani, 2013.

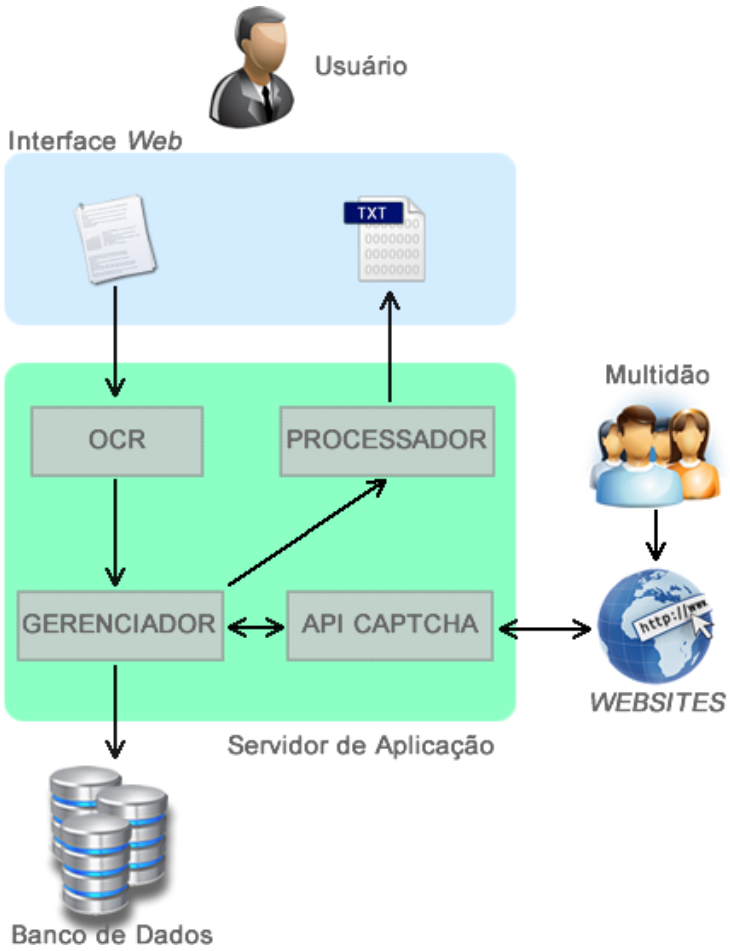
jogo, o *designer* pode coletar os resultados verificando as imagens mais representativas para diferentes perfis de usuário no ambiente (ROMANI, 2013).

Nesse modelo um pesquisador pode fazer uma análise estatística para verificar, por exemplo, se há diferenças regionais ou significativas entre os diferentes perfis de usuário, permitindo uma escolha mais bem informada dos elementos de interface de usuário para o sistema. Os dados recolhidos durante a execução dos jogos, estão associados aos perfis dos usuários para fornecer informações precisas, que deverão ajudar os *designers* a fazer escolhas. Por exemplo, um jogo GWIDO apresenta entradas de texto e imagens, fornecidas por um *designer*, o jogo instrui os jogadores a selecionar a imagem que melhor representa o texto. No caso de ambos os jogadores selecionarem a mesma imagem, ambos pontuam. Todas as escolhas são registradas pelo jogo, e estão disponíveis no ambiente para apoiar o *designer* em seu processo de decisão sobre qual imagem usar em sua aplicação (ROMANI, 2013).

4.2 ARQUITETURA PROPOSTA

Visando promover uma solução para uma digitalização de acervos livre de falhas, para usuários comuns da internet, foram levantados requisitos de forma a compreender as

FIGURA 14 – Principais componentes da arquitetura proposta



necessidades a serem atendidas pela solução. A fim de facilitar o entendimento da solução, bem como abstrair o problema a ser solucionado, a arquitetura aqui proposta encontra-se disposta na Figura 14. A arquitetura está baseada nos exemplos estudados anteriormente e apresenta uma visão superficial dos componentes que fazem parte da aplicação, e as ligações entre os mesmos.

Para um melhor entendimento da arquitetura proposta são apresentados a seguir os principais componentes da aplicação:

- Interface *web*: a interface *web*, disponibilizada pelo servidor de aplicação, conta com uma interface para que o usuário possa interagir e enviar os seus arquivos para serem digitalizados;
- Servidor de aplicação: o servidor de aplicação fornece todos os recursos necessários para o funcionamento da aplicação, ou seja, disponibilizar o *upload* de arquivos no formato de imagens e *download* de arquivos processados no formato texto. Esse componente também interage com a biblioteca Tesseract-OCR para o processamento dos arquivos enviados pelo usuário. Além disso, este também faz a comunicação com o banco de dados, e disponibiliza a API CAPTCHA para acesso externo;
- Sistema OCR: conforme estudado nas seções anteriores o *software* OCR a ser utilizada na aplicação proposta é o Tesseract-OCR. Este por sua vez recebe requisições do servidor de aplicação e retorna os trechos de texto reconhecidos, e também os trechos de texto não reconhecidos. A fim de facilitar a interação entre o *software* Tesseract-OCR e a aplicação, é utilizada uma interface denominada php-tesseract[1], codificada em linguagem de programação PHP. O uso dessa interface permite a utilização de vários formatos de arquivos de imagem (JPG, GIF, PNG, TIFF e etc), o uso dela também evita a utilização de arquivos de texto temporários durante processamento o OCR;
- Gerenciador: o gerenciador é responsável pela consulta e persistência no banco de dados;
- Banco de Dados: o banco de dados deve conter as tabelas onde são armazenados os dados do usuário, os trechos de texto reconhecidos pelo aplicativo OCR, e os trechos de texto não reconhecidos, que posteriormente são utilizados pela API CAPTCHA;

[1] https://code.google.com/p/php-tesseract/

- API CAPTCHA: a API CAPTCHA fornece o serviço de CAPTCHA como ferramenta de segurança para *websites* espalhados pela Internet. Essa API gera um CAPTCHA para autenticação do usuário, e também busca na base de dados uma palavra não reconhecida pelo OCR, para que o usuário faça o reconhecimento. A mesma palavra é enviada para inúmeros outros usuários para que ocorra um reconhecimento preciso;
- Processador: o processador é responsável pela geração do arquivo final em formato de texto editável;
- *Websites*: por fim, o último componente do sistema são os *websites*, ou qualquer outro dispositivo que deseja utilizar a API CAPTCHA como ferramenta de segurança. O papel desse componente é fazer o uso do paradigma de *crowdsourcing* para auxiliar na digitalização de acervos.

4.3 TECNOLOGIAS UTILIZADAS

Com o intuito de disponibilizar o serviço de digitalização de acervos para qualquer usuário da Internet, bem como disponibilizar a API CAPTCHA como ferramenta de segurança, optou-se por desenvolver uma aplicação *web*, cuja a linguagem de programação será PHP e o banco de dados MySQL.

Por ser um padrão largamente utilizado em aplicações *web*, o padrão MVC (*Model-view-controller*) foi a metodologia escolhida para o desenvolvimento da aplicação. Esse padrão também foi escolhido por favorecer a integração com outros sistemas através de APIs, permitindo que outras aplicações conectem-se e façam requisições e envio de dados. As três camadas que compõem o MVC são descritas a seguir:

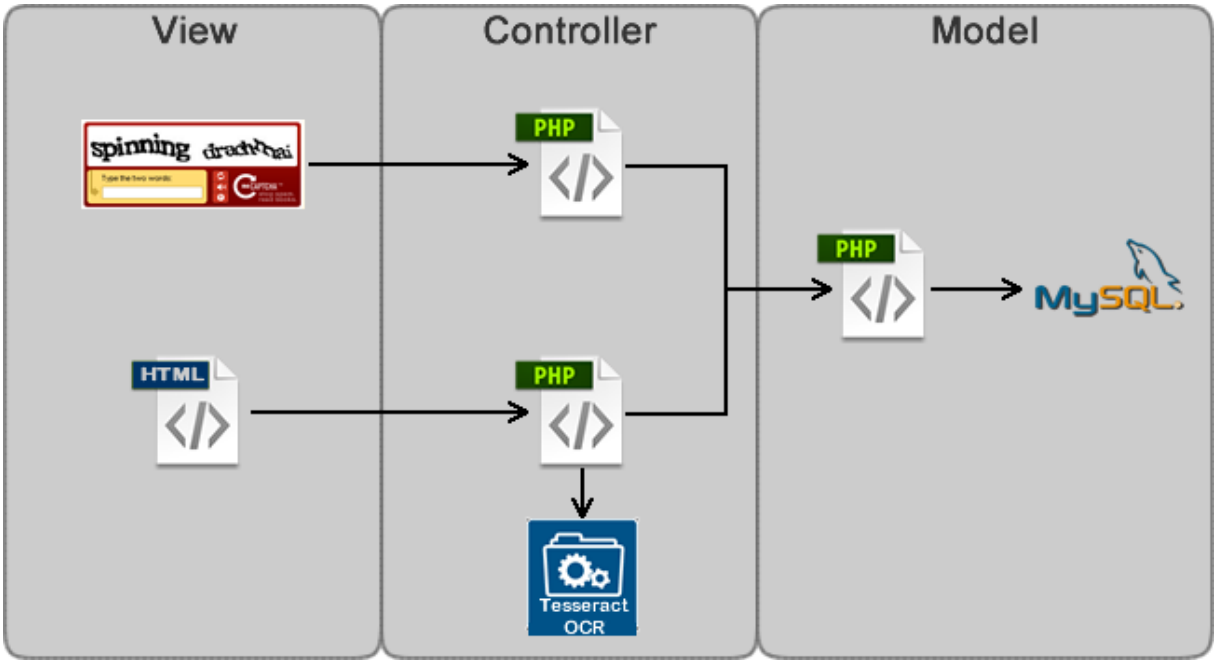
- Camada de modelo de dados (*Model*): essa camada é responsável pela interação com o banco de dados, além de representar a estrutura dos dados;
- Camada de aplicação (*View*): essa camada corresponde à interface com o usuário, é através dela que o usuário interage com o sistema;
- Camada de domínio ou negócio (*Controller*): essa camada é responsável por interligar as camadas de aplicação com a camada de modelo de dados. A função dessa camada é aplicar as regras de negócio da aplicação.

O modelo MVC da aplicação proposta pode ser visualizado na Figura 15, onde constam os principais componentes envolvidos no funcionamento da aplicação. A camada de visualização contém duas interfaces principais, a primeira interface é responsável pelo *upload* dos arquivos a serem digitalizados, e o *download* dos arquivos em formato de texto editável. A

segunda interface é disponibilizada para que os *websites* utilizem-na como ferramenta de segurança, essa interface faz a comunicação com a API CAPTCHA para a autenticação dos usuários.

A camada de controle também contém dois componentes principais, um deles é responsável pelo gerenciamento e digitalização dos arquivos postados pelo usuário. Esse componente interage com a biblioteca Tesseract-OCR através da interface php-tesseract. O outro componente é responsável pelo fornecimento da API CAPTCHA. Por fim, a camada de modelo faz a comunicação com a base de dados e interage com a camada de controle.

FIGURA 15 – Modelo MVC da aplicação proposta



Para o desenvolvimento da API CAPTCHA foi utilizada a arquitetura REST (*Representational State Transfer*). A arquitetura REST vem sendo amplamente utilizada para a troca de mensagens através de serviços *web*, esta garante que a troca de dados entre cliente e servidor seja feita de forma simples e dinâmica, sem acoplamento entre as partes. Sistemas REST costumam seguir uma metodologia parecida com MVC, ou seja, onde as camadas estão separadas tornando-as menos complexas. O REST também possibilita a troca de mensagens utilizando um formato extremamente leve para o intercâmbio de dados denominado JSON (*JavaScript Object Notation*), além de ser baseado na notação de objetos *JavaScript*, o que facilita a sua implementação em interfaces *web* (SCHMITZ, 2012).

4.4 MODELAGEM DO PROJETO

A fim de facilitar o desenvolvimento da aplicação proposta, a modelagem se mostra uma etapa essencial do projeto. A modelagem permite uma visão simplificada da realidade, e os modelos geralmente apresentam uma visão panorâmica da aplicação. Geralmente um modelo inclui os principais componentes de um sistema, omitindo os de menor importância de acordo com o nível de abstração. Para a modelagem da proposta de solução apresentada nesse trabalho, optou-se pela modelagem de requisitos funcionais, modelagem das principais classes, e também um modelo ER (Entidade Relacionamento), com as principais tabelas e seus relacionamentos. É válido ressaltar que de acordo com andamento do projeto, os componentes envolvidos na modelagem podem ser alterados a fim de atender o objetivo proposto.

4.4.1 Requisitos

Em um primeiro momento a solução proposta nesse trabalho possui apenas a necessidade de atender requisitos funcionais, que por sua vez descrevem as funcionalidades que o sistema oferece, e o que ele está apto a fazer. A Tabela 10 apresenta os requisitos funcionais do sistema.

TABELA 10 – Requisitos funcionais

Descrição
O sistema deve fornecer uma interface web com um cadastro de usuário simples.
O sistema deve permitir o upload de arquivos de imagem (JPG, GIF, PNG e TIFF).
O sistema deve fornecer um serviço, bem como uma API, que gere CAPTCHAs para *websites*.
O sistema deve fornecer um *link* para *download* dos arquivos digitalizados.

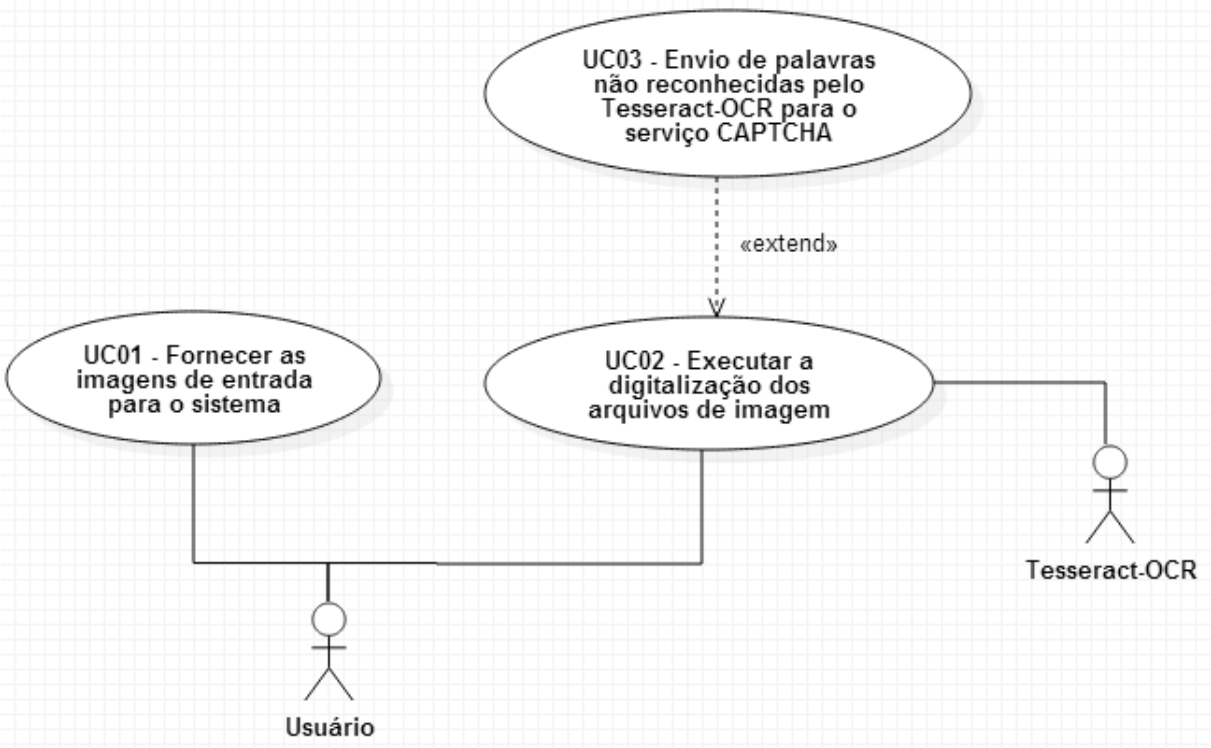
4.4.2 Diagramas UML

A fim de amparar o desenvolvimento do projeto com base nos requisitos analisados, foram elaborados alguns diagramas UML (*Unified Modeling Language*). Embora a definição da UML 2.0 especifique 13 diagramas diferentes, entende-se que para a especificação básica deste trabalho, são suficientes somente os diagramas de Caso de Uso e Diagrama de Classes.

O diagrama de casos de uso do sistema (Figura 16), apresenta as funcionalidades a nível de usuário da interface *web*. Conforme é possível visualizar no diagrama da Figura 16, o agente Usuário deve fornecer os arquivos de imagens para o sistema, e posteriormente executar a digitalização dos mesmos. A digitalização faz o uso de um agente externo denominado Tesseract-OCR, que ficará responsável pelo reconhecimento das palavras presentes nos arquivos de imagem. No caso de alguma palavra não ser reconhecida, o terceiro caso de uso

deve ser responsável por enviá-las para o serviço CAPTCHA. Devido ao fato da API CAPTCHA ser um serviço que *websites* poderão utilizar como ferramenta de segurança, essa interface não possui um diagrama de casos de uso a nível de usuário.

FIGURA 16 – Diagrama de casos de uso da interface web

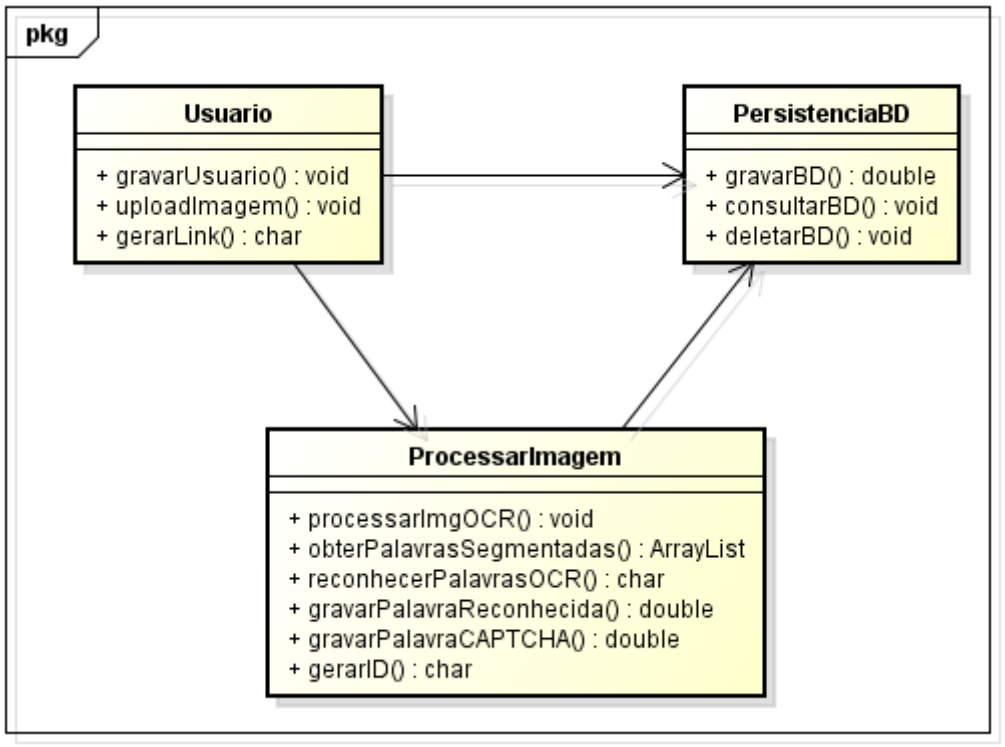


Um diagrama de classes especifica como um sistema orientado a objetos foi desenvolvido, apresentando as suas principais classes, atributos, métodos e como se relacionam conforme apresentado na Figura 17 e na Figura 18.

A Figura 17 especifica as principais classes que compõem o aplicativo OCR. As funcionalidades básicas de cada classe são:

- Usuario: essa classe tem como objetivos gerenciar o cadastro de usuário, enviar os dados para a classe PersistenciaBD, enviar os arquivos de imagem para a classe ProcessarImagem, e recuperar os arquivos processados;
- ProcessarImagem: essa classe é responsável pelo processamento dos arquivos de imagens através da biblioteca OCR, após esse processamento os dados resultantes são enviados para a classe PersistenciaBD;
- PersistenciaBD: essa classe é responsável unicamente pela consulta e persistência dos dados no banco de dados.

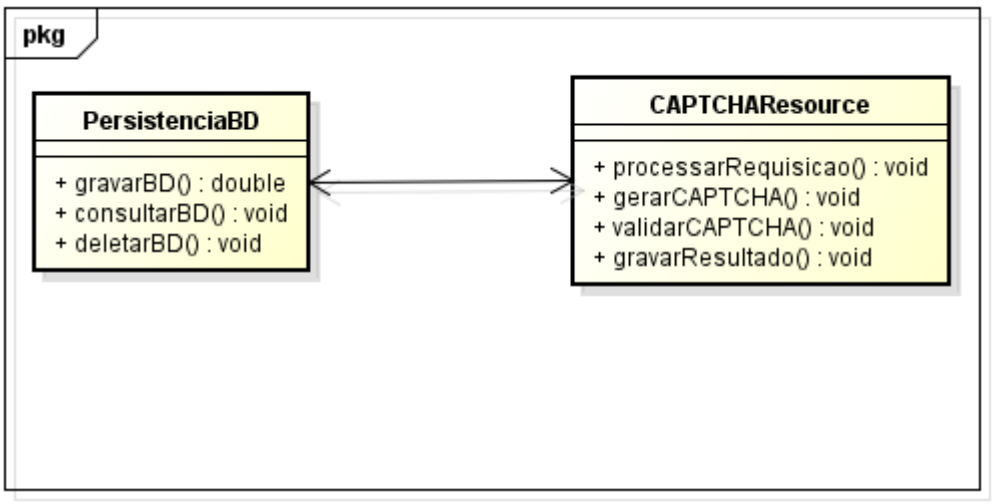
FIGURA 17 – Diagrama de classes do sistema, classes do aplicativo OCR



A Figura 18 especifica as principais classes que compoem a API CAPTCHA. As funcionalidades básicas de cada classe são:

- CAPTCHAResource: essa classe é responsável por fornecer o serviço que disponibiliza a API CAPTCHA. A sua principal função é gerar um CAPTCHA para a autenticação do usuário, e obter um trecho de texto, não identificado pelo OCR. Essa classe também valida o retorno da autenticação feito pelo usuário, e envia os dados resultantes para a classe PersistenciaBD;

FIGURA 18 – Diagrama de classes do sistema, classes da API CAPTCHA



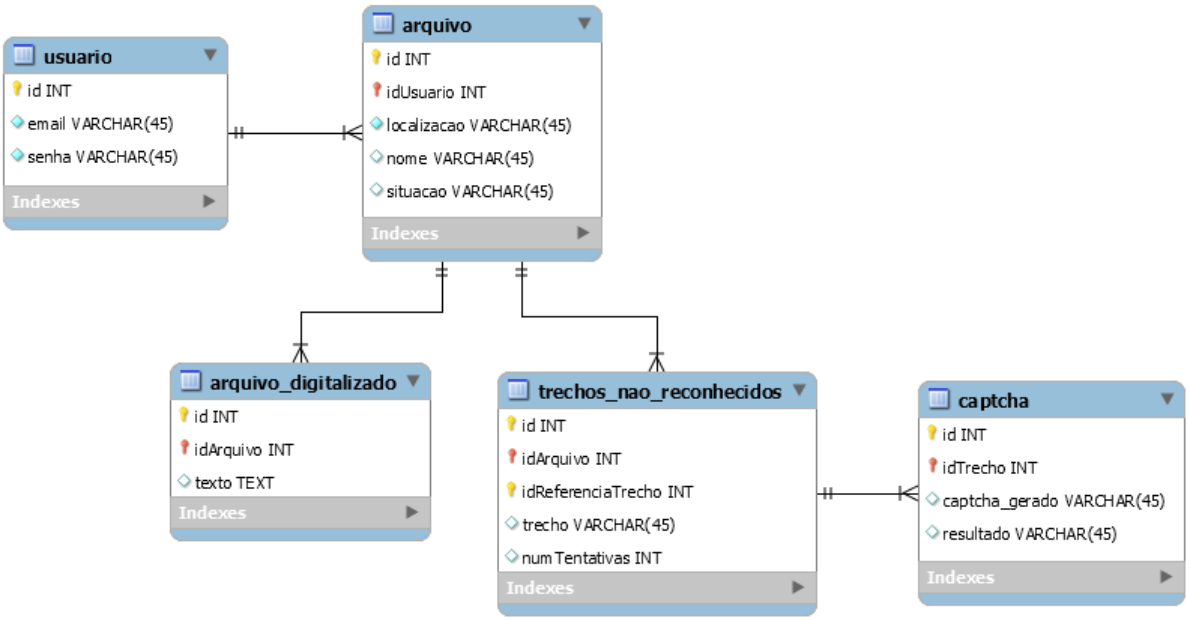
- PersistenciaBD: essa classe é a mesma utilizada pelo aplicativo OCR.

4.4.3 Banco de Dados

No diagrama apresentado na Figura 19, constam as principais tabelas, os seus principais atributos e relacionamentos. Todas elas estão representadas de uma forma genérica para um melhor entendimento do seu funcionamento. As funções de cada tabela são destacadas a seguir:

- usuario: a função dessa tabela é armazenar as informações do usuário;
- arquivo: a função dessa tabela é armazenar os arquivos que o usuário deseja digitalizar;
- arquivo_digitalizado: a função dessa tabela é armazenar os arquivos processados pelo OCR;
- trechos_nao_reconhecidos: a função dessa tabela é armazenar os trechos de textos não reconhecidos pelo OCR;
- captcha: essa tabela deve armazenar os CAPTCHAS gerados para a autenticação do usuário e uma referência ao trecho de texto a ser reconhecido.

FIGURA 19 – Diagrama EER simplificado da aplicação



5 DESENVOLVIMENTO E IMPLEMENTAÇÃO DO PROTÓTIPO

O desenvolvimento do protótipo seguiu o modelo de arquitetura apresentada na seção 4.2, e o padrão de desenvolvimento MVC apresentado na seção 4.3. No entanto, no decorrer do desenvolvimento, alguns itens apresentados na solução proposta foram alterados, a fim de atender o objetivo proposto. Nas próximas sessões serão apresentadas as alterações e etapas realizadas no andamento do desenvolvimento.

5.1 INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

O desenvolvimento do protótipo iniciou com a instalação e configuração dos componentes utilizados no ambiente de desenvolvimento:

- Sistema Operacional: Linux Ubuntu v14.10;
- Servidor *Web*: Apache 2.4;
- Linguagem de programação: PHP v5.6;
- Banco de dados: MySQL v5.6;
- Biblioteca OCR: Tesseract-OCR v3.02 (com Leptonica v1.69) + Pacote de idioma Português;
- IDE (*Integrated Development Environment*): NetBeans v8.0.2;
- Navegador: Mozilla Firefox v36.

5.2 ESTRUTURAÇÃO DO PROTÓTIPO

Conforme abordado na proposta de solução, a arquitetura REST é uma evolução do padrão MVC, onde a interação entre cliente (*View*) e servidor (*Controller*) é feita através requisições aos serviços disponibilizados pelo *WebService*. Nesse protótipo as requisições são feitas a partir de chamadas *ajax* (***A**synchronous **J**avascript **a**nd **X**ML*), utilizando bibliotecas *Javascript*. A utilização dessa arquitetura visa o desacoplamento da camada de apresentação do restante da aplicação. A utilização do *ajax* possibilita que as informações sejam carregadas na tela em tempo real, sem que haja a necessidade da atualização da página.

Nas próximas sessões estão descritas as estruturas e tecnologias utilizadas no desenvolvimento que cada camada da aplicação.

5.2.1 Camada de controle (*Controller*)

A camada de controle foi estruturada para atender as requisições feitas pela camada de visualização, processá-las e retornar um resultado. Nessa camada foram implementados: o *WebService*, para atender essas requisições, e as classes responsáveis pelas regras de negócio.

A fim de agilizar e facilitar o desenvolvimento do *WebService*, optou-se pela utilização de um *framework open source* que implementa a arquitetura REST. O *framework* utilizado é o *Slim Framework*[1] que mostrou ser o mais adequado, por ser leve, prático e conter uma documentação detalhada.

As etapas do desenvolvimento do *WebService* consistiram basicamente na criação de um arquivo, denominado *endpoint.php*, responsável por implementar o *framework Slim* e os métodos específicos da aplicação. Todas as requisições feitas pela camada de visualização, são direcionadas ao arquivo *endpoint.php*, onde são encaminhadas para o método correspondente. Conforme mencionado na seção 4.3, na arquitetura REST o formato JSON é o mais utilizado para a troca mensagens entre cliente e servidor e por esse motivo foi o formato escolhido.

As requisições recebidas pelo *WebService*, são enviadas para as classes de regras de negócio. No protótipo desenvolvido, essas classes são responsáveis pela validação, autenticação, processamento, *upload* de arquivos no formato de imagem, interação com as classes que compõem a camada de modelo, e integração com o *software* Tesseract-OCR. A integração com o *software* OCR, para a digitalização dos arquivos, está descrita detalhadamente na seção 5.3. Após a realização dos procedimentos necessários, é retornado o resultado para a camada de visualização.

5.2.2 Camada de modelo (*Model*)

Na camada de modelo, estão dispostas as classes responsáveis pelas funções de inclusão, alteração, consulta e exclusão de registros na base de dados. Nessa camada também estão as classes de entidade. Essas classes fornecem modelos orientados a objetos das tabelas presentes na base de dados. Esses modelos fornecem uma automatização na geração dos *SQLs*, que são enviados ao banco de dados.

Conforme mencionado na seção 4.3, o banco de dados utilizado para o desenvolvimento do protótipo é o MySQL. A conexão entre a aplicação *PHP* e o MySQL utiliza funções nativas do *PHP*. A classe responsável por estabelecer a conexão, e implementar as

[1] Informações disponíveis em http://www.slimframework.com

funções nativas, utiliza o padrão de projeto *Singleton*[2]. Esse padrão garante a existência de apenas uma instancia da classe, não sendo necessário estabelecer uma nova conexão com o banco de dados a cada nova inclusão, consulta, alteração ou exclusão.

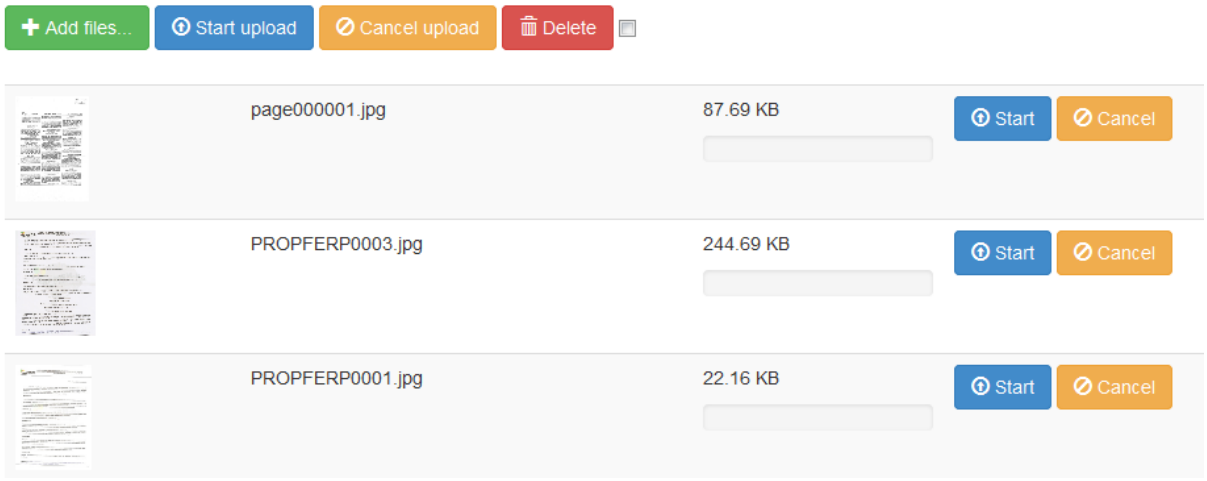
5.2.3 Camada de visualização (*View*)

Conforme o modelo apresentado na Figura 15, os dois componentes que fazem parte da camada de visualização (*View*) são: a interface *web*, onde os usuários farão o *upload* dos arquivos a serem digitalizados e a API CAPTCHA. Ambos os componentes consomem o *WebService* implementado pela camada de controle.

Os componentes dessa camada foram escritos em HTML, com auxílio do *framework Bootstrap*[3]. Esse *framework* é amplamente utilizado para aplicações *front-end*, e conta com uma série de bibliotecas *JavaScript* e *CSS*. Dentre as principais características desse *framework* estão: a compatibilidade com a maioria dos navegadores e com diversos tamanhos de telas.

Existem alguns *plugins*, para *upload* de arquivos, que são disponibilizados gratuitamente e que implementam o *Bootstrap*. A utilização desses *plugins* visa agilizar o processo de desenvolvimento. Dentre todos os *plugins* disponíveis, optou-se pelo uso do *jQuery File Upload Plugin*[4]. Esse *plugin* foi escolhido principalmente pela facilidade de implementação e customização. A Figura 20 apresenta um *layout* demonstrativo de utilização do *plugin* escolhido.

FIGURA 20 – *Layout* demonstrativo do *jQuery File Upload Plugin*



Fonte: https://blueimp.github.io/jQuery-File-Upload

[2] Mais informações em http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/pat/singleton.htm
[3] Mais informações em http://getbootstrap.com/
[4] Disponível em https://github.com/blueimp/jQuery-File-Upload

Esse *plugin* também disponibiliza uma classe em linguagem *PHP* no lado do servidor, para receber os arquivos carregados. Como essa classe implementa a arquitetura REST, optou-se pela sua utilização, restando apenas implementá-la ao *WebService*. A implementação consistiu basicamente em criar uma instancia dessa classe no arquivo que recebe as requisições (*endpoint.php*). Os arquivos carregados pelo usuário são armazenados em uma pasta do servidor, ao qual apenas o usuário logado tem acesso.

5.3 INTEGRAÇÃO DO PROTÓTIPO COM O *SOFTWARE* TESSERACT-OCR

Conforme abordado na proposta de solução, a integração entre o protótipo e o *software* Tesseract-OCR, deveria ser feita utilizando uma interface denominada php-tesseract, a fim de facilitar o desenvolvimento. Porém verificou-se que a interface, além de possuir uma documentação insuficiente, não atendia a alguns requisitos necessários para o funcionamento da aplicação, e seu uso foi descartado.

Dentre os requisitos não atendidos, destacam-se, a obtenção do fragmento de imagem, contendo a palavra processada, o texto resultante, e alguma informação que revele se a palavra foi ou não reconhecida pelo OCR. A medida que o *software* Tesseract-OCR foi sendo estudado, percebeu-se que o mesmo não possui métodos nativos que atendam a esses requisitos, e a partir disso verificou-se a necessidade da alteração do seu código fonte. No entanto, antes de realizar essa alteração, foi necessário entender como o Tesseract classifica as palavras processadas.

Com um estudo mais aprofundado, observou-se que o Tesseract classifica as palavras digitalizadas com uma taxa de certeza, ou seja, a probabilidade de uma palavra digitalizada estar correta. A taxa de certeza é calculada no momento da classificação das palavras e envolve cálculos complexos. A única informação disponível é que essa taxa pode variar entre 0,0 e -20,0. Quanto mais próxima de zero, maior a probabilidade da palavra estar correta. Para um melhor entendimento, existe um método no próprio Tesseract que converte esse número em percentual (*P*), utilizando a seguinte fórmula: $P = 100 + (5 * \textit{taxa de certeza})$

A utilização da taxa de certeza se mostrou necessária para a classificação das palavras entre reconhecidas e não reconhecidas. O arquivo de configuração de dicionário do Tesseract, define que palavras com uma taxa de certeza acima de -2,25 estão aptas a serem gravadas em seu dicionário interno. Tendo em vista essa informação, definiu-se que a classificação das palavras utilizará a mesma regra para definir se as palavras foram reconhecidas.

Sabendo-se da necessidade de alteração do código fonte do Tesseract-OCR, foram definidas e implementadas as funcionalidades essenciais que este deve conter para atender as necessidades da aplicação proposta. Essas puderam ser feitas alterando-se um único método do Tesseract. Originalmente, a função desse método consistia em imprimir cada palavra processada em um arquivo TXT. Atualmente o método realiza as seguintes etapas:

- As palavras processadas são divididas em fragmentos, onde cada fragmento contém uma palavra extraída do arquivo de imagem original (Figura 21). Nessa etapa também ocorre a geração de um arquivo XML para cada fragmento, contendo os dados da palavra extraída. O arquivo XML gerado (Figura 22) contém informações como o resultado do fragmento processado (text), a taxa de certeza (word_certainty), e um identificador único, que corresponde ao nome do fragmento;
- Após o processamento de todas as palavras do arquivo digitalizado, é gerado um arquivo TXT com a finalidade de preservar a estrutura original do texto. Cada palavra do texto é substituída pelo identificador único gerado no arquivo XML (Figura 23). Conforme é possível visualizar no exemplo de arquivo TXT da Figura 23, cada posição (ex.: 0000016) é uma referência para o identificador único do arquivo XML;
- Os arquivos gerados são armazenados em uma pasta temporária.

FIGURA 21 – Exemplo de fragmento de palavra digitalizada

FIGURA 22 – Exemplo de XML com o resultado do processamento

```
<?xml version="1.0" encoding="UTF-8"?>
<word_info>
   <text>rzipida</text>
   <word_certainty>-4.422705</word_certainty>
   <img_id>0000016</img_id>
</word_info>
```

FIGURA 23 – Exemplo de TXT com as referências das palavras no texto

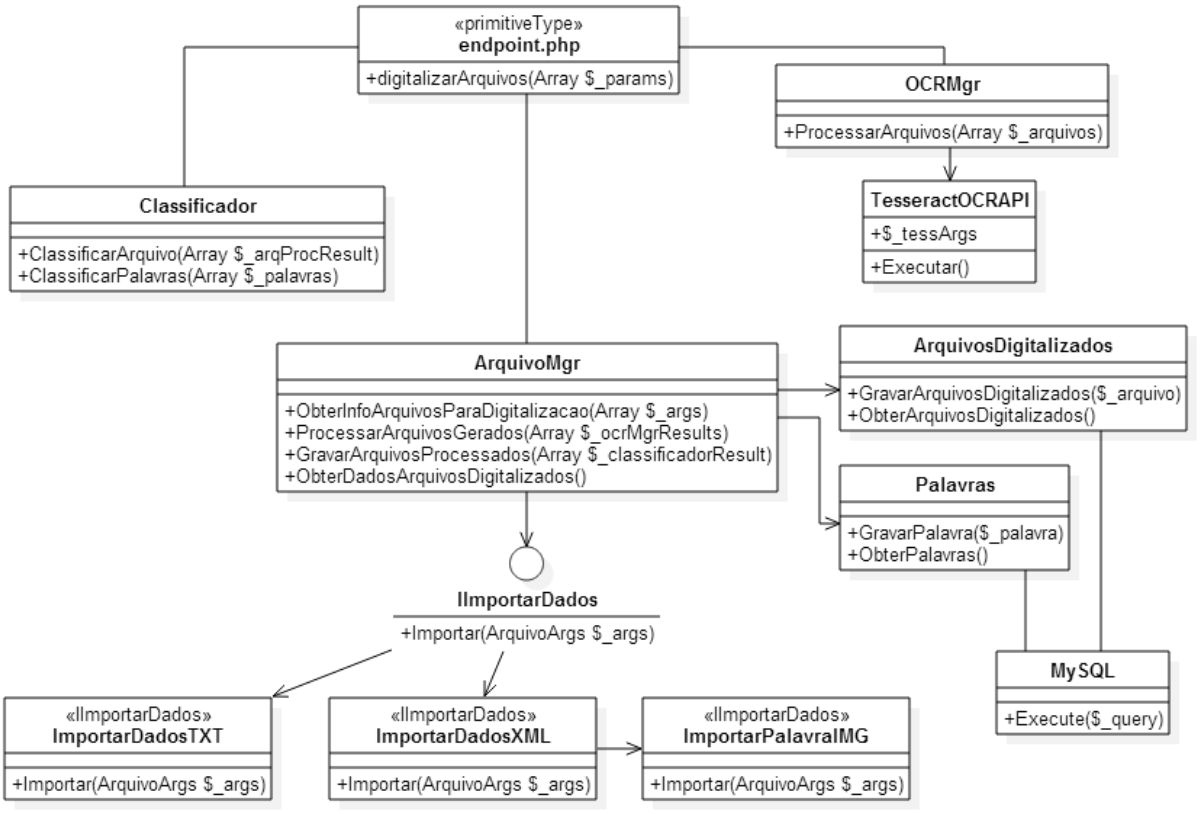
```
0000001 0000002 0000003

0000004 0000005 0000006 0000007 0000008 0000009 0000010 0000011 0000012
0000013 0000014 0000015 0000016 0000017 0000018 0000019 0000020 0000021 0000022
0000023 0000024 0000025 0000026 0000027 0000028 0000029 0000030 0000031 0000032
0000033 0000034 0000035 0000036 0000037 0000038 0000039 0000040 0000041 0000042
0000043 0000044 0000045 0000046 0000047 0000048 0000049 0000050 0000051 0000052
```

As alterações possibilitaram uma forma mais simples de integração com o *software* Tesseract-OCR, sem comprometer as funcionalidades do mesmo. A linguagem *PHP* possui uma função nativa que permite chamadas por linha de comando, assim como os comandos executados em um terminal Linux, por exemplo. Com as funcionalidades implementadas, iniciou-se o desenvolvimento das classes responsáveis pela integração entre a aplicação e o *software* Tesseract-OCR. Nessa etapa também foi definido que o idioma padrão a ser parametrizado na digitalização de um arquivo seria o português. A Figura 24 apresenta as principais classes e componentes envolvidos na integração desenvolvida:

- endpoint.php: Esse componente é responsável por receber as requisições do usuário, processá-las e retornar o resultado. Nesse caso a digitalização de um arquivo;
- ArquivoMgr: Essa classe é responsável por obter os acervos a serem digitalizados, processar os arquivos gerados pela alteração feita no Tesseract-OCR, e enviar os dados resultantes para a gravação;
- OCRMgr: A função dessa classe é fazer a parametrização dos arquivos a serem digitalizados e retornar o resultado do processamento;
- TesseractOCRAPI: Essa classe recebe as parametrizações e executa o comando para a biblioteca externa do Tesseract-OCR;
- ImportarDadosTXT: Essa classe obtém o conteúdo do arquivo contendo a referência das palavras no texto (Figura 23);
- ImportarDadosXML: Essa classe obtém os dados de cada palavra processada pelo Tesseract-OCR (Figura 22);
- ImportarPalavraIMG: Essa classe obtém o fragmento da palavra gerado pelo Tesseract-OCR (Figura 21);
- Classificador: Essa classe faz o processamento dos dados coletados e classifica as palavras, de acordo com a taxa de certeza;
- ArquivosDigitalizados: Essa classe implementa os métodos de consulta e gravação específicos dos arquivos digitalizados;
- Palavras: Essa classe implementa os métodos de consulta e gravação específicos das palavras processadas;
- MySQL: Por fim, essa classe faz a comunicação entre a aplicação e o banco de dados.

FIGURA 24 – Diagrama de classes da integração entre protótipo e o Tesseract-OCR

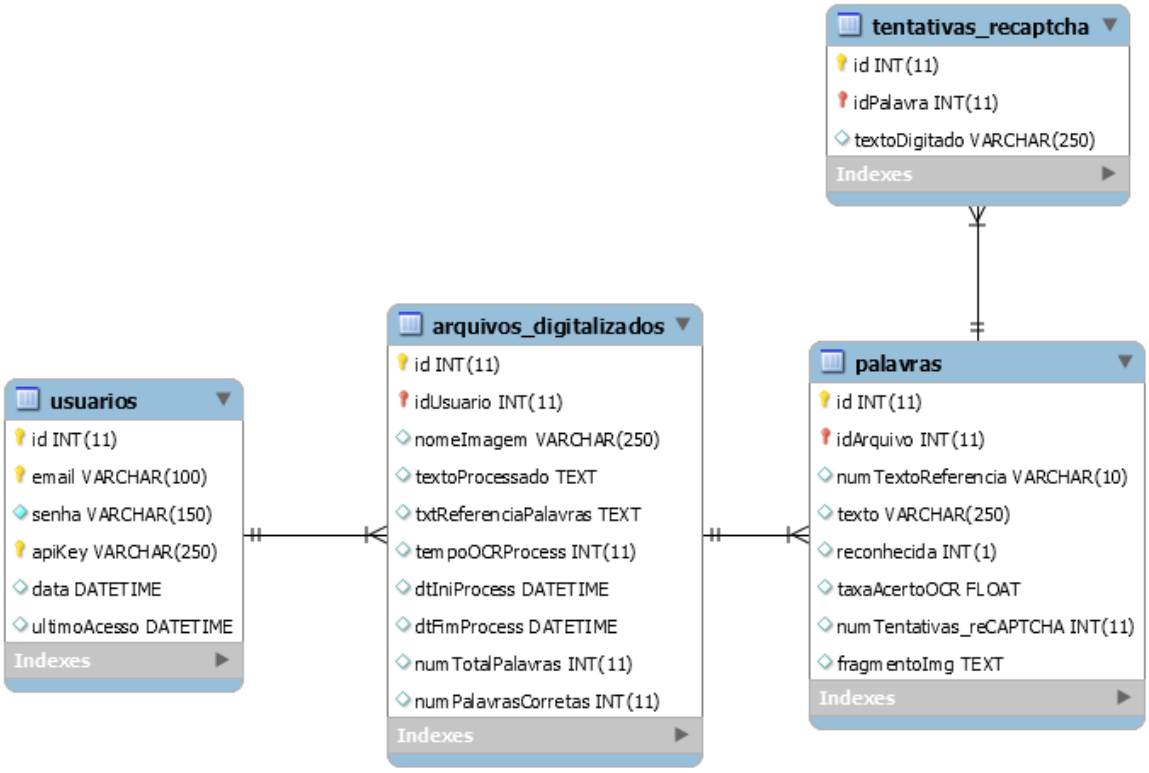


5.4 REMODELAGEM DA BASE DE DADOS DO PROTÓTIPO

No decorrer do desenvolvimento da integração entre o protótipo e o *software* Tesseract, percebeu-se que a modelagem da base de dados da aplicação deveria ser alterada e até simplificada. Como os arquivos ainda não digitalizados ficam salvos em uma pasta onde apenas o usuário tem acesso, não houve necessidade em manter a tabela para os arquivos carregados. A estrutura de tabelas ainda foi simplificada aproveitando-se as palavras classificadas como reconhecidas para fazer o teste de CAPTCHA e as palavras não reconhecidas para serem decifradas. As funções de cada tabela estão destacadas a seguir:

- usuarios: Essa tabela contém os dados de cadastro do usuário que são utilizados para a autenticação;
- arquivos_digitalizados: Essa tabela contém os dados dos arquivos processados pelo Tesseract;
- palavras: Essa tabela contém os dados das palavras presentes no arquivo digitalizado, sejam elas reconhecidas ou não;
- tentativas_recaptcha: Nessa tabela serão armazenadas as tentativas de reconhecimento de uma palavra classificada como não reconhecida. A cada autenticação feita pela API CAPTCHA, será gravado um novo registro contendo o texto digitado.

FIGURA 25 – Diagrama EER remodelado do protótipo



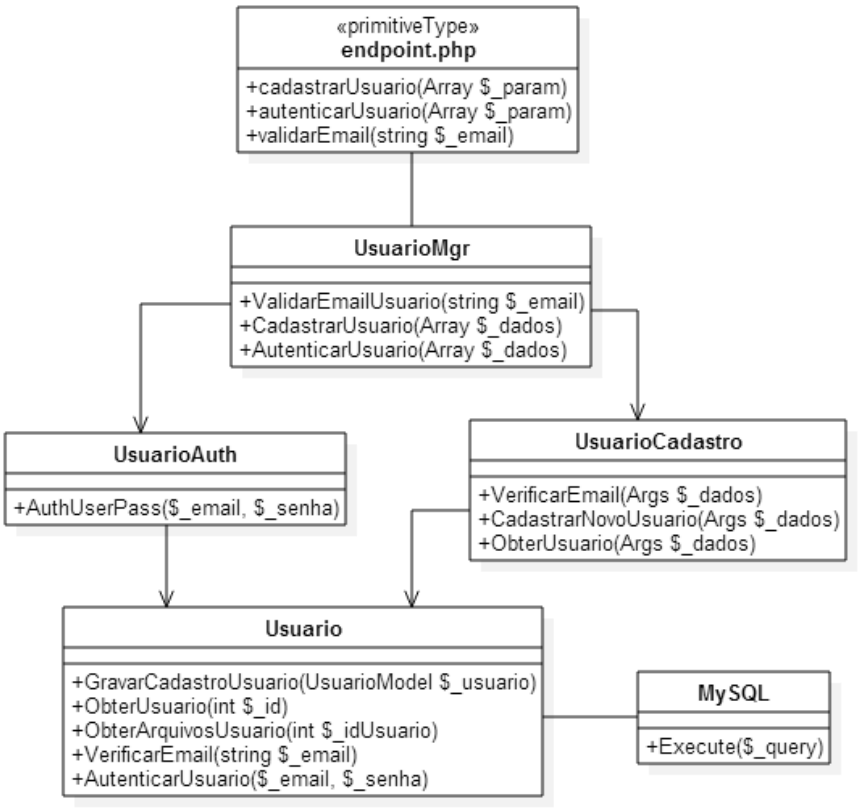
5.5 MODELAGEM E INTERFACES DA APLICAÇÃO

Com a estruturação do protótipo definida, a integração com o *software* Tesseract-OCR realizada e o modelo da base de dados remodelado, iniciou-se a modelagem das classes que compõem o restante da aplicação e as suas interfaces de interação. Nas próximas sessões serão abordadas as interfaces e os diagramas de classes envolvidos no funcionamento do protótipo.

5.5.1 Cadastro e autenticação de usuário

O cadastro e autenticação do usuário estão presentes na mesma interface, ou seja, na mesma página. Como a aplicação requer apenas informações básicas do usuário, foi possível adaptar as duas telas, de cadastro e autenticação, em uma única interface. Antes de criar a interface, que compõe a camada de visualização (*View*), foi necessário modelar e desenvolver os métodos responsáveis por atender aos requisitos de cadastro e autenticação. O diagrama de classes da Figura 26 apresenta as principais classes e componentes envolvidos no cadastro e autenticação do usuário.

FIGURA 26 – Diagrama de classes de cadastro e autenticação de usuário



Conforme definido na estruturação do protótipo, esses métodos devem ser disponibilizados pelo *WebService* a fim de serem consumidos pela camada de visualização, nesse caso, a interface de cadastro e autenticação de usuário. A Figura 27 apresenta a tela de cadastro de usuário, nessa tela são requisitados um *e-mail* e uma senha com confirmação. Nessa

FIGURA 27 – Interface de cadastro do usuário

tela o *e-mail* é validado para que ele seja cadastrado apenas uma vez, caso o usuário informe um *e-mail* já cadastrado, um aviso é exibido e a operação abortada. Se todos os dados forem informados corretamente, o cadastro é realizado e o usuário é direcionado para a interface de *upload* e digitalização de arquivos.

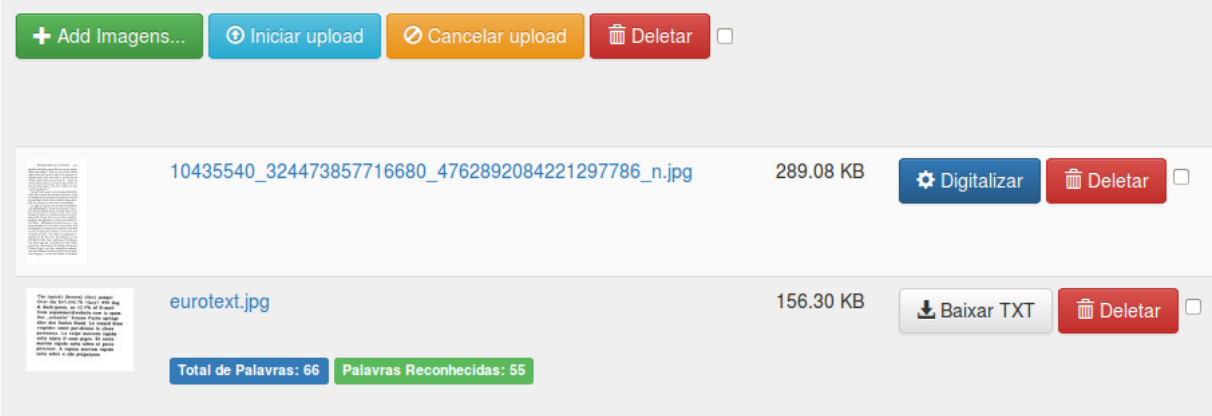
Na tela inicial (Figura 28), está disponível a tela de autenticação, utilizada para acessar a interface de *upload* e digitalização de arquivos. Essa tela conta com dois campos: *e-mail* e senha, que são validados a cada tentativa de acesso. No caso do usuário informar um *e-mail* inexistente e/ou uma senha inválida, um aviso é exibido e a operação abortada.

FIGURA 28 – Interface de autenticação do usuário

5.5.2 *Upload* e digitalização de arquivos

Conforme mencionado na seção de estruturação do protótipo, o *upload* é feito utilizando-se um *pluging* e os arquivos carregados são salvos em pastas separadas por usuário no servidor. Como esse componente não faz uso do banco de dados e é composto apenas de

FIGURA 29 – Interface para *upload* e digitalização de arquivos



uma classe na camada de controle, não houve necessidade da modelagem de um diagrama de classes.

A Figura 29 apresenta a interface para *upload* e digitalização dos arquivos. Nessa figura é possível observar os recursos que a interface oferece. Ela disponibiliza recursos de *uploads* de arquivos no formato de imagem, o cancelamento de *uploads*, deleção dos arquivos carregados, digitalização dos arquivos e *download* dos arquivos digitalizados em formato TXT.

Assim que é feito o *upload* de um arquivo nessa tela, é possível iniciar o processo de digitalização. Ao efetuar a ação de digitalizar um arquivo, o protótipo realiza a integração com o *software* Tesseract-OCR e persiste os dados resultantes na base de dados, conforme mencionado anteriormente na seção 5.3. Esse processamento leva apenas alguns segundos para ser realizado e ao final dele é informado o total de palavras encontradas no arquivo e o total de palavras que foram classificadas como reconhecidas, de acordo com a taxa de certeza. Também, após essa etapa, já está disponível para *download* o arquivo digitalizado em formato TXT, mesmo que todas as palavras ainda não tenham sido reconhecidas. Essas palavras ainda devem passar pela API CAPTCHA, e como isso pode demorar alguns dias, o usuário pode baixar o arquivo digitalizado e verificar o andamento da digitalização. Mais detalhes sobre o funcionamento da API CAPTCHA serão abordados na próxima seção.

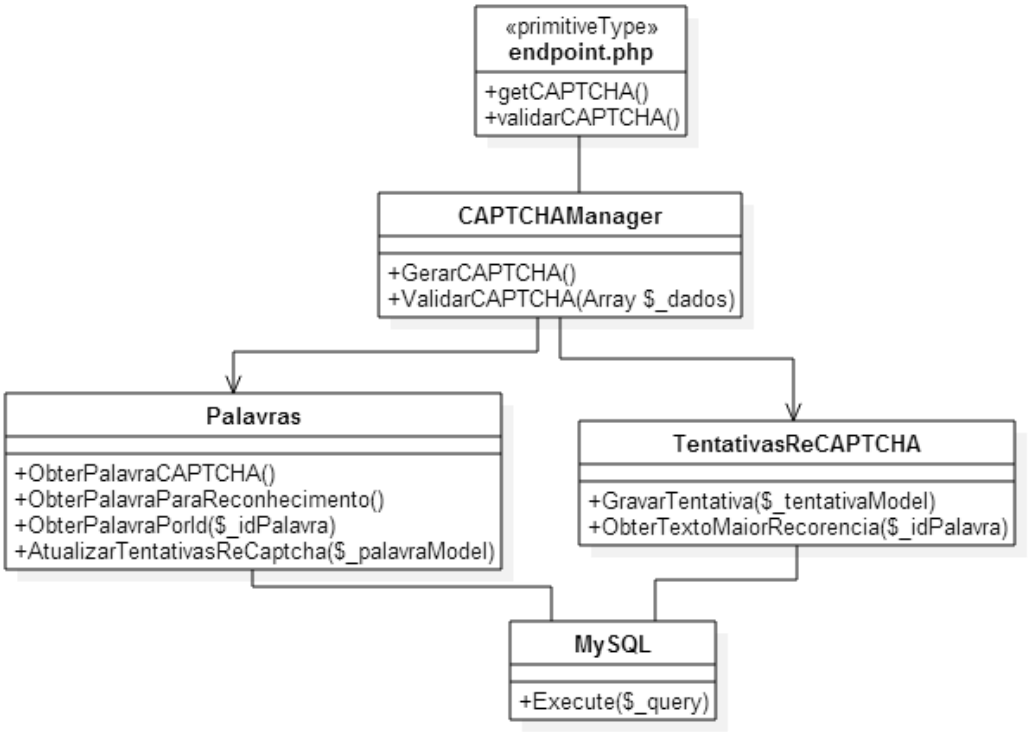
5.5.3 API CAPTCHA

O desenvolvimento da API CAPTCHA também foi baseado na arquitetura REST e faz uso das mesmas tecnologias apresentadas na seção 5.2.1. Essa API, disponibilizada pelo *WebService*, possui dois métodos, um para geração dos CAPTCHAs e outro para validá-los. O diagrama de classes da Figura 30 apresenta as principais classes e componentes envolvidos no funcionamento desses métodos. Como a API é um serviço que recebe e envia requisições, ela não possui uma interface de interação com o usuário. Cada *website* que decidir adotar essa API, como ferramenta de segurança, deverá consultar a sua documentação e implementá-la na sua página. A Figura 31 apresenta um exemplo de interface que foi desenvolvida para fins de testes e validação da API.

A função do método de geração de CAPTCHAs é buscar na base de dados uma palavra com o resultado conhecido, cuja função é fazer o teste para diferenciar um usuário humano de um "robô", conforme explicado na seção 2.3. Esse método também busca uma palavra, classificada como não reconhecida, para ser identificada pelos usuários espalhados pela *web*, aplicando-se assim o conceito de *crowdsourcing*, definido na seção 2. As palavras classificadas

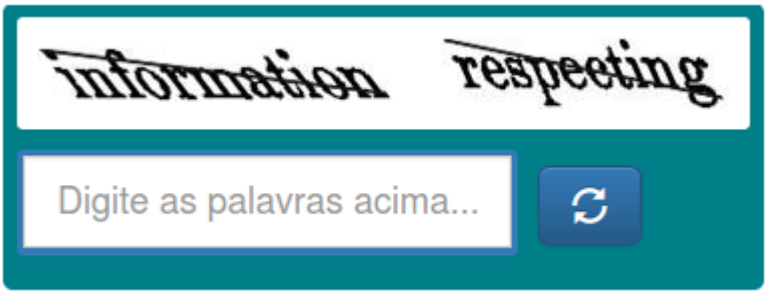
como não reconhecidas são selecionadas de acordo com a sua taxa de certeza, de forma ascendente, ou seja, da menor para maior. Lembrando que essa taxa pode variar entre -20.0 e 0.0. Dessa forma, as palavras que possuem maior probabilidade de estarem incorretas, passarão antes pelos testes de CAPTCHA.

FIGURA 30 – Diagrama de classes da API CAPTCHA



Após a seleção das palavras, suas imagens são levemente distorcidas, para dificultar ainda mais a ação de *scripts* automatizados. A fim de forçar o usuário a digitar as duas palavras corretamente, optou-se por colocá-las lado a lado, sendo que a segunda palavra é a que possui o resultado conhecido e tem a função de autenticar o usuário. Na Figura 31, por exemplo, a palavra "*information*" corresponde a uma palavra classificada como não reconhecida e a palavra "*respecting*" corresponde a uma palavra que possui um resultado conhecido.

FIGURA 31 – Interface de testes da API CAPTCHA



O método de validação do CAPTCHA, é responsável por autenticar um usuário humano e também por persistir o resultado dessa autenticação na base de dados. Conforme mencionado anteriormente, a segunda palavra é utilizada para a validação do usuário e a primeira para o reconhecimento.

O processo de autenticação tem início comparando-se o texto da segunda palavra com o seu texto no banco de dados. Se os textos forem compatíveis, ou seja, se a segunda palavra for digitada corretamente, assume-se que a primeira palavra digitada também está correta. Nessa etapa o texto da primeira palavra também é comparado com o seu valor no banco de dados, se os valores forem compatíveis, ela é assinalada como reconhecida. Caso contrário, essa mesma palavra passará por outros nove novos testes de CAPTCHA, totalizando dez tentativas de reconhecimento, todas essas armazenadas na base de dados. Ao final das dez tentativas, o valor com maior recorrência é definido como o texto final para essa palavra e então ela é definida como reconhecida.

5.6 CONSIDERAÇÕES FINAIS

Neste capítulo foram descritas as etapas envolvidas no desenvolvimento do protótipo. Também foram abordadas as tecnologias, ferramentas, bibliotecas, *frameworks* e arquitetura utilizada. Todas essas se mostraram de extrema importância para o sucesso do desenvolvimento.

Também foram descritos o processo de classificação das palavras pelo Tesseract-OCR, as alterações feitas em seu código fonte e a forma de integração com a aplicação. O estudo mais aprofundado do código fonte do Tesseract-OCR possibilitou um melhor entendimento das suas características e funcionalidades. Isso possibilitou a alteração do seu código de uma forma clara e segura, a fim de atender as necessidades da aplicação.

Nesse capitulo também foram apresentadas capturas de tela das interfaces para demonstrar as funcionalidades da aplicação e a forma como interagem com o servidor. O uso da arquitetura REST para a comunicação entre interface e servidor, possibilitou um maior desacoplamento entre a camada de apresentação e a camada de controle, tornando o desenvolvimento mais simples e de fácil manutenção. Na próxima seção serão apresentados os testes realizados e os resultados obtidos a fim de validar as funcionalidades e comprovar a viabilidade da aplicação.

6 CASOS DE TESTE E RESULTADOS OBTIDOS

Nessa seção são realizados os casos de testes a fim de comprovar o funcionamento do protótipo e se o mesmo atingiu os objetivos propostos. Os testes se concentraram principalmente na digitalização de arquivos no formato de imagem para o formato TXT, a fim de comprovar a viabilidade da aplicação da solução aqui proposta. Também foram realizados casos testes nas funcionalidades de cadastro/acesso de usuários, *upload* de arquivos e API CAPTCHA.

Os casos de testes foram organizados em tabelas, cada qual apresentando os procedimentos e resultados obtidos nos testes realizados em cada uma das interfaces do protótipo. A Tabela 11 e a Tabela 12 apresentam os testes realizados nas interfaces de cadastro e de acesso do protótipo.

Tabela 11 - Caso de Teste: Cadastro de Usuário

Caso de Teste	Cadastro de Usuário
Pré-condições	Nenhuma
Passos	1. Informar um *e-mail* ainda não cadastrado na base de dados; 2. Informar uma senha; 3. Confirmar a senha; 4. Pressionar o botão “Cadastre-se”.
Resultados Esperados	Usuário cadastrado, autenticado e redirecionado para a interface de *upload* de arquivos.
Comentários	No momento em que o usuário informa um *e-mail*, a aplicação consulta em sua base de dados se o mesmo já está cadastrado, caso esteja, é exibida uma mensagem de erro. O botão “Cadastre-se” permanece desabilitado até que todas as informações sejam informadas corretamente. Mensagem de *e-mail* já cadastrado: 

Tabela 12 - Caso de Teste: Autenticação de Usuário

Caso de Teste	Autenticação de Usuário
Pré-condições	O usuário deve estar previamente cadastrado no sistema

(Continua)

(Conclusão)

Passos	1. Informar um *e-mail* válido; 2. Informar uma senha válida; **3.** Pressionar o botão “Entre”.
Resultados Esperados	Usuário autenticado e redirecionado para a interface de *upload* de arquivos.
Comentários	O usuário deve informar um *e-mail* e uma senha previamente cadastrados. No caso das informações não corresponderem ao que está cadastrado no banco de dados, uma mensagem de erro é exibida. Mensagem de erro de acesso: 

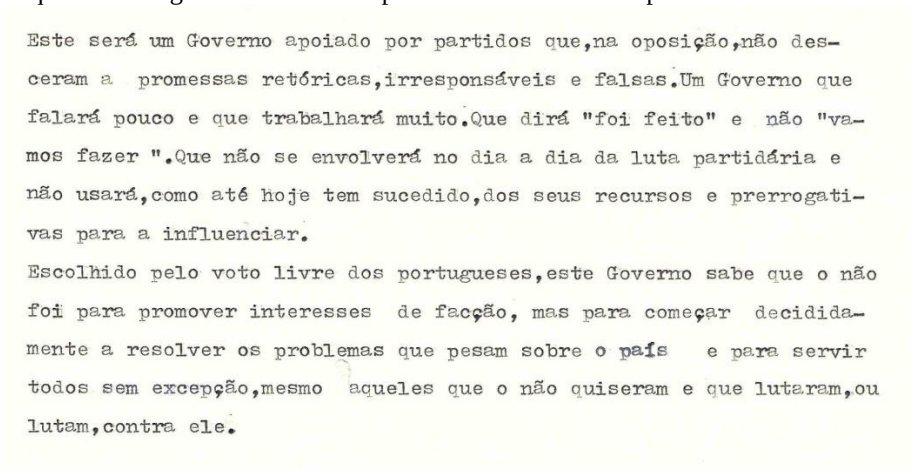
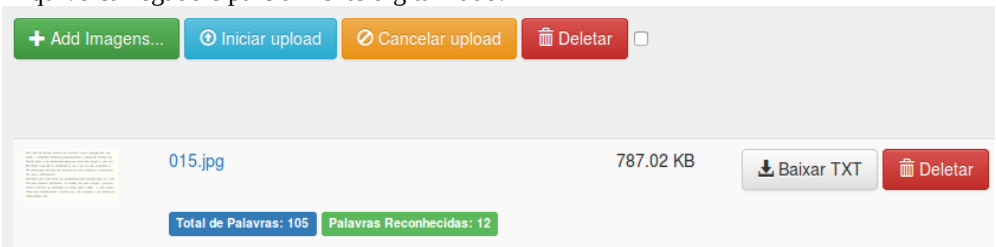
Para a realização dos testes na interface de *upload* e digitalização de arquivos foi necessário obter acervos previamente digitalizados, ou seja, arquivos de imagem gerados por um *scanner* ou qualquer máquina que realize foto cópia de um acervo impresso em papel. Visto que o arquivo final é gerado no formato TXT e também que o *software* Tesseract-OCR possui algumas limitações no processamento de documentos com *layouts* complexos, optou-se pela escolha de documentos com um *layout* simples e sem presença de imagens. Além disso, como um arquivo de imagem pode conter diversas características diferentes como o seu formato, resolução, densidade de *pixels*, etc., e sabendo que isso pode afetar diretamente o resultado do processamento do *software* OCR, foram definidos alguns requisitos mínimos que a imagem deve conter para a realização dos testes. A Tabela 13 contém os detalhes e os procedimentos realizados nos testes dessa interface.

Tabela 13 - Caso de Teste: *Upload* e Digitalização de Acervos

Caso de Teste	*Upload* e Digitalização de Acervos
Pré-condições	O usuário deve estar autenticado e ter acesso a interface de *upload* de arquivos;
Requisitos	1. Formatos de arquivo de imagem: JPEG, PNG ou GIF; 2. Tamanho máximo do arquivo: 10MB; 3. *Layout* do acervo: Uma coluna de texto sem imagens; 4. Resolução (Qualidade): Mínimo 100 DPI (*Dots Per Inch*);
Passos	1. Pressionar o botão “Add Imagens”; 2. Selecionar os arquivos desejados seguindo os requisitos; 3. Pressionar o botão “Iniciar upload”; 4. Aguardar o *upload* dos arquivos;

(Continua)

(Conclusão)

	5. Pressionar o botão “Digitalizar” para digitalizar um arquivo; 6. Aguardar o processamento; 7. Pressionar o botão “Baixar TXT” para baixar a digitalização parcial do arquivo.
Resultados Esperados	**Arquivos carregados e parcialmente digitalizados.**
Comentários	Arquivo de imagem de uma foto cópia de um documento impresso:  Este será um Governo apoiado por partidos que,na oposição,não des- ceram a promessas retóricas,irresponsáveis e falsas.Um Governo que falará pouco e que trabalhará muito.Que dirá "foi feito" e não "va- mos fazer ".Que não se envolverá no dia a dia da luta partidária e não usará,como até hoje tem sucedido,dos seus recursos e prerrogati- vas para a influenciar. Escolhido pelo voto livre dos portugueses,este Governo sabe que o não foi para promover interesses de facção, mas para começar decidida- mente a resolver os problemas que pesam sobre o país e para servir todos sem excepção,mesmo aqueles que o não quiseram e que lutaram,ou lutam,contra ele. Arquivo carregado e parcialmente digitalizado:  Arquivo TXT parcialmente digitalizado: Este será um Governo apoiado por partidos que,na oposição,não des- ceram a promessas reüóricas,irresponsáveis e fa1sas;Um Governo que falará pouco e que trabalhará muitoÇQue dirá "foi feito" e não "va- mos fazer ".Que não se envolverá no dia a dia da luta partidária e não usará,como até Hoje tem sucedido,dos seus recursos e prerrogati- vas para a inf1ueñciarQ Escolhido pelo voto livre dos portugueses,este Governo sabe que o não foi para promover interesses de facção, mas para começar decidida_ mente a resolver os problemas que pesam sobre 0 país e para servir todos sem excepção,mesmo aqueles que 0 não quiseram e que 1utaram,ou 1utam,contra ele;

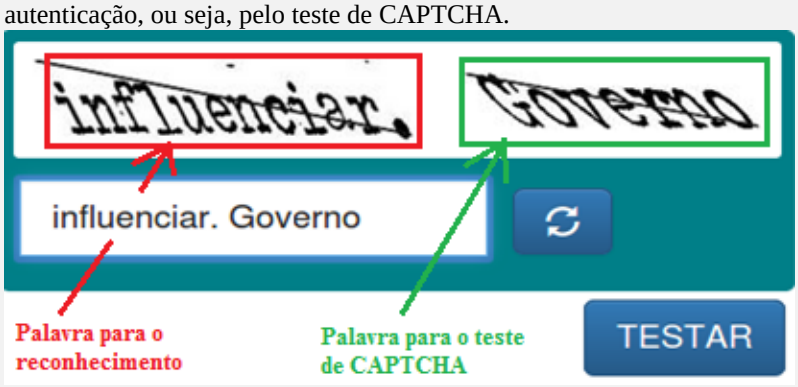
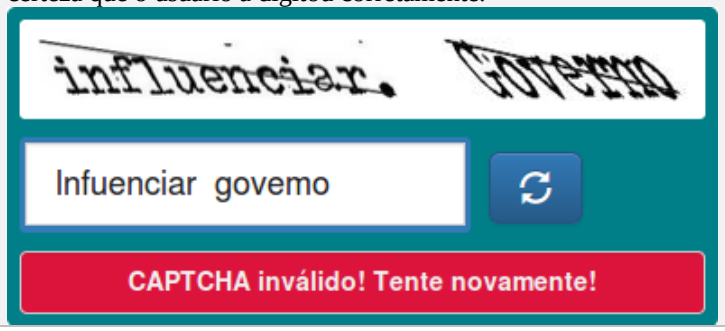
Por fim foram realizados testes de validação na API CAPTCHA. Conforme explicado na seção 5.5.3, a API não possui uma interface, mas para a realização dos testes foi desenvolvida uma interface para a validação das suas funcionalidades. A Tabela 14 contém os detalhes e os procedimentos realizados nos testes dessa interface.

Tabela 14 - Caso de Teste: Validação da API CAPTCHA

Caso de Teste	Validação da API CAPTCHA
Pré-condições	Devem conter pelo menos um arquivo parcialmente digitalizado na base de dados.

(Continua)

(Conclusão)

Requisitos	As palavras devem ser digitadas respeitando as seguintes regras: 1. Espaços; 2. Caracteres especiais e pontuação; 3. Letras maiúsculas e minúsculas.
Passos	1. Informar as duas palavras respeitando as regras acima; 2. Pressionar o botão "TESTAR";
Resultados Esperados	Usuário humano autenticado e palavra reconhecida.
Comentários	Abaixo está apresentada a interface de testes da API CAPTCHA, contendo duas das palavras do documento digitalizado anteriormente. Ambas as palavras são selecionadas na base de dados, a primeira palavra "influenciar." é enviada para o usuário fazer o reconhecimento, já que essa não possui um resultado conhecido. A segunda "Governo" possui um resultado conhecido e é enviada para fazer a autenticação do usuário. É válido ressaltar que a primeira palavra só é processada pela aplicação se o usuário passar pela autenticação, ou seja, pelo teste de CAPTCHA.  Arquivo TXT do documento digitalizado após a realização do teste acima: Este será um Governo apoiado por partidos que,na oposição,não desceram a promessas reüóricas,irresponsáveis e fa1sas;Um Governo que falará pouco e que trabalhará muitoÇQue dirá "foi feito" e não "vamos fazer ".Que não se envolverá no dia a dia da luta partidária e não usará,como até Hoje tem sucedido,dos seus recursos e prerrogativas para a influenciar. Escolhido pelo voto livre dos portugueses,este Governo sabe que o não foi para promover interesses de facção, mas para começar decididamente a resolver os problemas que pesam sobre 0 país e para servir todos sem excepção,mesmo aqueles que 0 não quiseram e que 1utaram,ou 1utam,contra ele; Caso o usuário não consiga resolver o teste CAPTCHA, no qual consiste em digitar a segunda palavra corretamente, a aplicação retorna um erro (Figura abaixo) e o processamento é abortado. Ou seja, a primeira palavra não é processada, já que não se tem certeza que o usuário a digitou corretamente. 

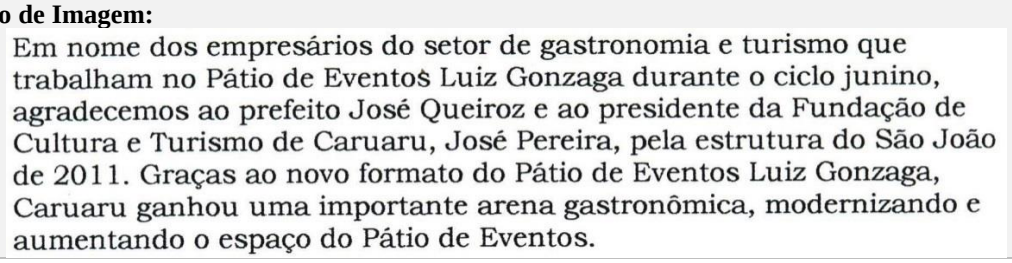
A fim de comprovar a viabilidade da aplicação da API CAPTCHA no auxílio a digitalização de acervos, foram realizados alguns casos de testes de digitalização. Esses foram

feitos utilizando foto cópias de fragmentos de alguns documentos selecionados aleatoriamente, seguindo os mesmos requisitos definidos anteriormente.

Os dados coletados nos testes estão apresentes na Tabela 15 e consistem basicamente no número de palavras presentes nos documentos, quantas dessas foram classificadas como reconhecidas e quantos testes de CAPTCHAs foram necessários realizar até que todas as palavras do documento fossem completamente reconhecidas. Conforme abordado na seção 5.3, as palavras são classificadas como reconhecidas de acordo com a sua taxa de certeza, que deve ser igual ou superior a -2,25.

Visto que esses testes foram realizados por um único usuário em um ambiente local, qualquer dado relativo ao tempo que um documento levou para ser completamente digitalizado, não corresponderia a realidade. Apenas em condições reais onde a API CAPTCHA seria utilizada em larga escala, seria possível fazer medições e estimativas relativas ao tempo que um documento levaria para ser digitalizado completamente. Na Tabela 15 constam os fragmentos de documentos utilizados nos testes e os dados coletados.

Tabela 15 – Testes de Digitalização

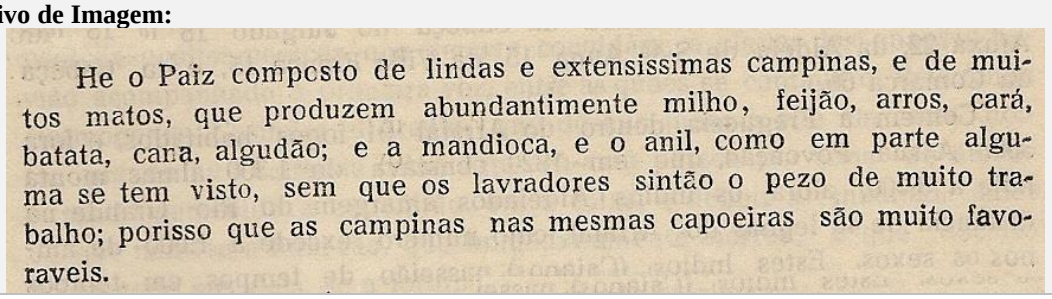
<table>
<tr><th colspan="3">Documento 1</th></tr>
<tr><td colspan="3">Arquivo de Imagem:
  </td></tr>
<tr><td colspan="3">Resultado da Digitalização Parcial (Tesseract-OCR):
Em nome dos empresários do setor de gastronomia e turismo que
trabalham no Pátio de Eventos Luíz Gonzaga durante o ciclo junino,
agradecemos ao prefeito José Queiroz e ao presidente da Fundação de
Cultura e Turismo de Caruaru, José Pereira, pela estrutura do São J oão
de 2011. Graças ao novo formato do Pátio de Eventos Luiz Gonzaga,
Caruaru ganhou uma importante arena gastronômica, modernizando e
aumentando o espaço do Pátio de Eventos.</td></tr>
<tr><th>Nros Palavras</th><th>Reconhecidas (Classificação)</th><th>Nros de Testes CAPTCHA</th></tr>
<tr><td>73</td><td>38</td><td>45</td></tr>
<tr><td colspan="3">Resultado Final da Digitalização:
Em nome dos empresários do setor de gastronomia e turismo que
trabalham no Pátio de Eventos Luíz Gonzaga durante o ciclo junino,
agradecemos ao prefeito José Queiroz e ao presidente da Fundação de
Cultura e Turismo de Caruaru, José Pereira, pela estrutura do São João
de 2011. Graças ao novo formato do Pátio de Eventos Luíz Gonzaga,
Caruaru ganhou uma importante arena gastronômica, modernizando e
aumentando o espaço do Pátio de Eventos.</td></tr>
</table>

(Continua)

(Continuação)

Documento 2

Arquivo de Imagem:



He o Paiz composto de lindas e extensissimas campinas, e de muitos matos, que produzem abundantimente milho, feijão, arros, cará, batata, cana, algudão; e a mandioca, e o anil, como em parte alguma se tem visto, sem que os lavradores sintão o pezo de muito trabalho; porisso que as campinas nas mesmas capoeiras são muito favoraveis.

Resultado da Digitalização Parcial (Tesseract-OCR):

```
He o Paiz composto de lindas e extensissimas campinas, e de mui-
tos maias, que produzem ahundantimente milho, ieu o, arros, cará,
batata, cara, algudão; e a tnandioca, e o anil, como em parte algu-
ma se tem visto, sem que os lavradores sintão o pezo de muito tras
balho; porisso que as campinas nas mesmas Capoeiras são muito favo-
raveis.
```

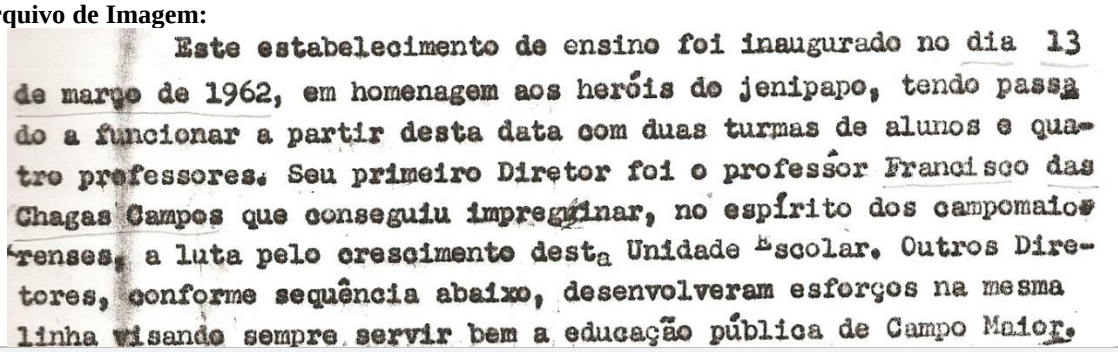
Nros Palavras	Reconhecidas (Classificação)	Nros de Testes CAPTCHA
60	0	124

Resultado Final da Digitalização:

```
He o Paiz composto de lindas e extensissimas campinas, e de mui-
tos matos, que produzem abundantimente milho, feijão, arros, cará,
 batata, cana, algudão; e a mandioca, e o anil, como em parte algu-
ma se tem visto, sem que os lavradores sintão o pezo de muito tra-
balho; porisso que as campinas nas mesmas capoeiras são muito favo-
raveis.
```

Documento 3

Arquivo de Imagem:



Este estabelecimento de ensino foi inaugurado no dia 13 de março de 1962, em homenagem aos heróis do jenipapo, tendo passado a funcionar a partir desta data com duas turmas de alunos e quatro professores. Seu primeiro Diretor foi o professor Francisco das Chagas Campos que conseguiu impregginar, no espírito dos campomaiorenses, a luta pelo crescimento desta Unidade Escolar. Outros Diretores, conforme sequência abaixo, desenvolveram esforços na mesma linha visando sempre servir bem a educação pública de Campo Maior.

Resultado da Digitalização Parcial (Tesseract-OCR):

```
 Este estabelecimento da ensino foi :Inaugurado no dia 13

_do maria da 1962, em homenagem aos heróis do jenipapo, bando passa

do a figxeionar s. partir desta data com duas turma: de almas e qua»
tre professores. seu primeim Diretor foi o profasior Francisco das
 que oonsaguiu imprpguzar, no" espírito dos oampomaioií
@uma  a luta pelo crescimento desta Unidade kscol?r. Outros Diz-a-
tores, nfçrma sequência abaixo, desenvolveram esforços na. mesma

linha ?aan?shaemprvzunervir bem a.; eduumção pública de campo !mag
```

Nros Palavras	Reconhecidas (Classificação)	Nros de Testes CAPTCHA
83	0	325

(Continua)

(Continuação)

Resultado Final da Digitalização:

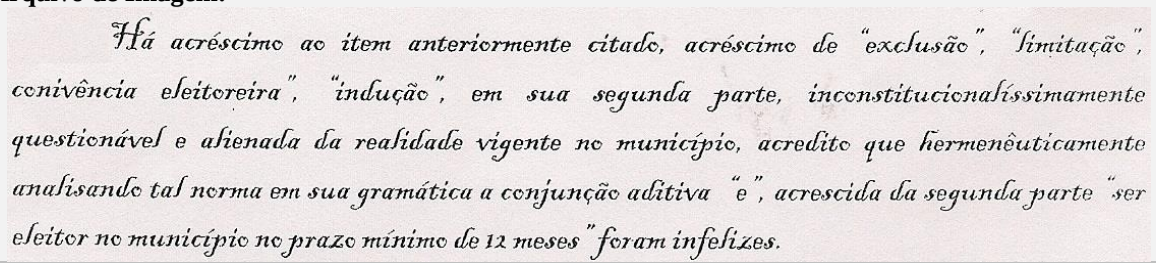
Este estabelecimento de ensino foi inaugurado no dia 13

de março de 1962, em homenagem aos heróis de jenipapo, tendo passado a funcionar a partir desta data com duas turmas de alunos e quatro professores. Seu primeiro Diretor foi o professor Francisco das Chagas Campos que conseguiu impregnar, no espírito dos campomaiorenses a luta pelo crescimento desta Unidade Escolar. Outros Diretores, conforme sequência abaixo, desenvolveram esforços na mesma

linha visando sempre servir bem a educação pública de Campo Maior.

Documento 4

Arquivo de Imagem:



Há acréscimo ao item anteriormente citado, acréscimo de "exclusão", "limitação", conivência eleitoreira", "indução", em sua segunda parte, inconstitucionalíssimamente questionável e alienada da realidade vigente no município, acredito que hermenêuticamente analisando tal norma em sua gramática a conjunção aditiva "e", acrescida da segunda parte "ser eleitor no município no prazo mínimo de 12 meses" foram infelizes.

Resultado da Digitalização Parcial (Tesseract-OCR):

#É acTéJdmc ao ítem anteriormente dtmk, aneírdnzo (Á: "iamc/umãc Íinzitelçãc,
umívêltcía e/zaítioreírañ "ímfuçãoí em .rua áwgjuntflz farta, incorutítucíonuÁÍyoímmzzenm
yuestionáve/ e u/Êenmfz Ja rea/iaíazlé wjenie no ?llunl/LÍÍQZIQ. arcreríito (lua /zêrnzertêuztícmrtontzv

unu/Ártnuãv caÍncnrza em Juajramãticu t1 conjunção ufiíva «e acrewcitãz nã: Jejuvnzúzjuzrtre . ar

e/Êaitor m7 município 71o JJTLIZO nrírtímo :1ê 11 nzeJeJ fuma íigfêÍize-.V,

Nros Palavras	Reconhecidas (Classificação)	Nros de Testes CAPTCHA
55	0	466

Resultado Final da Digitalização:

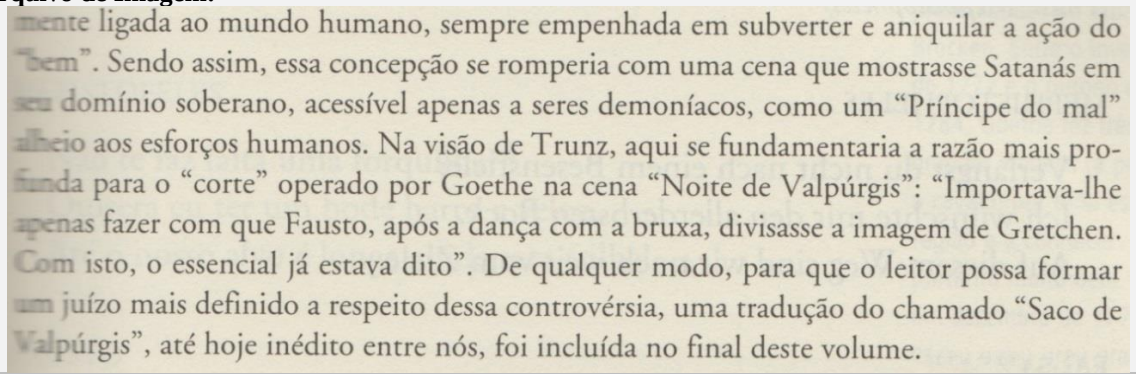
Há acréscimo ao item anteriormente citado, acréscimo de "exclusão", "limitação", conivência eleitoreira", "indução", em sua segunda parte, inconstitucionalíssimamente questionável e alienada da realidade vigente no município, acredito que hermenêuticamente

analisando tal norma em sua gramática a conjunção aditiva "e", acrescida da segunda parte "ser

eleitor no município no prazo mínimo de 12 meses" foram infelizes.

Documento 5

Arquivo de Imagem:



mente ligada ao mundo humano, sempre empenhada em subverter e aniquilar a ação do "bem". Sendo assim, essa concepção se romperia com uma cena que mostrasse Satanás em seu domínio soberano, acessível apenas a seres demoníacos, como um "Príncipe do mal" alheio aos esforços humanos. Na visão de Trunz, aqui se fundamentaria a razão mais profunda para o "corte" operado por Goethe na cena "Noite de Valpúrgis": "Importava-lhe apenas fazer com que Fausto, após a dança com a bruxa, divisasse a imagem de Gretchen. Com isto, o essencial já estava dito". De qualquer modo, para que o leitor possa formar um juízo mais definido a respeito dessa controvérsia, uma tradução do chamado "Saco de Valpúrgis", até hoje inédito entre nós, foi incluída no final deste volume.

(Continua)

(Conclusão)

Resultado da Digitalização Parcial (Tesseract-OCR):

```
_mta ligada ao mundo humano, sempre empenhada em subverter e aniquilar a ação do
12mm?. Sendo assim, essa concepção se romperia com uma cena que mostrasse Satanás em
:n domínio soberano, acessível apenas a seres demoníacos, como um "Príncipe do mal?
Elcio aos esforços humanos. Na visão de Trunz, aqui se fundamentaria a razão mais pro-
&da para o ?corte? operado por Goethe na cena ?Noite de Valpúrgis?: "importava-lhe
qxnas fazer com que Fausto, após a dança com a bruxa, divisasse a imagem de Gretchen.
Com isto, o essencial já estava dito". De qualquer modo, para que o leitor possa formar
»um ¡uizo mais definido a respeito dessa controvérsia, uma tradução do chamado "Saco de
Valpúrgis?, ate hoje inédito entre nós, foi incluída no final deste volume.
```

Nros Palavras	Reconhecidas (Classificação)	Nros de Testes CAPTCHA
127	38	224

Resultado Final da Digitalização:

```
mente ligada ao mundo humano, sempre empenhada em subverter e aniquilar a ação do
"bem". Sendo assim, essa concepção se romperia com uma cena que mostrasse Satanás em
seu domínio soberano, acessível apenas a seres demoníacos, como um "Príncipe do mal"
alheio aos esforços humanos. Na visão de Trunz, aqui se fundamentaria a razão mais pro-
funda para o "corte" operado por Goethe na cena "Noite de Valpúrgis": "Importava-lhe
apenas fazer com que Fausto, após a dança com a bruxa, divisasse a imagem de Gretchen.
Com isto, o essencial já estava dito". De qualquer modo, para que o leitor possa formar
um juízo mais definido a respeito dessa controvérsia, uma tradução do chamado "Saco de
Valpúrgis", até hoje inédito entre nós, foi incluída no final deste volume.
```

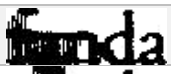
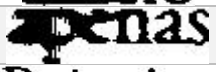
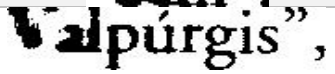
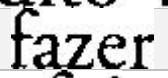
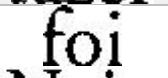
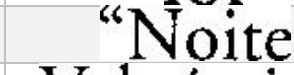
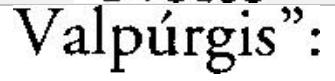
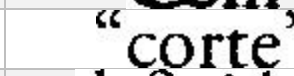
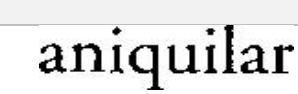
Nos dados apresentados na Tabela 15 é possível observar algumas das limitações presentes no *software* OCR. Apesar de ter conseguido digitalizar corretamente boa parte das palavras presentes na maioria dos documentos, a taxa de certeza utilizada na classificação dessas palavras ficou abaixo do especificado e por esse motivo poucas delas foram consideradas como reconhecidas. Mesmo que para um ser humano seja relativamente fácil conseguir identificar quais palavras foram digitalizadas de forma correta, para um *software* isso pode ser uma tarefa difícil. Mesmo fazendo uso de dicionários existe a possibilidade do OCR ter transformado uma palavra em outra que também exista no dicionário.

Levando-se em consideração as regras de validação do CAPTCHA abordadas na seção 5.5.3, é possível comprovar que os documentos com um maior número de erros de digitalização tiveram que passar por mais testes de CAPTCHAs. No entanto, nos documentos que apresentaram menos erros, a maior parte dos testes de CAPTCHA serviram apenas para confirmar o resultado do processamento do OCR. Ou seja, a maioria das palavras passaram apenas uma vez pelo teste de CAPTCHA.

Por fim, foram coletados os dados resultantes dos testes realizados na digitalização do Documento 5 da Tabela 15, para validação da API CAPTCHA. Nessa etapa foram selecionados apenas os dados de palavras enviadas para o reconhecimento, visto que essa é funcionalidade que está sendo testada. A Tabela 16 apresenta a ordem em que a palavra foi enviada para o teste

de CAPTCHA, o fragmento[1] de imagem da palavra, o texto resultante do processamento do OCR, a taxa de certeza, o número de testes CAPTCHAs necessários para o seu reconhecimento e o texto final, ou seja, o texto que apresentou maior recorrência nas tentativas de reconhecimento.

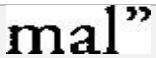
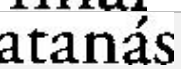
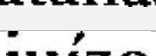
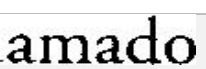
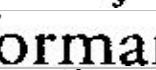
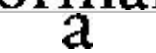
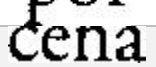
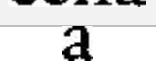
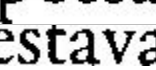
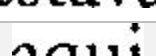
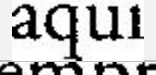
Tabela 16 – Resultado dos Testes da API CAPTCHA

Ordem	Fragmento	Texto OCR	Taxa Certeza	Nros CAPTCHAs	Texto Final
1º		&da	-11,34	10	funda
2º		Elcio	-9,41	10	alheio
3º		qxnas	-8,24	10	apenas
4º		"Príncipe	-7,94	1	“Príncipe
5º		"Saco	-7,24	1	“Saco
6º		"importava-lhe	-6,91	10	“Importava-lhe
7º		12mm?.	-6,89	10	“bem”.
8º		Valpúrgis?,	-6,65	10	Valpúrgis”,
9º		:n	-6,57	10	seu
10º		_mta	-6,35	10	mente
11º		»um	-6,34	10	um
12º		dito".	-6,16	1	dito”.
13º		fazer	-6,07	1	fazer
14º		foi	-5,89	1	foi
15º		?Noite	-5,27	10	“Noite
16º		Valpúrgis?:	-5,03	10	Valpúrgis”:
17º		Com	-4,84	1	Com
18º		?corte?	-4,76	10	“corte”
19º		definido	-4,21	1	definido
20º		aniquilar	-4,15	1	aniquilar

(Continua)

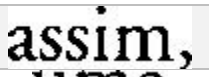
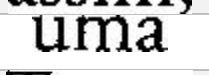
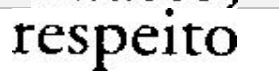
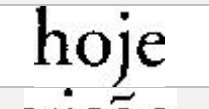
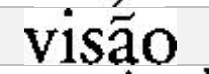
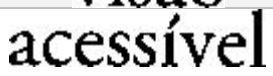
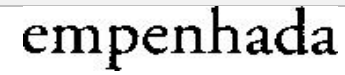
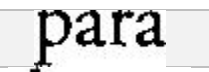
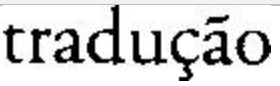
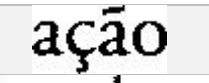
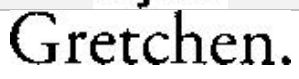
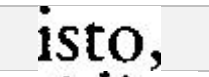
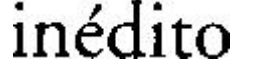
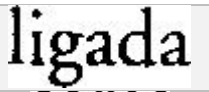
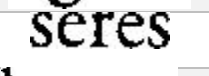
[1] A distorção foi removida para facilitar a visualização da palavra.

(Continuação)

Ordem	Fragmento	Texto OCR	Taxa Certeza	Nros CAPTCHAs	Texto Final
21º		mal?	-4,13	10	mal”
22º		final	-4,12	1	final
23º		Satanás	-4,10	1	Satanás
24º		¡uizo	-4,06	10	juízo
25º		já	-3,94	1	já
26º		fundamentaria	-3,90	1	fundamentaria
27º		chamado	-3,68	1	chamado
28º		subverter	-3,63	1	subverter
29º		cena	-3,44	1	cena
30º		ate	-3,44	10	até
31º		deste	-3,43	1	deste
32º		esforços	-3,42	1	esforços
33º		formar	-3,41	1	formar
34º		a	-3,25	1	a
35º		aos	-3,17	1	aos
36º		entre	-3,17	1	entre
37º		incluída	-3,14	1	incluída
38º		por	-3,09	1	por
39º		cena	-3,09	1	cena
40º		a	-3,06	1	a
41º		possa	-3,00	1	possa
42º		estava	-3,00	1	estava
43º		aqui	-2,99	1	aqui
44º		sempre	-2,98	1	sempre
45º		divisasse	-2,98	1	divisasse
46º		Goethe	-2,97	1	Goethe
47º		concepção	-2,95	1	conscepção

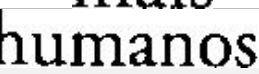
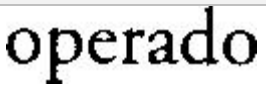
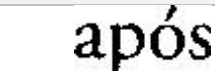
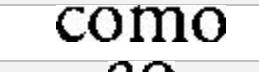
(Continua)

(Continuação)

Ordem	Fragmento	Texto OCR	Taxa Certeza	Nros CAPTCHAs	Texto Final
48º		assim,	-2,95	1	assim,
49º		uma	-2,94	1	uma
50º		Trunz,	-2,92	1	Trunz,
51º		mostrasse	-2,92	1	mostrasse
52º		Fausto,	-2,90	1	Fausto,
53º		respeito	-2,90	1	respeito
54º		controvérsia,	-2,89	1	controvérsia,
55º		hoje	-2,88	1	hoje
56º		visão	-2,78	1	visão
57º		acessível	-2,78	1	acessível
58º		soberano,	-2,78	1	soberano,
59º		empenhada	-2,74	1	empenhada
60º		para	-2,72	1	para
61º		dança	-2,68	1	dança
62º		essencial	-2,68	1	essencial
63º		tradução	-2,67	1	tradução
64º		ação	-2,66	1	ação
65º		Gretchen.	-2,66	1	Gretchen.
66º		isto,	-2,65	1	isto,
67º		inédito	-2,65	1	inédito
68º		ligada	-2,62	1	ligada
69º		seres	-2,61	1	seres
70º		bruxa,	-2,57	1	bruxa,

(Continua)

(Conclusão)

Ordem	Fragmento	Texto OCR	Taxa Certeza	Nros CAPTCHAs	Texto Final
71º		volume.	-2,56	1	volume.
72º		uma	-2,55	1	uma
73º		imagem	-2,54	1	imagem
74º		razão	-2,53	1	razão
75º		**romperia**	**-2,52**	**1**	**romperia**
76º		mais	-2,51	1	mais
77º		humanos.	-2,51	1	humanos.
78º		demoníacos,	-2,50	1	demoníacos,
79º		a	-2,47	1	a
80º		operado	-2,46	1	operado
81º		qualquer	-2,46	1	qualquer
82º		nós,	-2,39	1	nós,
83º		se	-2,39	1	se
84º		após	-2,39	1	após
85º		essa	-2,39	1	essa
86º		apenas	-2,38	1	apenas
87º		humano,	-2,36	1	humano,
88º		como	-2,32	1	como
89º		ao	-2,27	1	ao

Em um primeiro momento, analisando os dados da Tabela 16 é possível observar que as palavras foram enviadas para o reconhecimento de forma ascendente, de acordo com a taxa de certeza, ou seja, da menor para maior. Isso comprova que a API CAPTCHA está selecionando as palavras de forma correta. Além disso é possível verificar que as palavras que

OCR reconheceu corretamente, necessitaram de apenas um teste CAPTCHA. Por outro lado, as palavras reconhecidas incorretamente necessitaram passar mais vezes pelos testes.

Conforme apresentado na Tabela 16, no primeiro teste realizado foi exibida a palavra "funda", que possui a menor taxa de certeza entre todas as outras listadas. Após a realização do teste, a aplicação verificou, na base de dados, que o texto digitado pelo usuário não correspondia com o texto gerado pelo processamento do OCR ("&da"). A aplicação então gravou na base de dados essa tentativa de reconhecimento, realizada pelo usuário, e enviou a mesma palavra outras nove vezes para o teste de CAPTCHA. Após o último teste a aplicação selecionou o texto com maior recorrência entre todas as dez tentativas de reconhecimento e definiu-o como o texto final. A palavra então foi assinalada como reconhecida e a aplicação enviou a próxima para o reconhecimento, esse processo se repetiu até que todas as palavras passassem pelo teste. No caso do texto digitado pelo usuário corresponder ao texto resultante do processamento do OCR, a palavra é imediatamente definida como reconhecida.

Com os resultados obtidos nos casos de testes realizados, é possível verificar que o protótipo atingiu os objetivos propostos no início desse trabalho. Primeiramente, os casos de testes se concentraram nas funcionalidades básicas do protótipo, como: cadastro e acesso de usuários, *upload* de arquivos e API CAPTCHA. Em um segundo momento, os casos de testes focaram naquilo em que se propôs o protótipo: a digitalização de acervos fazendo uso de um *software* OCR em conjunto com uma ferramenta baseada em *crowdsourcing*. Nesses testes foi possível observar que, com o auxílio da API CAPTCHA, foi possível minimizar ou até eliminar as falhas resultantes do processamento do OCR.

7 CONCLUSÃO

Esse estudo aprofundou o conhecimento sobre o *crowdsourcing* e como o seu uso pode possibilitar a resolução de problemas computacionais abertos. O problema que foi abordado neste trabalho refere-se a uma limitação presente nos *softwares* de reconhecimento ótico de caracteres. Nos testes e estudos realizados, os *softwares* OCR se mostraram imprecisos e muitas vezes não reconheceram boa parte das palavras presentes em um arquivo de imagem. Destacou-se também a importância da digitalização de acervos, e como fazer isso atualmente, sem o auxílio de um *software* OCR, pode ser uma tarefa trabalhosa e cara. O resultado final deste foi o protótipo de uma aplicação *web*, para a digitalização de acervos, que trabalha em conjunto com uma série de ferramentas, como o *software* Tesseract-OCR e uma API CAPTCHA, cada qual com uma função pré-definida, conforme explicado na seção 5.

A API CAPTCHA atua como ferramenta de *crowdsourcing* para o reconhecimento das palavras não identificadas pelo Tesseract-OCR. Para que isso seja possível, foi desenvolvida uma arquitetura onde essas duas ferramentas distintas trabalham em conjunto. No decorrer do desenvolvimento, percebeu-se que seria necessário fazer alterações no código fonte do Tesseract-OCR, a fim de coletar os dados resultantes do processamento dos arquivos e persisti-los em uma base de dados. Esses dados são utilizados na classificação das palavras como reconhecidas e não reconhecidas para posteriormente serem consumidos pela API CAPTCHA.

Tendo em vista que adesão, ou mudança, de qualquer API CAPTCHA, pelos *websites*, pode levar algum tempo, não foi possível realizar testes em larga escala, mas isso não interferiu na validação da solução proposta. Os casos de testes realizados com a API CAPTCHA para o reconhecimento das palavras sugerem a viabilidade da utilização da mesma. Em um cenário ideal, onde milhares de *websites* adotassem a API CAPTCHA em questão, teríamos milhões de palavras reconhecidas diariamente, tornando essa uma poderosa ferramenta de *crowdsourcing*.

Não há requisitos mínimos de *hardware* para que usuários possam utilizar o protótipo para a digitalização de seus arquivos, visto que todo esse processamento fica a cargo do servidor. No entanto há algumas considerações principalmente no que tange ao *layout* do acervo e a qualidade dos arquivos de imagem. Conforme mencionado na seção anterior, o Tesseract-OCR possui algumas limitações ao processar documentos com *layouts* complexos, e isso se revelou como uma limitação do protótipo. A qualidade da imagem também pode prejudicar o processamento do *software* OCR, afetando também a API CAPTCHA, que receberá fragmentos

de imagens muito pequenos, fazendo com que os usuários tenham dificuldade ou simplesmente não consigam reconhecer qual palavra está presente no fragmento visualizado.

Analisando as considerações abordadas, o protótipo necessitaria passar por algumas melhorias antes de poder ser distribuído em larga escala, tais como: a criação de uma interface onde o usuário possa fazer a seleção dos blocos de texto que deseja digitalizar e um *script* que valide a qualidade mínima que uma imagem deve conter para poder ser processada. A API CAPTCHA também poderia ser aprimorada com a inclusão de algumas variáveis de controle, como: quantos *websites* estão fazendo o seu uso, quanto tempo o usuário leva para resolver um CAPTCHA, entre outras.

Existe também a possibilidade de substituir o Tesseract-OCR por outros *softwares* OCR *open source* ou proprietários. Também seria possível fazer com que dois ou mais *softwares* OCR trabalhassem em conjunto, usufruindo-se das melhores características que cada um disponibiliza. No primeiro caso, as alterações no protótipo seriam pontuais e simples. No entanto, no segundo caso, onde mais *softwares* trabalhariam em conjunto, seriam necessárias alterações mais profundas na estruturação e funcionamento da aplicação.

A tendência da utilização de sistemas baseados em *crowdsourcing* tende a crescer nos próximos anos. Por isso, como projetos futuros pode-se destacar o desenvolvimento de sistemas baseados em *crowdsourcing*, que auxiliem na resolução de problemas computacionais abertos, ou seja, problemas que os computadores são incapazes de resolver sozinhos.

Seguindo a linha do problema aqui proposto, esse trabalho pode auxiliar no desenvolvimento de sistemas similares, que sejam voltados para línguas que fazem uso de alfabetos diferentes do ocidental, como árabe, mandarim, russo, entre outras. Aproveitando também esse mesmo conceito, poderiam ser desenvolvidos tipos de CAPTCHAs sonoros que auxiliem na transcrição de áudios e vídeos.

Há também outras formas de explorar o poder do *crowdsourcing*, o CAPTCHA é apenas uma delas. Essas formas podem variar desde jogos eletrônicos, que auxiliam na descoberta de novos tipos de proteínas, a portais de aprendizagem de idiomas, onde os usuários contribuem na tradução de páginas da Internet.

REFERÊNCIAS

ANH, L. V. **Human Computation**. 87 f. Tese (Doutorado). School of Computer Science, Carnegie Mellon University, USA, 2005. Disponível em: <http://reports-archive.adm.cs.cmu.edu/anon/2005/CMU-CS-05-193.pdf>. Acesso em: 20 Mar. 2014.

ANH, L. V. **TED:** Colaboração On-line em Escala Massiva. TEDxCMU, 2011. Disponível em: < http://goo.gl/T1oSpc>. Acesso em: 22 Mar. 2014.

BRABHAM, D. **Crowdsourcing as a Model for Problem Solving:** An Introduction and Cases. 16 f. Convergence: The International Journal of Research into New Media Technologies, University of Utah, USA, 2008. Disponível em: <http://www.clickadvisor.com/downloads/Brabham_Crowdsourcing_Problem_Solving.pdf>. Acesso em: 21 Mar. 2014.

BREUEL, T. M. **Two geometric algorithms for layout analysis**. In: DAS '02: Proceedings of the 5th International Workshop on Document Analysis Systems V. London, UK: Springer-Verlag, 2002. Disponível em: < http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=EC14F8987259040EA0DAD0AD36A45B05?doi=10.1.1.17.5758&rep=rep1&type=pdf >. Acesso em: 10 de jun. 2014.

CHERIET, M. C**haracter recognition systems: a guide for students and practitioners**. USA: Wiley, 2007. Disponível em: <http://people.mokk.bme.hu/~kornai/OCR/Irodalom/Cheriet_Character_Recognition_Systems__A_Guide_for_Students_and_Practitioners.pdf >. Acesso em: 20 Mai. 2014.

CHIU C., LIANG T., TURBAN E. **What can crowdsourcing do for decision support?** Taiwan, 2014.

CONARQ, Recomendações para Digitalização de Documentos Arquivísticos Permanentes, 2010. Disponível em: <http://www.conarq.arquivonacional.gov.br/media/publicacoes/recomenda/recomendaes_para_digitalizao.pdf>. Acesso em: 25 Mar. 2014.

COSTA, R. A. **Uma Abordagem para Utilização de CAPTCHAs Clicáveis para Combater a *Click Fraud***. 2010. 137 p. Dissertação de Mestrado em Ciência da Computação. Universidade Federal de Pernambuco. Recife, Pernambuco.

DOAN, A., Ramakrishnan, R., Haleyy, A. Y. **Crowdsourcing Systems on the World-Wide Web**, In: Communications of the ACM, CACM, 54 (4), 2011.

DONELAN, H., KEAR, K., RAMAGE, M. **On-line Communication and Collaboration:** A Reader. 1 ed. EUA: Routledge, 2010.

EIKVIL, L. **OCR Optical Character Recognition**. Noruega, 1993. Disponível em: <http://www.nr.no/~eikvil/OCR.pdf >. Acesso em: 15 Mai. 2014.

ESTELLÉS, A.; GONZÁLES, E. Ladrón-de-Guevara, F. 14 f. **Towards an integrated crowdsourcing definition**. Journal of Information Science, Technical University of Valencia, Espanha, 2012. Disponível em: < http://goo.gl/xqKHRs >. Acesso em: 20 Abr. 2014.

GATAUTISA R., VITKAUSKAITEA E. **Crowdsourcing application in marketing activities**. Kaunas University of Technology, Lituânia, 2013.

GOCR. 2014. Disponível em: <http://jocr.sourceforge.net/>. Acesso em: 01 Jun. 2014.

GOOGLE-TESSERACT. 2014. Disponível em: <http://code.google.com/p/tesseract-ocr/>. Acesso em: 30 Mai. 2014.

HOWE, J. **O poder das multidões: por que a força da coletividade está remodelando o futuro dos negócios**. Tradução de Alessandra Mussi Araujo. 1 ed. Rio de Janeiro: Campus/Elsevier, 2008.

ICT, Data and Statistics Division. The World in 2013: ICT Facts and Figures. IUT. 03, 2013. Disponível em: <http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx>. Acesso em: 04 Abr. 2014.

PHP-TESSERACT. 2014. Disponível em: <https://code.google.com/p/php-tesseract>. Acesso em: 30 Mai. 2014.

QUINN, A. J.; BEDERSON, B.B. **Human computation: a survey and taxonomy of a growing field**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. University of Maryland, USA, 2011. Disponível em: < http://alexquinn.org/papers/Human%20Computation,%20A%20Survey%20and%20Taxonomy%20of%20a%20Growing%20Field%20%28CHI%202011%29.pdf>. Acesso em 25 Mai. 2014.

ROMANI, R. **Jogos com Propósito e Construção de Conhecimento em *Design***. 2013. 138 p. Tese de Doutorado em Ciência da Computação. Universidade Estadual de Campinas, Instituto de Computação. Campinas, São Paulo.

SCHMITZ, D. **A evolução do MVC para REST**. 2012. Disponível em: <http://imasters.com.br/gerencia-de-ti/tendencias/a-evolucao-do-mvc-para-rest/>. Acesso em: 02 Nov. 2014.

SELINGER, P. **Review of Linux OCR software**. Department of Mathematics and Statistics, Dalhousie University. Halifax, Canada, 2007. Disponível em: <http://www.mathstat.dal.ca/~selinger/ocr-test/>. Acesso em 05 Out. 2014.

SILVA, B. N. **KA-CAPTCHA: An Opportunity for Knowledge Acquisition on the Web**. 2007. 62 p. Dissertação de Mestrado em Computação. Universidade Federal Fluminense. Niterói, Rio de Janeiro.

SMITH. 2007. **An Overview of the Tesseract OCR Engine**. Disponível em: <http://code.google.com/p/tesseract-ocr/>. Acesso em 30 Mai. 2014.

OCROPUS. 2014. Disponível em: <http://www.ocropus.org>. Acesso em: 01 Jun. 2014.

WANG, Z.; LU, Y.; TAN, C. L. **Word extraction using area Voronoi diagram**. In: 2003 Conference on Computer Vision and Pattern Recognition Workshop. Wisconsin, EUA: cvprw, 2003.

XINTONG G., HONGZHI W., SONG Y., HONG G. **Brief survey of crowdsourcing for data mining**. School of Computer Science and Technology, Harbin Institute of Technology, China, 2014.