

WARD: Uma Aplicação de Reconhecimento e Descrição de Imagens para Deficientes Visuais

Lucas Izidorio Almeida¹, Rodrigo Gonçalves Ribeiro¹, Maria Inês Lage de Paula¹,
Lesandro Ponciano dos Santos¹

¹PUC Minas em Contagem

{izidorio.lucas7, roxdriago1883}@gmail.com, {milpaula, lesandrop}@pucminas.br

Resumo. *Muitos brasileiros são deficientes visuais. Em um mundo cada vez mais tecnológico, a inclusão de pessoas com problemas na visão se tornou um tópico importante a ser discutido. A experiência do usuário deve ser pensada para que todos possam utilizar os sistemas de informação, principalmente os sistemas em que os usuários são criadores de conteúdo, como em redes sociais. No contexto de imagens geradas e publicadas pelos usuários do sistema, a geração de descrição do conteúdo das imagens se torna ainda mais desafiadora. Este trabalho tem como objetivo desenvolver uma extensão para browser capaz de identificar imagens na Web e criar descrições acessíveis, para que os usuários com problemas na visão tenham acesso à informação sobre o conteúdo que é exibido dentro da imagem.*

1. Introdução

A deficiência visual é um problema que acomete milhares de brasileiros. Segundo as informações do último censo demográfico realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), o Brasil possuía, em 2010, cerca de 6,5 milhões de pessoas deficientes visuais, sendo 528 mil dessas o número aproximado de indivíduos com ausência total da visão [de Geografia e Estatística IBGE 2011]. Tais pessoas, entretanto, acabam enfrentando dificuldades ao utilizar os sistemas que estão disponíveis na Web, principalmente quando se trata de redes sociais. Apesar de existirem funcionalidades de acessibilidade em algumas redes sociais, a maior parte do conteúdo visual publicado é feito pelos usuários e fica sem uma descrição acessível.

Existem softwares de acessibilidade para sistemas operacionais que auxiliam no uso de computadores pelo público que possui alguma deficiência visual. Dentre tais softwares, podemos citar ferramentas que fazem a leitura da tela para o usuário, como o *Non Visual Desktop Access* (NVDA)¹. Entretanto, os leitores de tela não permitem que o usuário compreenda o contexto completo quando o conteúdo apresenta imagens. Esse problema pode ser contornado por meio de descrições das imagens, que podem ser providas pelo desenvolvedor do site durante sua criação.

Apesar da existência do atributo *alt* (texto alternativo) em páginas Web, muitos sites não o utilizam nas imagens e, quando falamos de redes sociais, o assunto fica ainda mais complexo. A maior parte dos usuários não sabem como criar uma descrição acessível ou simplesmente não pensam no público com deficiência na hora de fazer uma postagem.

¹NV Access - Acesso disponível em: <https://www.nvaccess.org/>. Último acesso em 12 de setembro de 2021.

Ao pensar na experiência do usuário durante a utilização dos sistemas de informação, é necessário considerar quais são os requisitos de acessibilidade para aumentar o público da aplicação. Mas e quando os desenvolvedores não são os únicos responsáveis pela acessibilidade? Como melhorar a experiência dos usuários com deficiência visual?

Grande parte do conteúdo das redes sociais mais utilizadas atualmente atrela fortemente a descrição (ou o texto da postagem) com alguma imagem associada (ou até mesmo mais de uma imagem). Para tornar a experiência de utilização das aplicações com mídia visual acessível é necessário prover descrições alternativas para as imagens.

O objetivo geral deste trabalho é desenvolver um sistema para funcionar em conjunto com o *browser* do usuário, denominado Web-Assistente de Reconhecimento Digital de Imagens (WARD), que seja capaz de acrescentar textos alternativos às imagens na tela, tornando-as acessíveis às pessoas que possuem alguma deficiência visual.

Para atingir o objetivo principal do trabalho é necessário cumprir alguns objetivos específicos, entre eles:

- Analisar as ferramentas atuais de acessibilidade para pessoas com deficiência visual e levantar as principais funcionalidades em comum entre elas.
- Criar uma extensão para *browser* capaz de ler e modificar o arquivo de Linguagem de Marcação de Hipertexto (HTML, do inglês *Hypertext Markup Language*) de uma página Web.
- Criar uma Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*) para ser consumida pela extensão capaz de gerar uma descrição para uma determinada imagem de entrada.
- Criar um algoritmo de descrição de imagens capaz de gerar descrições acessíveis.

Este trabalho está estruturado em 6 seções. A Seção 2 deste documento contém os fundamentos teóricos utilizados no trabalho. Na Seção 3, são apresentados os trabalhos relacionados. Em seguida, apresenta-se a metodologia utilizada no projeto, na Seção 4. A Seção 5 contempla a especificação, projeto e implementação do sistema. As conclusões sobre o trabalho estão na Seção 6.

2. Referencial Teórico

Nesta seção são descritos os principais conceitos que estão relacionados a sistemas de acessibilidade e técnicas que serão utilizadas no trabalho.

2.1. Acessibilidade para Deficientes Visuais

Quando pensamos em acessibilidade na Web, não se trata de um assunto completamente novo. Na verdade, é uma questão que vem sendo estudada há muitos anos, e uma das principais diretrizes a ser considerada nesse assunto é a *Web Content Accessibility Guidelines* (WCAG), um conjunto de recomendações a respeito de acessibilidade na Web que foi publicada em maio de 1999 pela *World Wide Web Consortium* (W3C), principal organização de padronização da internet.

Dentre as recomendações da WCAG, uma das principais é referente aos textos alternativos para conteúdo não-textual. De acordo com a W3C, é necessário que os textos alternativos estejam ligados através do código-fonte ao conteúdo não-textual de forma que

as tecnologias assistivas sejam capazes de lê-los e usá-los conforme o necessário (W3C, 2018, tradução nossa)².

Textos alternativos podem ser inseridos em imagens na página HTML através do atributo *alt*, e geralmente é esse o atributo analisado pelos leitores de tela para descrever a imagem para o usuário. Entretanto, nem sempre fica nas mãos do programador prover uma descrição para a imagem. Em redes sociais, onde o conteúdo é publicado por outros usuários e não pelo próprio programador, por exemplo, a falta de textos alternativos se torna um problema comum. A partir disso, existem algumas soluções que já foram criadas na Web.

O Twitter³ e o Facebook⁴ disponibilizam uma funcionalidade para que os usuários adicionem uma descrição de texto alternativo às imagens anexadas na postagem. Além disso, também existe um movimento conhecido como “#PraCegoVer”⁵ (ou “#PraTodos-Verem”) nas redes sociais que utilizam *hashtags* que consiste em trazer uma descrição textual para postagens que têm imagens anexadas, provida pelo próprio autor da publicação. As publicações feitas utilizando alguma das *hashtags* são agrupadas dentro da rede social.

[Sacramento et al. 2020] pontuam que, apesar das funcionalidades existentes nessas plataformas, muitos usuários não as usam por não conseguir encontrá-las ou por não saber o que escrever. Além disso, também falam sobre a falta dessas funcionalidades em outros sites e redes sociais, como o WhatsApp e o YouTube. [Gleason et al. 2019] reforçam essa ideia em sua pesquisa, na qual em uma amostra de 9,22 milhões de tuítes, 1,09 milhões possuíam fotos anexadas e apenas 0,1% destes possuíam textos alternativos criados com a funcionalidade provida pela rede social.

2.2. Ferramentas Assistivas, Identificação e Descrição de Imagens

Para atingir os objetivos do trabalho, é necessário utilizar tecnologias e técnicas já existentes para criar textos alternativos. Do ponto de vista técnico, a identificação de imagens se dá através da leitura do documento HTML de uma página Web em busca de imagens (pela *tag img*), enquanto a descrição de imagens está relacionada ao conceito de visão computacional e Inteligências Artificiais (IAs) capazes de identificar quais são os elementos presentes em uma determinada imagem.

O trabalho de [Gleason et al. 2020] apresenta uma extensão para *browser* que identifica imagens no conteúdo HTML e extrai o *Uniform Resource Locator* (URL) para fazer uma requisição no servidor *backend*, sem a necessidade de uma ação do usuário. A resposta da requisição é o texto alternativo adequado e, após recebê-la, a extensão altera o atributo *alt* na página para possibilitar a leitura da descrição pelas ferramentas assistivas. [Gleason et al. 2019] também especificam processos de descrição de imagens por pessoas e por robôs. Podemos citar os métodos de descrição de cores, determinação da

²No original: In order for people with disabilities to be able to use this text - the text must be “programmatically determinable” - This means that the text must be able to be read and used by the assistive technologies (and the accessibility features in browsers) that people with disabilities use.

³Twitter. It’s what’s happening. - Acesso disponível em: <https://twitter.com/>. Último acesso em 12 de setembro de 2021.

⁴Facebook - Log In or Sign Up - Acesso disponível em: <https://facebook.com/>. Último acesso em 12 de setembro de 2021.

⁵Pra Cego Ver - Home — Facebook - Acesso disponível em: <https://www.facebook.com/PraCegoVer/>. Último acesso em 20 abr. 2021.

importância de objetos na imagem, fundo e outros conteúdos que o usuário talvez não seja capaz de perceber somente através do texto original (tradução nossa)⁶.

2.3. Algoritmos e Tecnologias de Descrição de Imagens

Existem diversas IAs disponíveis atualmente que fazem a identificação do conteúdo de uma imagem. A principal ferramenta que é utilizada neste trabalho para fazer o reconhecimento de imagens é o Google Cloud Vision API⁷.

A criação de uma extensão de *browser* exige a utilização de HTML, Folha de Estilo em Cascatas (CSS, do inglês *Cascading Style Sheets*) e JavaScript. A lógica necessária para a extensão é bem simples, pois só é necessário fazer a manipulação do HTML da página Web que está sendo acessada pelo usuário. A parte principal da aplicação é feita através de um servidor *backend* utilizando Node.js⁸.

Além disso, também existe o desafio da tradução da descrição do inglês para o português, já que a aplicação construída neste trabalho é voltada para o público brasileiro e todas as ferramentas utilizadas utilizam o inglês como língua base. Para fazer a tradução, a aplicação desenvolvida neste trabalho utiliza o Google Cloud Translation API⁹.

3. Trabalhos Relacionados

[Gleason et al. 2020] propôs uma extensão para *browser* capaz de ler e editar o HTML do Twitter. Sempre que uma nova imagem é carregada, a extensão extrai a URL da imagem e faz uma requisição no servidor *backend* da aplicação, onde a imagem é processada para gerar o texto alternativo adequado. A resposta da requisição com o texto alternativo é enviada de volta para a extensão, que edita o HTML do Twitter novamente para inserir o texto na propriedade *alt* da imagem processada. A extensão Twitter A11Y, no entanto, funciona apenas no Twitter, diferente da proposta deste trabalho.

[Guinness et al. 2018] desenvolveu um *plugin* para *browser* que funciona de maneira semelhante ao Twitter A11Y, identificando as imagens na página HTML para inserir textos alternativos na propriedade *alt*. Todavia, o Caption Crawler utiliza o mecanismo de pesquisa reversa do Google¹⁰ para encontrar correspondências da imagem e reutilizar descrições que foram usadas anteriormente. O trabalho não é capaz de inserir descrições em qualquer imagem, pois é necessário que ela tenha sido descrita previamente por outra pessoa ou mecanismo, diferente da proposta deste trabalho.

[Bodi et al. 2021] apresentam uma ferramenta para auxiliar deficientes visuais a assistirem vídeos com dois robôs diferentes: *NarrationBot*, um robô que descreve cenas nos vídeos a cada nova cena, e *InfoBot*, um robô que responde perguntas específicas

⁶No original: One participant mentioned describing the colors that appeared in the image. Others described determining the importance of objects, background, and other content in the image that the reader may not be able to ascertain from the main text of the tweet.

⁷Vision AI — Derive Image Insights via ML ... - Acesso disponível em: <https://cloud.google.com/vision>. Último acesso em 20 abr. 2021.

⁸Node.js - Acesso disponível em: <https://nodejs.org/>. Último acesso em 20 abr. 2021.

⁹Translation API Basic - Google Cloud - Acesso disponível em: <https://cloud.google.com/translate>. Último acesso em 12 de setembro de 2021.

¹⁰Google - Acesso disponível em: <https://www.google.com.br/>. Último acesso em 26 de setembro de 2021.

sobre a cena quando solicitado. Para fazer a descrição de cenas, a aplicação desenvolvida pelos autores escolhe o melhor frame do vídeo entre os frames repetidos onde a cena muda e analisa os elementos mais importantes da imagem, processo semelhante ao que é executado pela aplicação desenvolvida neste trabalho.

4. Metodologia

Este trabalho tem como objetivo desenvolver uma extensão para o *browser* Google Chrome¹¹ que seja capaz de ler o HTML das páginas acessadas pelo usuário e identificar imagens, criar descrições alternativas para elas através da API de descrição de imagens e editar o HTML da página inserindo as descrições na propriedade *alt* das imagens.

4.1. Etapas do Trabalho

Os seguintes processos serão realizados para o desenvolvimento do trabalho:

- I. Especificação
- II. Projeto do Sistema
- III. Implementação
- IV. Validação

Na etapa de especificação, as principais funcionalidades e regras de negócio são identificadas e representadas. Os principais meios de obtenção dos requisitos são a análise de sistemas relacionados e pesquisas sobre o funcionamento de ferramentas assistivas para deficientes visuais. Ainda na etapa, o diagrama de caso de uso e o diagrama de atividades foram desenvolvidos, com auxílio da ferramenta Lucidchart¹².

Na etapa de projeto do sistema, foram criados os diagramas de pacotes e de implantação, todos com o objetivo de nortear o desenvolvimento do sistema. O diagrama de pacotes detalha como o sistema foi dividido e suas respectivas interações. Já o diagrama de implantação mostra a maneira de execução do sistema, incluindo o meio pelo qual os componentes se comunicam.

Já na etapa de implementação, todos os algoritmos necessários para entregar as funcionalidades previstas nos diagramas são implementados e testados. As informações a respeito do ambiente de desenvolvimento são detalhadas na Seção 4.2. A implementação foi dividida entre os dois principais componentes do sistema: a API WARD, responsável por criar a descrição das imagens e tradução das descrições através das respectivas APIs do Google Cloud utilizadas no projeto, e a Extensão WARD, responsável por identificar novas imagens em uma página Web, enviá-las para a API e editar o HTML da página para inserir a propriedade *alt* na imagem de acordo com a descrição retornada pela API WARD.

Para a etapa de validação, foram feitos testes para avaliar a integração entre ferramentas assistivas de leitura de tela e o sistema WARD, a assertividade das descrições e a qualidade das descrições. Para a descrição ser considerada assertiva, é necessário que os elementos identificados estejam de fato presentes na imagem. Para definir o nível de qualidade da descrição, foi usado o critério de facilidade de identificação do conteúdo da imagem apenas com a descrição.

¹¹Google Chrome - Download the Fast, Secure Browser from Google - Acesso disponível em: <https://www.google.com/chrome/>. Último acesso em 26 de setembro de 2021.

¹²Lucidchart: Intelligent Diagramming - Acesso disponível em: <https://www.lucidchart.com/>. Último acesso em 26 de setembro de 2021.

4.2. Implementação

O Node.js é um *framework* de alta escalabilidade na qual utiliza o JavaScript como linguagem padrão em seu ambiente de execução e foi utilizado como escolha para o desenvolvimento da API do sistema WARD. A escolha do Node.js se deve a diversas vantagens de uso da aplicação, como a flexibilidade de utilizar o *npm*¹³ para gerenciar pacotes, além de ser o maior repositório de softwares do mundo, a alta escalabilidade e a facilidade de uso devido a linguagem de programação ser JavaScript, que apresenta uma quantidade enorme de materiais auxiliares para a implementação da mesma no sistema.

O JavaScript, HTML e o CSS juntos são conhecidos por constituir a estrutura de grande partes das páginas Web da internet, fato que não diferencia muito das extensões de navegadores. No caso deste trabalho, uma extensão do navegador Google Chrome é o escolhido para consumir a API do sistema, já que a criação é consideravelmente fácil e é possível contar com um próprio tutorial¹⁴ de como fazer fornecido pelo próprio Google.

O Visual Studio Code¹⁵ é um editor de código multiplataforma da Microsoft com a finalidade de desenvolvimento de aplicações Web e ele foi escolhido para o desenvolvimento tanto quanto da API utilizando JavaScript no Node.js, quanto na criação da extensão no navegador Google Chrome que utiliza JavaScript, HTML e CSS que são linguagens que possuem grande suporte nessa ferramenta.

Para controle de versão de código, foi utilizado o sistema Git¹⁶, junto a plataforma GitHub¹⁷, na qual todas as mudanças e atualizações de códigos serão armazenadas. A Tabela 1 demonstra as funcionalidades utilizadas no trabalho com suas respectivas tecnologias.

O Kanban foi o *framework* para gerenciamento do projeto escolhido para o processo de desenvolvimento do sistema WARD, devido a sua simplicidade de uso e suporte na ferramenta Trello¹⁸. As tarefas foram criadas com base nas *issues* do repositório do projeto no GitHub, que foram criadas ao longo do desenvolvimento do projeto e divididas em três principais entregas (*milestones*): v0.1, v0.5 e v1.0, sendo a última a entrega da primeira versão funcional do sistema por completo.

Para gerenciar as versões do projeto, foi criado um repositório no GitHub com duas pastas de controle: uma para a extensão e outra para a API. O fluxo de trabalho escolhido foi o Trunk-based Development¹⁹, um método onde os desenvolvedores trabalham exclusivamente no Trunk principal (*main*), devido a facilidade de uso para projetos em início e times pequenos de desenvolvimento.

¹³npm - Acesso disponível em: <https://www.npmjs.com/>. Último acesso em 26 de setembro de 2021.

¹⁴Criar e publicar extensões e apps personalizados do Chrome - Acesso disponível em: <https://support.google.com/chrome/a/answer/2714278?hl=pt-BR>. Último acesso em 26 de setembro de 2021.

¹⁵Visual Studio Code - Code Editing. Redefined - Acesso disponível em: <https://code.visualstudio.com/>. Último acesso em 26 de setembro de 2021.

¹⁶Git SCM - Acesso disponível em: <https://git-scm.com/>. Último acesso em 26 de setembro de 2021.

¹⁷GitHub: Where the world builds software · GitHub - Acesso disponível em: <https://github.com/>. Último acesso em 26 de setembro de 2021.

¹⁸Trello - Acesso disponível em: <https://trello.com/>. Último acesso em 26 de setembro de 2021.

¹⁹Trunk Based Development - Acesso disponível em: <https://trunkbaseddevelopment.com/>. Último acesso em 26 de setembro de 2021.

A API WARD foi desenvolvida com o *framework* Node.js na versão v14.15.0. A extensão foi feita utilizando o JavaScript na versão ES2015, a linguagem de marcação HTML5 e a linguagem CSS3. Além disso, o ambiente de desenvolvimento e testes foi o sistema operacional Windows 10 na versão 10.0.19042 *Build* 19042. Para validar a integração das ferramentas assistivas com o sistema, foi utilizado o leitor de tela NVDA na versão 2021.2.

5. Resultados

Esta seção contempla a especificação do sistema através do levantamento e modelagem dos requisitos, arquitetura definida pelo projeto nos diagramas desenvolvidos e considerações relevantes referentes à etapa final de desenvolvimento do projeto.

5.1. Especificação

O processo de levantamento de requisitos foi feito a partir da análise de sistemas similares e pesquisas sobre o contexto de uso de ferramentas assistivas por deficientes visuais. Através do processo, foram definidos os principais requisitos para desenvolver o Mínimo Produto Viável (MVP, do inglês *Minimum Viable Product*). O sistema WARD é composto por duas camadas que operam em conjunto: uma extensão para *browser* feita para o navegador Google Chrome capaz de identificar imagens em uma página Web e uma API responsável pelo processo de descrição de imagens e tradução de descrições.

A Figura 1 apresenta o diagrama de casos de uso (*use case*) do sistema WARD. Através dele é possível identificar os principais atores, que são o próprio usuário, o leitor de tela e as APIs de terceiros, além de identificar suas interações. O diagrama foi criado no padrão da Linguagem de Modelagem Unificada (UML, do inglês *Unified Modeling Language*).

O diagrama de atividades da UML (Figura 2) foi desenvolvido para demonstrar o fluxo de controle das atividades do sistema. O fluxo começa na primeira raia, onde o usuário, ao navegar por páginas Web com um leitor de tela (segunda raia), ativa a extensão WARD, que lê e absorve o conteúdo HTML da página (terceira raia), filtrando todas as *tags* HTML *img* com atributos *alts* vazios e enviando a URL da imagem para a API do sistema WARD (quarta raia). A API então envia a URL para a API Google Cloud Vision, que interpreta a imagem e gera elementos identificados na imagem e os manda de volta para a API WARD. Ao receber os elementos identificados da imagem, são enviados para a API Google Cloud Translate (quinta e última raia) que traduz os elementos e manda de volta como resposta. Com os elementos traduzidos, a API WARD gera uma descrição coerente e envia de volta para a extensão, que por sua vez altera o atributo *alt* das imagens da página Web.

5.2. Projeto do Sistema

No projeto de sistema, são apresentados os diagramas de atividades, implantação e pacotes.

5.2.1. Diagrama de Implantação

O diagrama de implantação da UML (Figura 3) foi desenvolvido para demonstrar como a arquitetura do sistema WARD é executada. A extensão WARD é desenvolvida para o

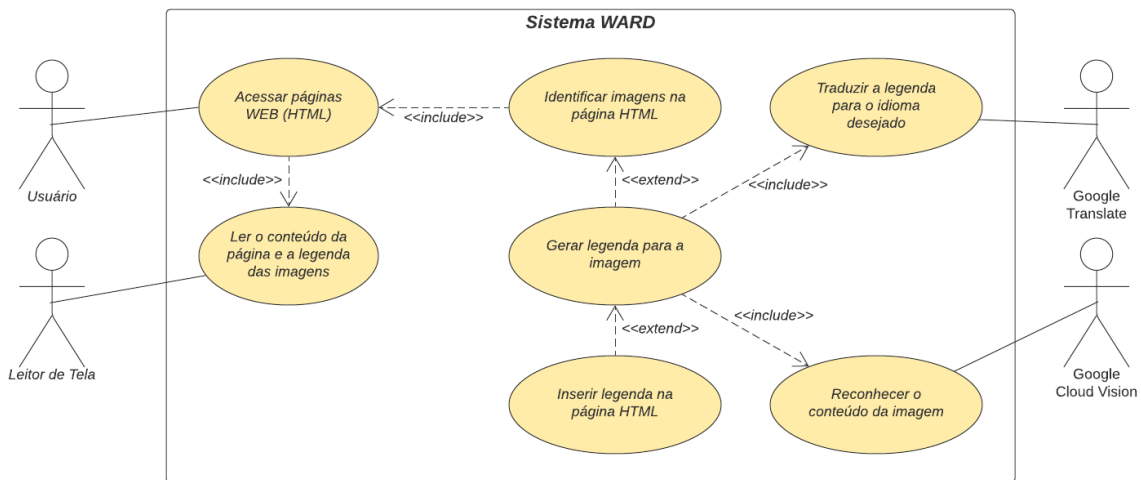


Figura 1. Diagrama de de Caso de Uso

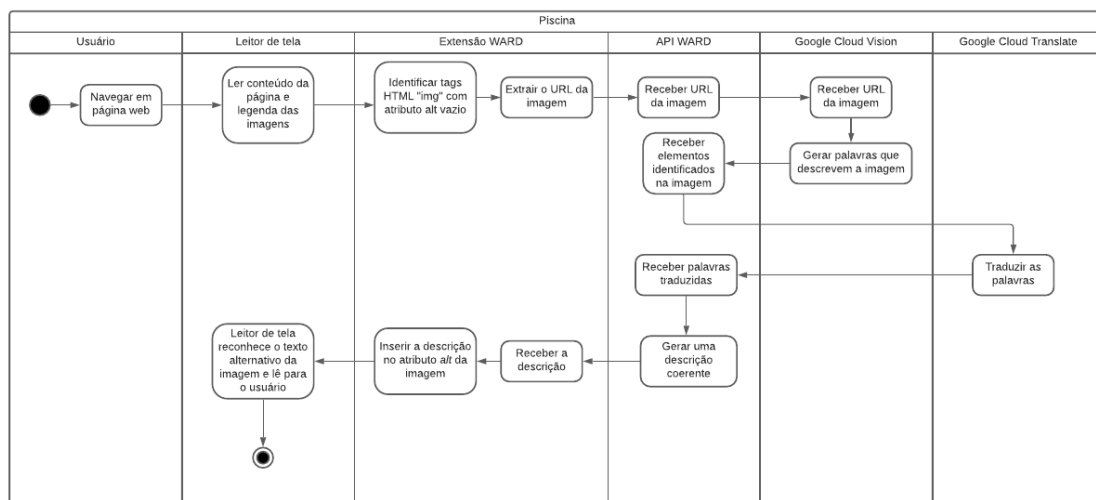


Figura 2. Diagrama de Atividades

navegador Google Chrome e comunica com a API desenvolvida em Node.js para o envio de URLs de imagens através de conexões HTTP. A API WARD comunica também através de conexões HTTP com duas APIs da Google, sendo elas Google Cloud Vision e Google Cloud Translate. A comunicação é feita através de requisições HTTP.

5.2.2. Diagrama de Pacotes

O diagrama de pacotes (Figura 4) foi criado para especificar a divisão de pacotes do sistema e organizar os modelos utilizados no sistema. O sistema WARD é dividido em dois principais componentes, sendo o primeiro a extensão WARD. A extensão utiliza o segundo principal componente, que é a API WARD. A API é responsável por realizar a comunicação do sistema com os demais sistemas externos do Google Cloud, que são as APIs Google Cloud Vision e Google Cloud Translate.

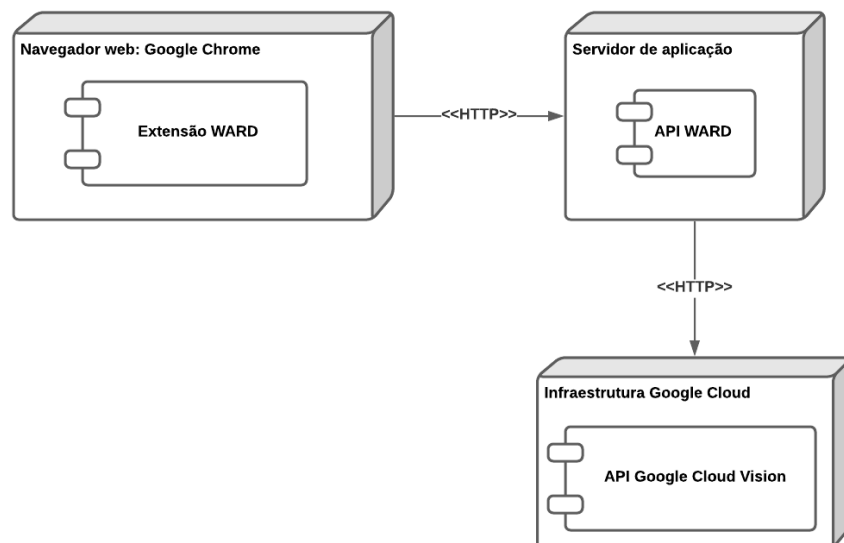


Figura 3. Diagrama de Implantação

5.3. Implementação

A API WARD foi desenvolvida para funcionar como ponte entre a aplicação disponível para o usuário final (a extensão WARD) e as APIs externas do Google Cloud. Para desenvolvê-la, foi utilizado o *framework* Node.js. O projeto possui três estruturas principais. A estrutura de inicialização do servidor Node foi definida no arquivo *server.js*. A estrutura de rotas da API, que possibilita que as funções internas sejam chamadas através de requisições, é distribuída entre os arquivos *app.js*, *routes/index.js* e *routes/imageRoute.js*. Por fim, a estrutura principal é o *controller* definido através do arquivo *imageController.js*, que é responsável por definir os métodos de reconhecimento de imagem através do consumo da API Google Cloud Vision, de tradução de textos através da API Google Cloud Translate, tratamento dos dados gerados e criação da descrição de uma determinada imagem.

A função disponibilizada pela API recebe por parâmetro dois campos principais no corpo de uma requisição do tipo POST no formato *JavaScript Object Notation* (JSON), sendo eles a URL da imagem a ser descrita e a linguagem de tradução da descrição. Por padrão, a API WARD considera o português como parâmetro para a linguagem. A resposta da requisição contém um campo contendo o resultado (*true* ou *false*), um campo com uma mensagem a respeito do resultado e o campo “*caption*”, que contém a descrição gerada pela API. Caso não seja possível gerar a descrição devido a falta de elementos identificados pela API Google Cloud Vision, a descrição será preenchida com uma mensagem de aviso. Caso ocorra qualquer outra exceção, a requisição não retornará o campo “*caption*” e o erro será descrito no campo “*message*”.

A extensão WARD ...

As APIs pertencentes ao Google Cloud utilizadas no sistema não são gratuitas. Para utilizá-las, é necessário ter uma chave de API que pode ser criada através do Go-

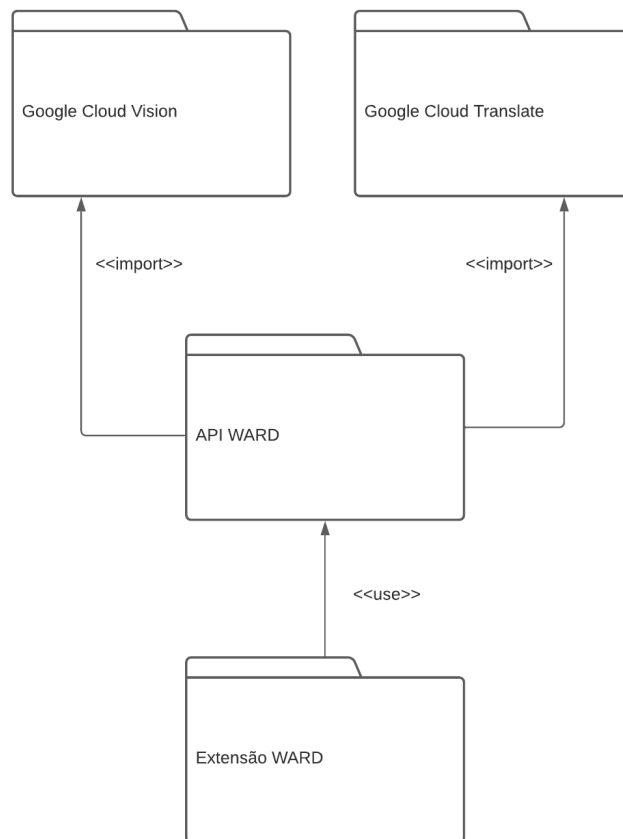


Figura 4. Diagrama de Pacotes

ogle Cloud Console²⁰. O sistema WARD foi desenvolvido apenas para o ambiente de testes devido aos impedimentos de preços existentes. As chaves de API utilizadas nos testes não serão disponibilizadas ao público. Entretanto, o código-fonte do projeto foi disponibilizado no GitHub através do repositório ward²¹. Um vídeo de apresentação do sistema em execução na perspectiva do usuário final também foi disponibilizado através do YouTube²².

5.4. Validação

Referências

Bodi, A., Fazli, P., Ihorn, S., Siu, Y.-T., Scott, A. T., Narins, L., Kant, Y., Das, A., and Yoon, I. (2021). Automated video description for blind and low vision users. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

²⁰Google Cloud Console - Acesso disponível em: <https://console.cloud.google.com/>. Último acesso em: 05 de outubro de 2021

²¹Repositório do WARD no GitHub - Acesso disponível em: <https://github.com/lucas-izidorio/ward/>. Último acesso em: 05 de outubro de 2021.

²²Vídeo de apresentação do WARD no YouTube - Acesso disponível em: xxxx. Último acesso em: xx de outubro de 2021.

- de Geografia e Estatística IBGE, I. B. (2011). *Sinopse do Censo Demográfico 2010*. Instituto Brasileiro de Geografia e Estatística - IBGE, Rio de Janeiro.
- Gleason, C., Carrington, P., Cassidy, C., Morris, M. R., Kitani, K. M., and Bigham, J. P. (2019). “it’s almost like they’re trying to hide it”: How user-provided image descriptions have failed to make twitter accessible. In *The World Wide Web Conference, WWW ’19*, page 549–559, New York, NY, USA. Association for Computing Machinery.
- Gleason, C., Pavel, A., McCamey, E., Low, C., Carrington, P., Kitani, K. M., and Bigham, J. P. (2020). Twitter ally: A browser extension to make twitter images accessible. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI ’20*, page 1–12, New York, NY, USA. Association for Computing Machinery.
- Guinness, D., Cutrell, E., and Morris, M. (2018). Caption crawler: Enabling reusable alternative text descriptions using reverse image search. pages 1–11.
- Sacramento, C., Nardi, L., Ferreira, S. B. L., and Marques, J. a. M. d. S. (2020). prace-gover: Investigating the description of visual content in brazilian online social media. In *Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems, IHC ’20*, New York, NY, USA. Association for Computing Machinery.