UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

HENRIQUE CHAVES PACHECO

A Web-Based Image Annotation Tool Supporting Multiple Users and Formal Ontologies

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Eduardo Simões Lopes Gastal

Porto Alegre
October 2022

ABSTRACT

This work describes the implementation and design process of an image-annotation web application. The main goal of the proposed system is to generate annotated data to support the training of Machine-Learning classification algorithms. For the purpose of maximizing quantity and quality of the annotated data output and minimizing end-user effort, the proposed system supports multiple remote users interacting simultaneously. The system also includes two types of image annotation interfaces, in addition to integrating with formal category ontologies through OWL files. The development process took into consideration: the nature of the task of establishing relations between images and categories, existing technologies (both for web development and image annotation), as well as requirements elicitation through end-user feedback from incremental releases. Ultimately, user experiments with the proposed **Labelweb** system (using the single-image annotation interface) generated more than 30,000 annotations and 10,000 labeled images in 9 weeks.

Keywords: Image annotation. Web-based tool with concurrent access.

Uma Ferramenta Web de Anotação de Imagens com Suporte para Múltiplos Usuários e Ontologias Formais

RESUMO

Este trabalho descreve o processo de implementação e design de uma aplicação web de anotação de imagem. O principal objetivo do sistema proposto é gerar dados anotados para auxiliar o treinamento de algoritmos de classificação de Aprendizado de Máquina. Com o objetivo de maximizar a quantidade e a qualidade dos dados anotados e minimizar o esforço do usuário final, o sistema proposto suporta múltiplos usuários remotos interagindo simultaneamente. O sistema, também, inclui dois tipos de interfaces de anotação de imagem, além de integração com ontologias de categorias formais por meio de arquivos OWL. O processo de desenvolvimento levou em consideração: a natureza da tarefa de estabelecer relações entre imagens e categorias, tecnologias existentes (tanto para desenvolvimento web quanto para anotação de imagens), bem como o levantamento de requisitos através do *feedback* de usuário das entregas incrementais. Por fim, os experimentos com usuários no sistema proposto, chamado de **Labelweb**, utilizando a interface de anotação de imagem-única, geraram mais de 30.000 anotações e 10.000 imagens rotuladas em 9 semanas.

Palavras-chave: Anotação de imagens. Ferramenta Web com acessos simultâneos.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Data annotation is the process of associating metadata with the elements of a dataset. For example, each image (the element) in an image collection (the dataset) may be associated with a list of textual labels (the metadata) describing the contents of the image. This is a well known necessary step to build an Artificial-Intelligence (AI) system based on Machine Learning (ML), capable of, for example, automatically classifying images (DENG et al., 2009a). The annotated dataset used for training an ML system is often called a *training dataset*.

The quality of the training dataset, i.e. how close the labels correspond to the real-world semantic concepts of the content represented by the data, will determine the quality of the resulting solution. Further, quantity is also a key factor and a concern, since this dataset must sample the semantic categories well enough in order to generate knowledge – the capacity of identifying an artifact by a neural network. However, a major challenge of the annotation task is scalability (DENG et al., 2009b). The number of images needed for state-of-the-art computer vision algorithms is up to thousands or millions for training and evaluation, making it, potentially, a very costly step (DENG et al., 2014).

A web-based annotation system is a common approach to tackle the data annotation step, allowing a large set of users to simultaneously hand-annotate data in favor of building a labeled dataset. The simple yet time-consuming task of iteratively and manually creating labels can be challenging due to the repetitive nature of the task. This can negatively impact both quantity and quality of the system's output (ie, the labeled dataset) and the subsequent computation results of the ML algorithm trained on the labeled dataset. Therefore, when designing such a system, one must take into consideration features that improve the overall user experience, such as helping the user with category suggestions and providing an annotation editor (eg, for correcting a previously-made annotation). It is also important to consider how the users' annotation efforts will be distributed across the dataset, providing a "smart" data selection strategy (ie, efficiently defining which data elements will be annotated by which users to maximize the system's throughput, but still guaranteeing some level of redundancy for error correction). Furthermore, considering the diversity of annotation scenarios, it is critical to evaluate the impact of customization features that attend the needs of the end-user considering the specific annotators group context, tools and domain familiarity, input and output data types, and interaction.

Existing annotation tools (presented in Chapter 2) offer good solutions for specific

situations, but do not always fit a particular annotation context with other requirements, such as loading a predefined list of categories from a formal ontology or connecting to a back-end category suggester service.

1.1 Important Definitions

The following **concepts** will be used throughout this text:

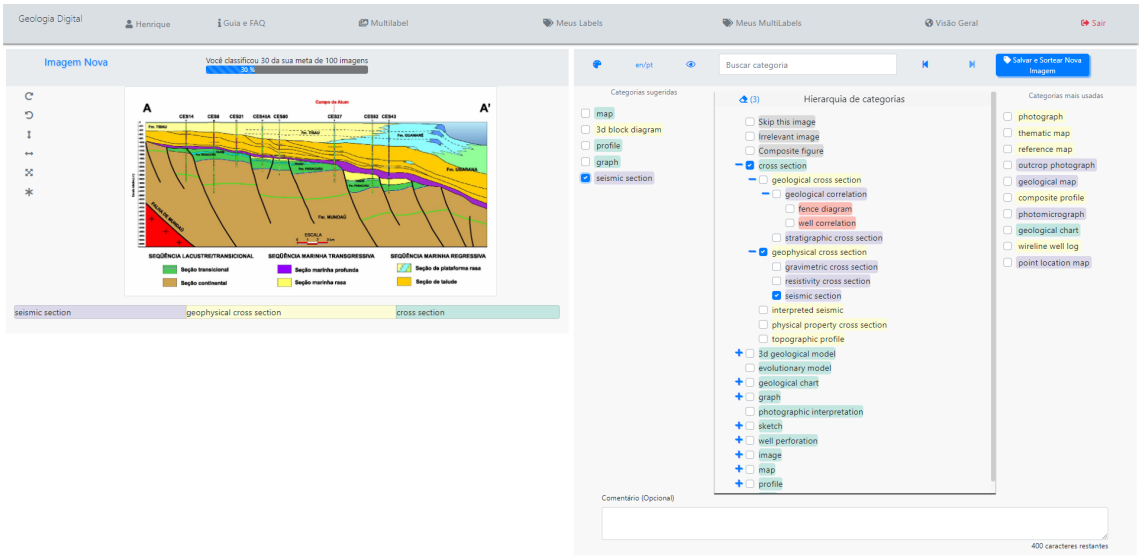
- a **category** is a unique named tag within a semantic hierarchy;
- a **label** is an association between an image and one or multiple categories;
- an **interaction** is an association between one or multiple images and one category.

1.2 Work Proposal

This work proposes the implementation of a new system for image annotation on the web, named **Labelweb** (Figure 1.1). This tool was developed using Node.js, Express, MongoDB, HTML, CSS, JavaScript, JQuery and other common modules used for web development. The main features of Labelweb are:

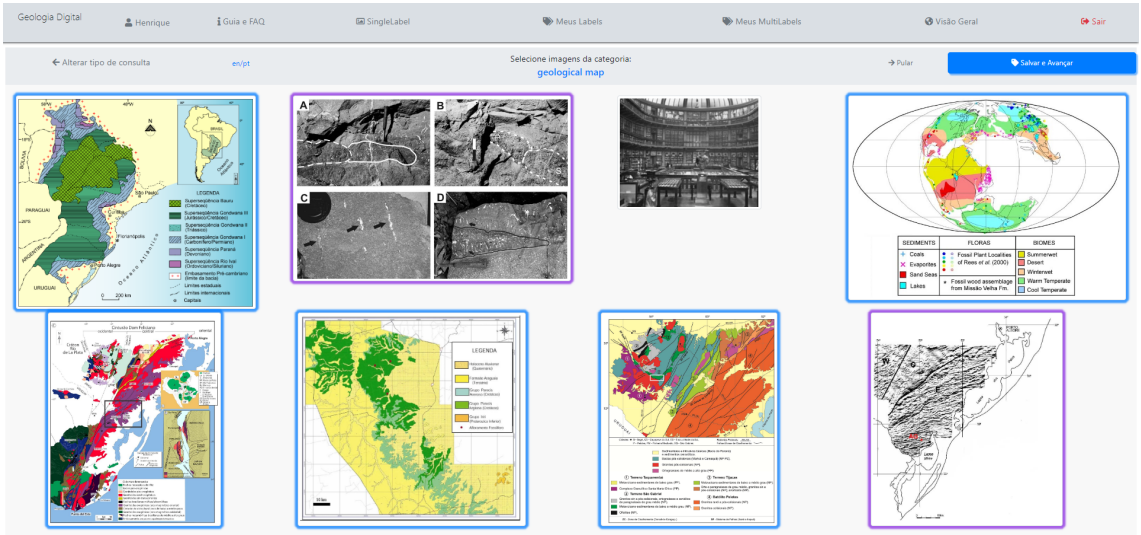
- **Support for multiple simultaneous users.** In this way, the dataset can be annotated more efficiently by dividing the necessary annotation work among several users;
- **Support for two types of image annotation interfaces.** In particular:
  - Interface for assigning multiple categories to one image at a time, referred to as **single-label** annotation (Figure 1.1);
  - Interface for assigning one category to a group of several images at a time, known as **multi-label** annotation (Figure 1.2);
- **Support for loading and displaying categories from a formal ontology.** The categories used for labeling are defined through the OWL ontology format, and are displayed hierarchically in the system's interface;
- **Support for assigning labels to whole images.** Users are not required to select objects inside the images using bounding boxes or polygonal shapes. This makes the annotation process faster and easier for the user;
- **Support for secure self hosting.** Labelweb can be hosted in a closed Local Area

Figure 1.1: Single-image interface of Labelweb for annotating one image at a time, possibly assigning several categories to the image.



Source: The author

Figure 1.2: Multi-image interface of Labelweb for annotating several images at a time, assigning a single pre-defined category (selected by the user or the system).



Source: The author

Network (LAN), thus guaranteeing secure annotation of a private database of images.

At the system core, Labelweb provides two annotation methods as mentioned: single and multi-image. The default annotation mode is single-image annotation. In this interface, the user assigns system categories, which are stored in the system database (using an ontology as input in OWL format (W3C, 2012)), to each random image picked by the back-end, iteratively, one at a time. This allows for precise per-image annotation (depending on the user expertise in the data domain). Alternatively, the multi-image interface allows the user to, iteratively, associate a subset of images to a single specific category, suggested by the system, from the same original ontology. In this way the multi-image approach can be perceived as a faster annotation method, generating multiple labels at a time, but also generating possibly less precise information, depending on how the suggested category was generated and how granular the resulting labels are.

Following, we will explore these concepts further through Labelweb's related works (Chapter 2), architecture, features and development methodology (Chapter 3), as well as details on the system interface and front-end modules (Chapter 4), back-end modules, control state machines, business logic and settings (Chapter 5). Finally, we will explore some of the experiments results achieved with this system (Chapter 6).

2 RELATED WORKS

An annotation tool aims to build a collection of images with ground-truth labels to be used for training classification algorithms (for use in object detection and recognition research (DENG et al., 2009a)). However, this resource can be scarse, considering how costly the manual task of annotation can be. Thus, an annotation tool must take into account many interaction challenges and should provide enough features that minimize the cost of picking categories for images and keep the user motivated towards a goal. Many strategies can be used to design this type of tool: specific domain-focused features for a single type of user, data analysis of user behaviour in social media or gamified collaborative efforts. Nevertheless, the objective is to maximize quality and quantity of data.

Regarding Labelweb and the challenge of the annotations, Deng et al. (2014) propose three properties that can be explored to design a more efficient tool:

- Correlation: since sets of correlated categories tend to be presented in an image related to a context. This is a property explored in the Labelweb Multi-image annotation mode by providing to the user multiple images correlated by visual similarity in a single query;
- Hierarchy: since humans tend to categorize groups of higher semantic abstraction more efficiently. Labelweb provides categories in a semantic hierarchy extracted from the system ontology crafted by specialists. This enables an easier and faster navigation through the categories list and encourages the end-user to be as specific as possible, but also sustaining value to higher level and more general annotations;
- Sparsity: since the cost of classifying multiple categories in an image is logarithmic in relation to the total number of possible categories (DENG et al., 2014). Labelweb ensures that every image has at least one category. To ensure this, Single-Image and Multi-image annotation modes provide exception categories that allow the end-user to signal an image that cannot be identified, thus not being able to annotate. In those minority cases, post-processing might be needed to identify the best way to treat the data.

In the following sections we present existing image annotation tools, which are compared in Table 2.1.

Table 2.1: Characterization of existing image annotation tools and comparison of their features. The following features are compared: (i) Support for multiple simultaneous users working together to generate a labeled dataset; (ii) Support for assigning labels to whole images, without requiring the user to draw a Bounding Box (BBOX); (iii) Support for loading categories from a formal ontology in the OWL format; (iv) Support for self-hosting in a closed and secure LAN; (v) Free to use.

Tool	Multiple Simultaneous Users	Non-BBOX Categories	Ontology OWL	Self Hosting	Free to use
Labelweb (Ours)	Yes	Yes	Yes	Yes	Yes
OpenLabeler	No	No	No	Yes	Yes
Sloth	No	Yes	No	Yes	Yes
LabelImg	No	No	No	Yes	Yes
RectLabel	No	No	No	Yes	No
LabelMe	Yes	No	No	No	Yes
Labelbox	Yes	Yes	No	No	No

2.1 OpenLabeler

OpenLabeler (WONG, 2022) is an open source application for image annotation written in OpenJDK (Figure 2.1). This application cannot be used on the Web and it works using a bounding box segmentation during its annotation process. It generates a PASCAL Visual Object Classes (VOC) format XML annotation file (EVERINGHAM et al., 2010), supported in artificial intelligence and deep learning training. A key feature of this tool is that it uses an inference method implemented with TensorFlow to improve accuracy and assist in the annotation process. Since the user must select the bounding box manually, this feature helps speeding up the annotation process. This suggestion feature also allows the user to train an intermediary model at any point during the annotation process using the dataset provided by the labels created so far, which optimizes the labeling suggestions. However, since this tool must run locally, it does not support multiple simultaneous users and could be problematic for the annotators group management.

2.2 Sloth

Sloth (HCI Lab, Karlsruhe Institute of Technology, 2014) is a versatile bounding box labeling tool (Figure 2.2) which provides a framework and a set of configurable settings to attend specific user needs running the application in a local machine. An advantage of the sloth annotation system is precisely how configurable the system is. For instance,

the tool allows the user to define the type of geometric class of the annotation bounding box (point, rectangle or polygon), which is essential since it will balance output data quality and complexity of the annotation task. Other than that, the settings describe how one annotation type is visualized, inserted and modified on the interface. Also, in order to facilitate the annotation process, the system has a configurable set of hotkeys usable from the user interface. Nevertheless, besides not supporting multiple simultaneous users, The Sloth tool interface must be configured in order to be support a Non-Bounding Box annotation.

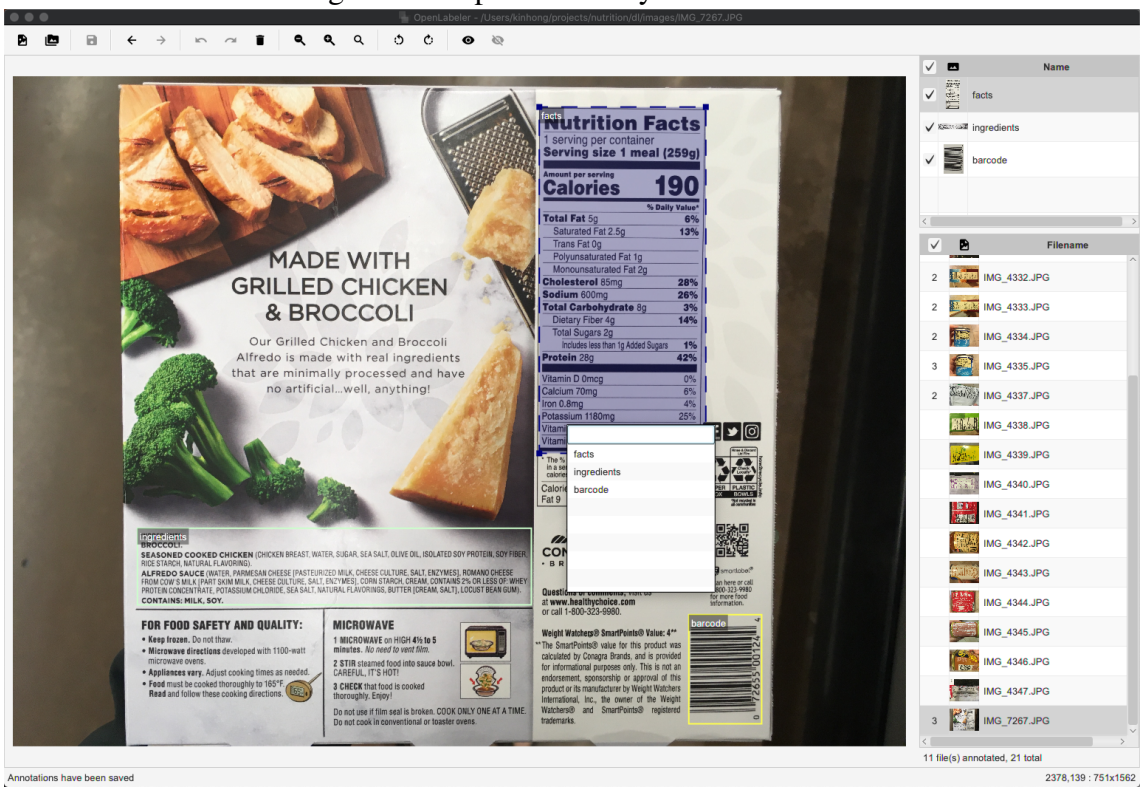
2.3 LabelImg

Similar to the Openlabeler and Sloth tools, LabelImg (TZUTALIN, 2018) is a graphical image annotation tool (Figure 2.3) that uses bounding boxes and exports to XML files in PASCAL VOC format used by ImageNet (DENG et al., 2009a). The system also provides many hotkeys for ease of usage of many commonly used functions such as rectangular box creation, skipping current image, saving annotation, returning to previous image, etc. Beyond annotating classes, the user can also provide feedback on how clearly identifiable is the element, this helps a customization of the deep neural network implementation which can later exclude elements that are hardly identifiable during training. Still, this tool does not support online annotation and generates the annotated dataset based exclusively on bounding boxes images annotations.

2.4 RectLabel

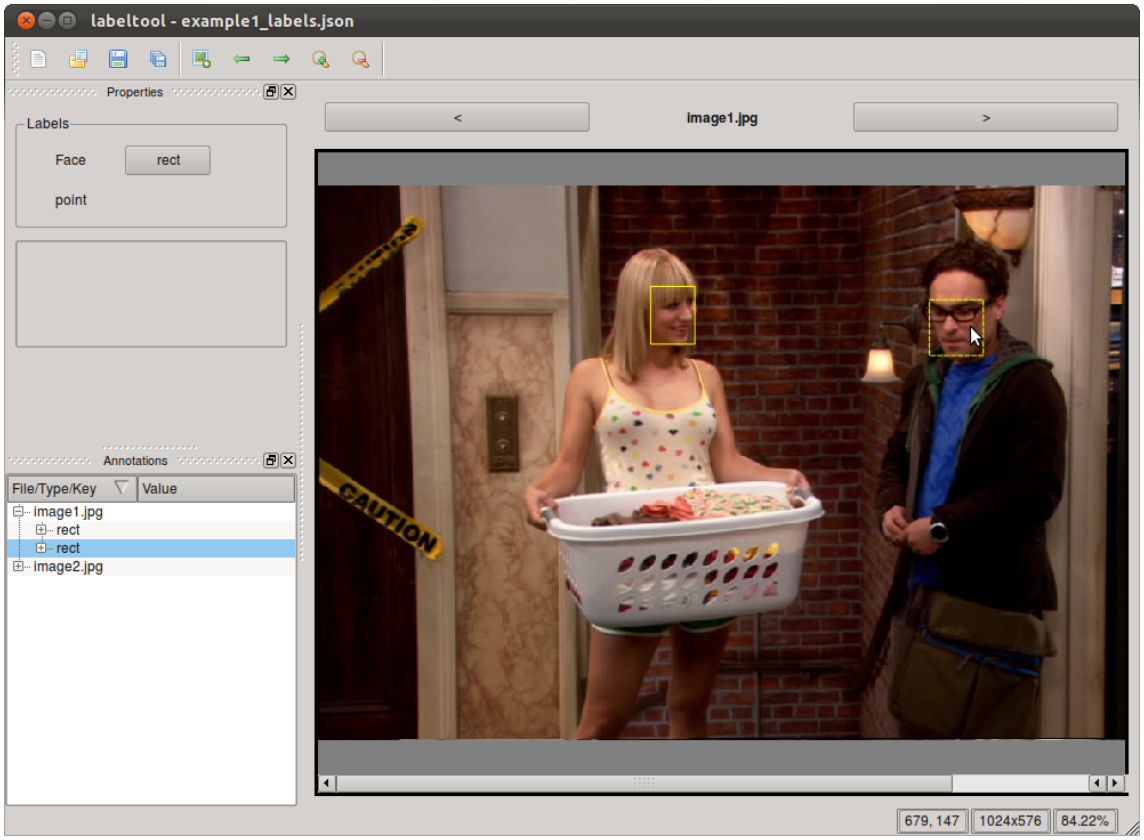
RectLabel (KAWAMURA, 2022) is an annotation tool to label images or video frames for bounding box object detection and segmentation (Figures 2.4 and 2.5). It supports YOLO text format (REDMON; FARHADI, 2018) for the rectangular annotation, but it also supports high-precision annotation through image segmentation using complex shapes such as polygons and cubic bezier curves (Figure 2.5a) and keypoints with skeleton for human body annotation (Figure 2.5b). It also provides automatic image annotation using Core ML models pre-trained for more than 5000 objects (Figure 2.4b). The system also embeds a variety of image transformation features, which may enhance the quality of the output data. However, similarly to the tools described previously, RectLabel must run

Figure 2.1: OpenLabeler system interface.



Source: Wong (2022)

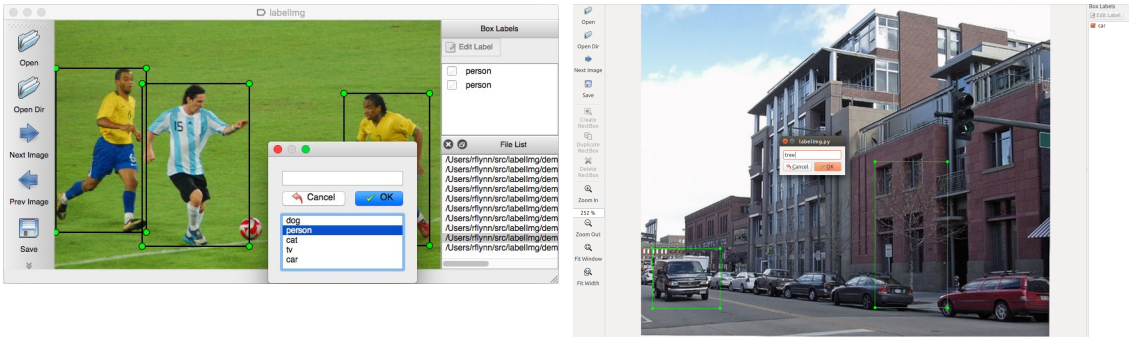
Figure 2.2: Sloth system interface.



Source: HCI Lab, Karlsruhe Institute of Technology (2014)

Figure 2.3: LabelImg system interface.

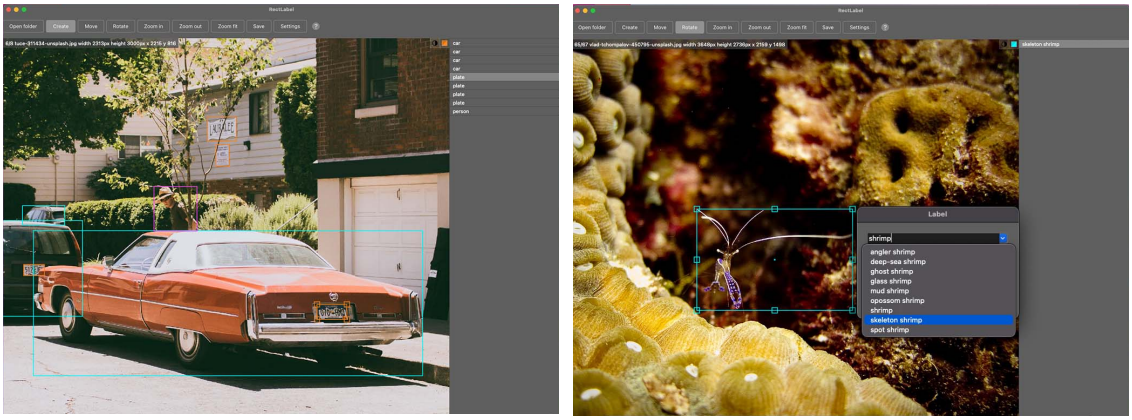
(a) LabelImg system annotation (b) LabelImg label creation



Source: Deng et al. (2009a)

Figure 2.4: RectLabel Tool Interface.

(a) RectLabel system interface (b) RectLabel recommendation feature



Source: Kawamura (2022)

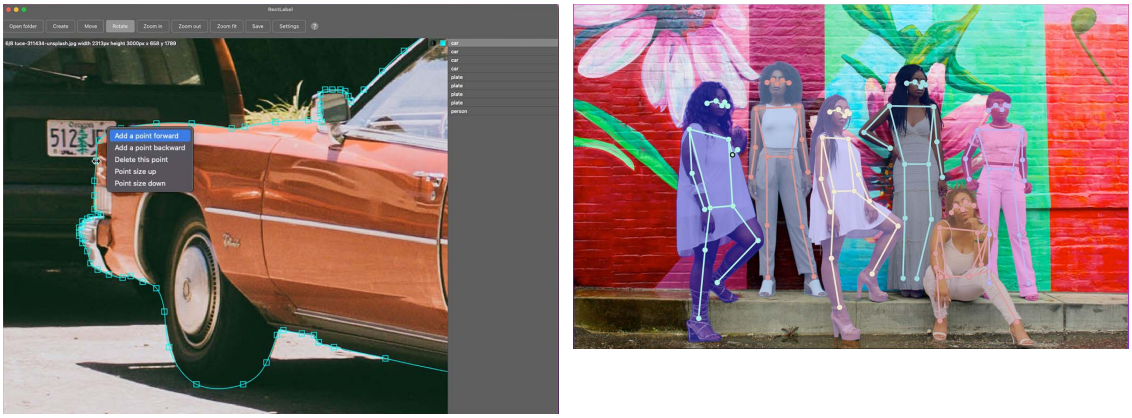
locally and is available exclusively on Mac OS for a subscription fee.

2.5 LabelMe

The LabelMe (HEARTEX, 2018; RUSSELL et al., 2008) annotation tool provides a web based image annotation tool supporting multiple remote annotations simultaneously (Figure 2.6). This system allows the user to host a server for annotating with complex cropped shapes, such as: bounding boxes, polygons or segmentation masks. Each shape can be associated with classes and the annotations can be visualized in an image folder (Figure 2.6b). The tool is designed to provide a drawing interface that is easy to use and allows instant sharing of the collected data. After completion, the label created by an user is immediately available for download and is viewable by subsequent users who visit the same image. The resulting labels are stored in the XML file format, which makes the

Figure 2.5: RectLabel high precision annotation.

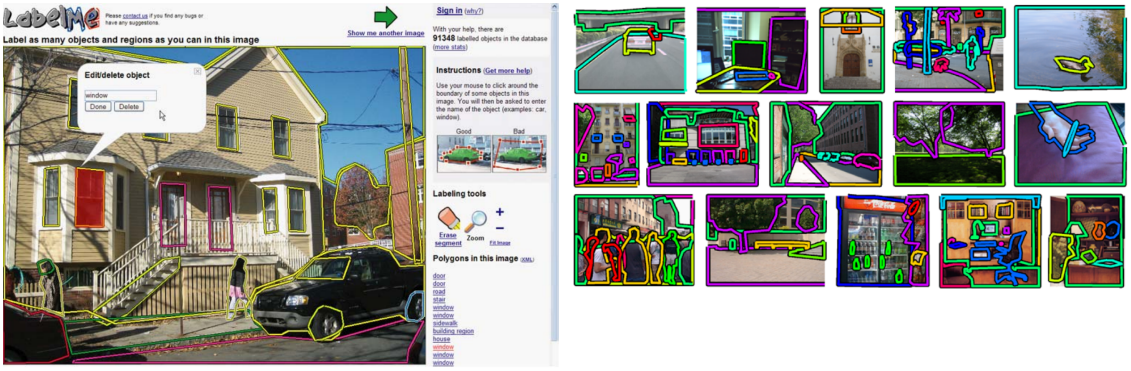
(a) Cubic bezier curves annotation (b) Skeleton keypoints annotation



Source: Kawamura (2022)

Figure 2.6: LabelMe Tool.

(a) LabelMe Interface (b) LabelMe Annotated Images



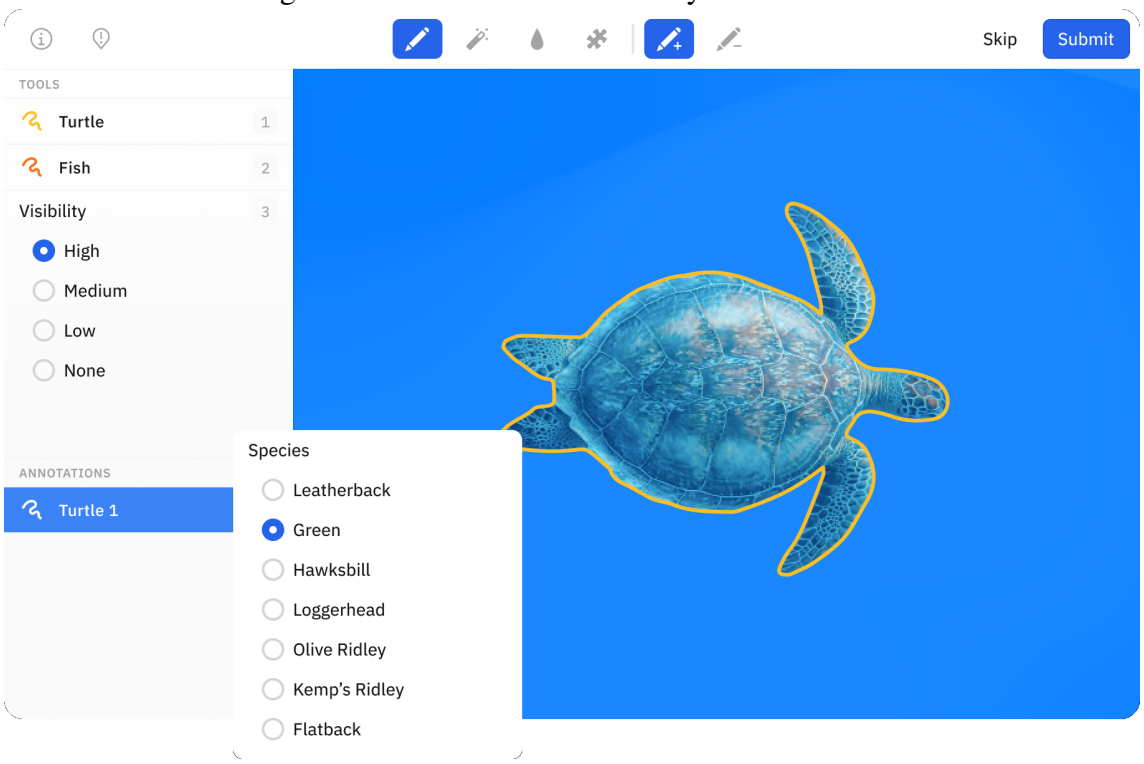
Source: Heartex (2018), Russell et al. (2008)

annotations portable and easy to extend. Even though this tool speeds up the annotation process by supporting multiple simultaneous users, it does not support Non-Bounding Box annotations, users may provide complex polygon boundaries and this can lead to a cumbersome annotation process.

2.6 Labelbox

Labelbox (Labelbox, Inc., 2022) is a data engine for Artificial Intelligence that provides many advanced features to support Machine Learning solutions in general (Figure 2.7). This tool addresses three general tasks for AI development, which are: cataloging, annotating and modeling. The online annotation tool product in this engine offers many highly customizable features for different types of data (image, video, text, document, audio, medical, geospatial and custom). In order to speed up the annotation process without

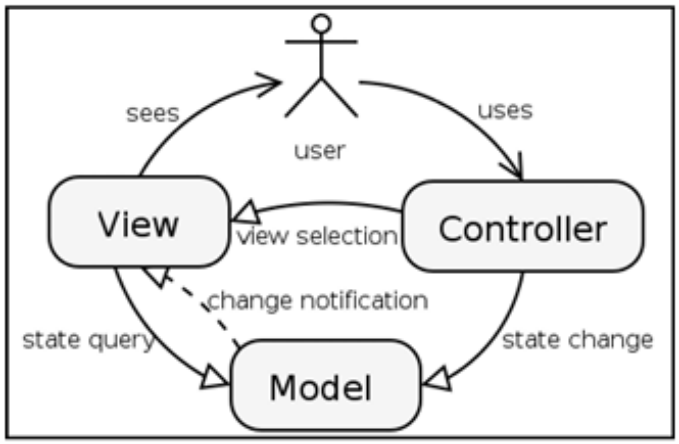
Figure 2.7: Labelbox annotation system interface.



Source: Labelbox, Inc. (2022)

compromising data quality, Labelbox's annotation interface combines automation tools with bounding boxes to assist the user when using the tool for image segmentation. Other features that are worth mentioning are the performance dashboard, which includes dynamic statistics and relevant information for data analysis and quality-control of the annotations, and communication tools that encourage issue resolution and collaboration. Currently, Labelbox offers a free trial limited by number of users, annotations and data rows, as well as the paid Labelbox Pro and Enterprise versions.

Figure 3.1: Model-View-Controller Architectural Framework separates the concerns of the application between the components responsible for handling data, user interface and client-side logic. This framework is commonly applied to web application design as a means to easily separate business and view logic, while keeping the code maintainable and extendable.



Source: Madeyski and Sochmialek (2005)

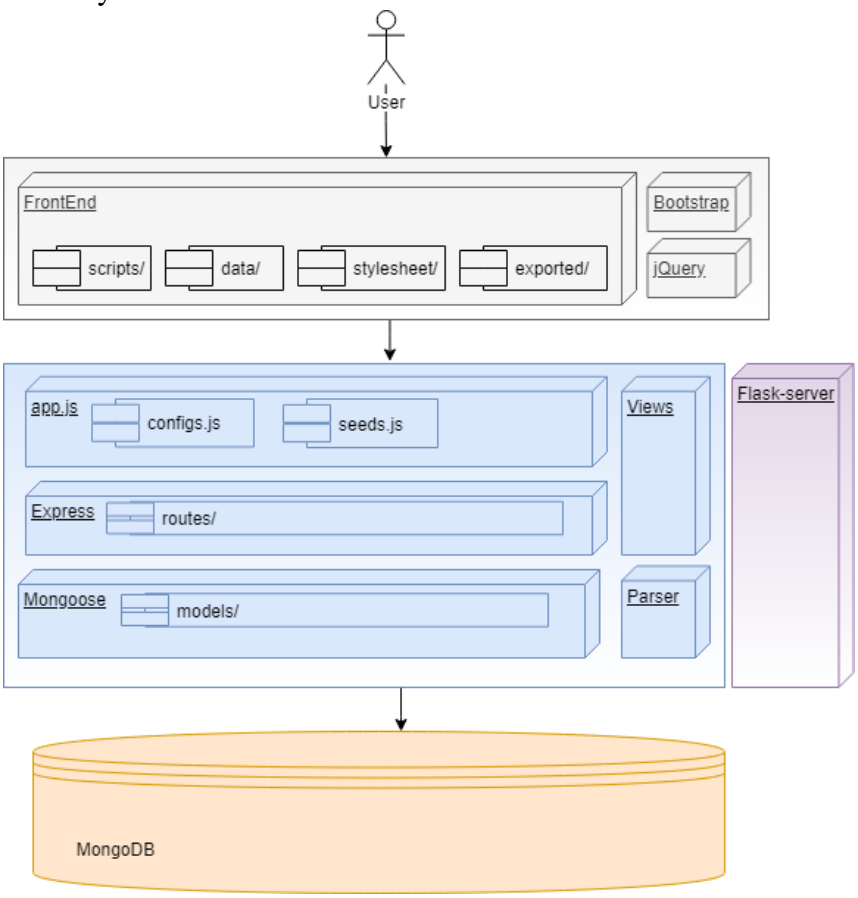
3 ARCHITECTURE OF THE LABELWEB SYSTEM

The Labelweb system architecture is based on the *Model-View-Controller* architectural framework (Figure 3.1), which separates the system concerns into three modules:

- Model – contains business logic and is responsible for the application data and behavior;
- View – specifies the graphical presentation of the model;
- Controller – reacts to user actions and controls the system state (MADEYSKI; SOCHMIALEK, 2005).

This is a common pattern used in web applications. It is important for the Labelweb system in order to achieve asynchronous communication on the back-end to generate potentially many simultaneous data annotations, while keeping it consistent with business logic on the server-side It is also focused on optimizing the performance and user interaction of less demanding tasks, such as generating dynamic dashboards and image transformations, on the user-side. Figure 3.2 presents the general layered system architecture.

Figure 3.2: Labelweb System Macro Layered Architecture. The following architecture separates front-end (gray layer), back-end (blue and purple layer) and database (orange) technologies and system modules.



Source: The author

3.1 Features Set

The key features integrated in the system are the single-image online annotation, the multi-image online annotation, automatic loading of categories in a hierarchical semantic web language from a OWL file format, category suggestion from a Neural Network model designed and trained for image prediction and hosted in parallel in a Flask server (Section 5.6), and annotation data extraction in CSV and ZIP file formats, which then can be used as input for retraining Machine Learning algorithms.

The following list displays the system features' subsets according to each user authorization level. That is,

- authorization level 1 – common users;
- authorization level 2 – authorized users;
- authorization level 3 – superusers;

- authorization level 4 – system administrators.

The authorization level of a particular user determines the subset of features available for the user, where higher authorization levels include all features from lower levels. The features are as follows:

Common Users (Level 1)

- Creating an account
- Checking basic user guide
- Annotating images using single and/or multi-label annotation method
- Visualizing self-created single and multi-label annotations
- Editing account information
- Changing labels' hierarchy visualization/color palette
- Switching interface language between pt-br (Brazilian Portuguese) and en-us (US English)
- Commenting annotation

Authorized users (Level 2)

- Creating new categories in the ontology hierarchy
- Editing created categories

Superusers (Level 3)

- Visualizing single and multi-label annotations from all users
- Checking system's statistics
- Editing categories' interface visibility
- Exporting annotation data in CSV or ZIP format

Administrators with access to the `config.js` file (Level 4)

- Configuring user annotation goal
- Configuring number of simultaneously active images (Section 5.2.2)
- Configuring minimum number of required labels per image
- Configuring system route keys for user registration (Section 5.3) and administrator password
- Allowing multilabel annotation and category editing

3.2 Annotation System Development Methodology

After analyzing existing toolsand technologies, and validating a solution idea prototype, the development process of the annotation system was carried out through incremental iterations, each iteration lasting one week and composed of these sequential steps: validation of the current system state, gathering of new requirements, code development followed by unitary and integration tests.

The use of Node.js as a tool for building the backend of the annotation system allowed the development to be realized mostly in the Javascript language. In addition to being a popular tool with extensive documentation and online support, Node.js is an asynchronous event-driven runtime server model and has many advantages over other concurrent blocking thread based models (DAYLEY, 2014). The framework allowed the system to be designed and written in Javascript end-to-end, which led to easier refactoring, and, since it has a large development community, the system leverages many available modules and resources for the development of Labelweb's funcionalities. Therefore, the resulting Labelweb system uses a popular stack architecture for web applications, consisting of open-source tools such as MongoDB, Express and Node.js.

4 FRONT-END LAYER

This section describes the front-end layer of the Labelweb system, with a user interface that is accessible through any modern Web browser software.

4.1 System Access Routes

We start by listing the routes (URLs) in the system's address space, used to access the different *pages* of the system's interface. The following list of routes are accessible by different levels of users, obeying the respective restrictions. Each route's page is available at the address defined by appending the route to the hostname or IP address where the system is hosted. For example, if the system is hosted locally at `127.0.0.1`, the `/index` route can be accessed by pointing the Web browser to `https://127.0.0.1/index`. Furthermore, some routes are "dynamic" and allow for parameters in the URL, indicated by keywords starting with a colon, ":". For example, the route `/survey/:title` is accessible by substituting the `:title` keyword with the name of an image in the system's database, eg, `/survey/image-1.jpg`.

Routes available to users without an active Session (ie, logged-off):

- **/index:** Route to login.
- **/:key/register:** Route to perform the registration of a new user. Registration is only allowed if the user sets the `:key` parameter to the same value of the registration key determined by the configuration option *routeKey*, defined in the system configuration file config.js.
- **/image/:state:** Route for textual listing of image titles (filenames) in the system's database that are in a particular state, determined by the `:state` parameter (one of: inactive, active, ready, or labeled).
- **/csv:** Support route for extracting system data in CSV format.
- **/zip:** Support route for extracting system data in ZIP format.
- **/stats:** Support route for textual visualization of basic system statistics, in a JSON format.

Routes available to users with an active Session (ie, logged-in):

- **/about:** Route to access the page containing useful information about using the system.

- **/logout:** Route to perform session log-out operation.
- **/survey:** Route for selecting a new image to be annotated by the user, if the current selected image in the user's session has already been annotated.
- **/survey/:title:** Route for annotating the image determined by the `:title` parameter.
- **/survey/:title/predictor:** Communication route with the Flask server (Section 5.6) to obtain suggested categories for the image determined by the `:title` parameter.
- **/survey/:title/comment:** Support route for getting user comment for the image determined by the `:title` parameter.
- **/survey/:title/back:** Support route for obtaining the annotated image that appeared before the current image, determined by the `:title` parameter.
- **/survey/:title/forward:** Support route for obtaining the annotated image that appeared after the current image, determined by the `:title` parameter.
- **/MultiImage-survey/type:** Route for for the user to choose the type of multi-image annotation to be performed (Section 4.2.2). between category annotation and random image actions or only random image. This determines the `:action` parameter in the multi-image search route (below).
- **/MultiImage-survey/next-action/:action/category/:owlid/image/:imagetitle:** Multi-image annotation route, determined by the `:action` parameter, a system category `:owlid` and a pre-selected search image `:imagetitle`, used as input query for selecting other images.
- **/popular:** Support route to obtain the list of most used categories by all system users.
- **/empty:** Support route for warning in case of exception, where the user has cycled through all the images in the database.
- **/unavailable:** Support route for warning in case of exception, where the user tried to access an image that does not exist in the database.
- **/user:** Route to access user data edit page.
- **/labels:** Route to visualize all images annotated by the user using the single-image annotation interface. The corresponding page has a search box for searching through all images annotated by the user.

Route available to authenticated Superusers

- **/all:** Route to access the search and overview page showing all images annotated by all users and all annotations types. addThis page also allows for configuring the

visibility of categories in the annotation interface and data extraction.

- **/all/visibility:** Route to access the system category visibility edit page.
- **/all/getbatch:** Support system route for obtaining a set of 10 system images coupled with all their annotations.
- **/usercomments:** Support system route for getting all system feedback by the users.

4.2 Annotation: Modes of Interaction

The Labelweb annotation system allows the user, through a user registration or login, to use the resources of the system interface to annotate one image at a time, in the single-image annotation mode (1), or multiple images at a time, in the multi-image mode (2). The design of the annotation system's graphical interface takes into account the minimization of effort for a user to select categories for each image.

Each mode can be described in the following manner:

1. The process of assigning multiple categories to one image at a time in the system's database, known as **single-image** annotation;
2. the process of associating a single category to a group of images at the same time, known as **multi-image** annotation.

4.2.1 Single-Image Interaction

To annotate one image at a time, the user proceeds as follows:

- The user requests a new image to annotate by accessing the `/survey` route. This is the default system route: the user is redirected to `/survey` after logging-in to the system;
- The system randomly selects an image that does not have enough annotations from the list of active images (Section 5.2.2);
- The selected image is shown to the user alongside the entire categories hierarchy (Figure 1.1). The system also provides a list of (i) most commonly used categories in the system and (ii) the categories predicted for the current image based on the Flask server (Section 5.6).
- The user can interact with the image, rotating, mirroring or zooming-in, in order to

properly analyze it;

- The user selects the desired categories from the hierarchy, which is displayed as a tree with expandable nodes (Figure 1.1). The user can also filter the list of categories by entering a search term in the search box. Furthermore, when the user positions the mouse cursor over any of the categories, a tooltip shows the ontological definition of the category.
- The user can optionally enter a comment about the particular image being shown;
- Finally, the user saves the annotations by clicking on the button titled “Salvar e Sortear Nova Imagem” (Save and Select New Image). The user will then be redirected to the first step, where a new image will be randomly selected for annotation.

4.2.2 Multi-Image Interaction

To annotate multiple images at a time, there are two approaches for selecting the images that are shown to the user: (1) based on visual image similarity; and (2) based on the confidence of a pre-trained classifier.

4.2.2.1 Based on visual similarity

Image selection based on visual similarity works as follows:

- The user selects a category among all available, or asks the system to randomly select one of the categories from the ontology hierarchy. The category selected by the user must have at least one image already annotated as belonging to this category;
- The system randomly selects an image that belongs to the selected category, according to the annotations that already exist in the system;
- According to a pre-computed visual similarity table in the database of Labelweb, the system selects other images that are visually similar to the selected image;
- The selected image and its similar images are shown to the user in a quick annotation interface (Figure 1.2);
- If the user for any reason does not like the selection of images, the user can “skip” these images and ask for a new selection;
- The user then clicks to select all images that belong to the selected category (which become highlighted with a blue border). Images which do not belong to the category

are not selected by the user, and images for which the user is uncertain can be marked as such (which become highlighted with a purple border);

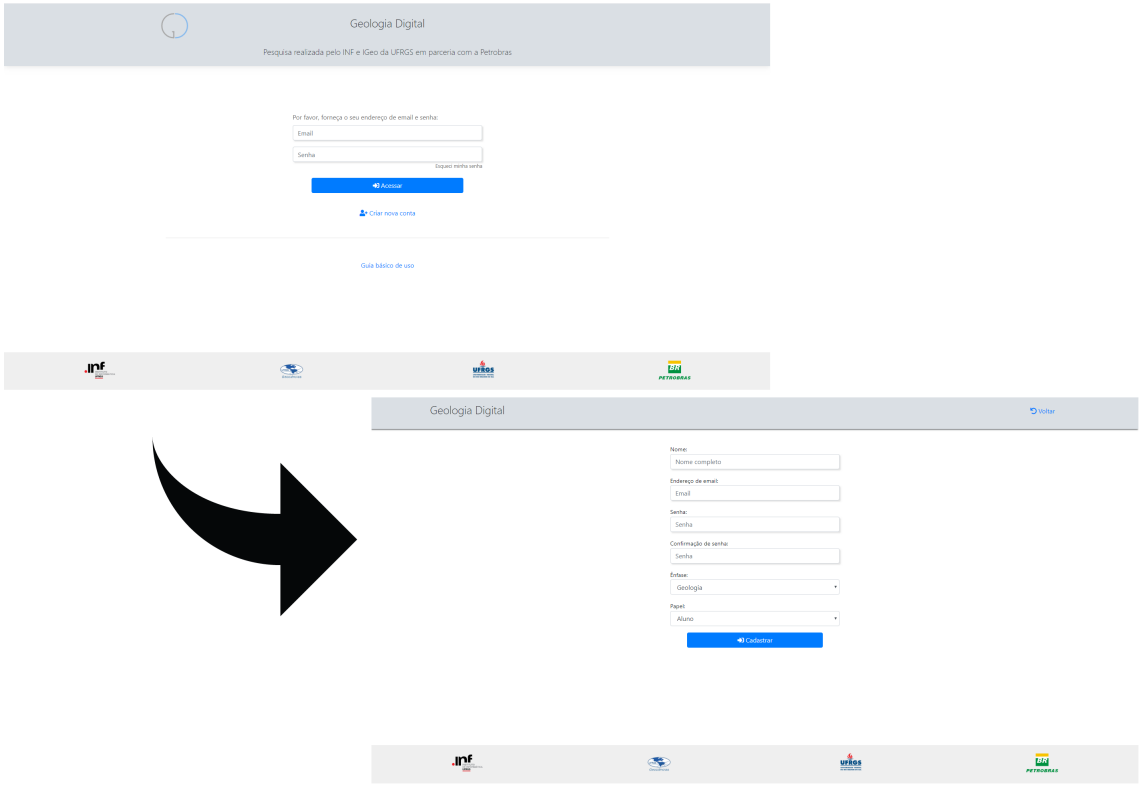
- Finally, the user clicks the "Save and Advance" button to save the annotations. The user will then be redirected to a new page with a new selection of images. If the user initially asked the system for a random category, a new random category will also be selected (otherwise, the same category initially selected by the user will be used).

4.2.2.2 Based on pre-trained classifier confidence

Image selection based on the confidence of a pre-trained classifier works as follows:

- The user selects a version of one of the pre-trained classifiers;
- The user selects a category among the categories known by the selected classifier, or asks the system to randomly select on of the categories;
- The system checks, among all the images in the image database that have not yet been annotated, which ones have the highest probability (according to the selected classifier) of belonging to the selected category;
- The most likely images are shown to the user in the multi-image annotation interface, along with some medium and low probability images (this is done to ensure classifier consistency in both true-positives and true-negatives);
- If the user for any reason does not like the selection of images, the user can "skip" these images and ask for a new selection;
- The user then clicks to select all images that belong to the selected category (which become highlighted with a blue border). Images which do not belong to the category are not selected by the user, and images for which the user is uncertain can be marked as such (which become highlighted with a purple border);
- Finally, the user clicks the "Save and Advance" button to save the annotations. The user will then be redirected to a new page with a new selection of images. If the user initially asked the system for a random category, a new random category will also be selected (otherwise, the same category initially selected by the user will be used).

Figure 4.1: User login and Registration Interface.



Source: The author

4.3 Interface

This section discusses in detail the main user interface elements of the proposed Labelweb system.

4.3.1 Login and Registration

Figure 4.1 demonstrates the interface for login and registration functionalities. These pages allow a registered user to access the system through valid credentials. If enabled, it is also possible to click on the “Criar nova conta” (Create New Account) button to register a new user by filling out a basic information form. It is also possible to hide the Create New Account button, in which case only someone that has access to the *routeKey* mentioned previously will be able to create an account. This is done by directly accessing the user registration page by through the route `/:key/register` with the parameter `:key` set to the same value as the *routeKey* configuration option.

Figure 4.2: Navigation Bar at the top of the Labelweb Single-image annotation interface.

Source: The author

4.3.2 Single-image Annotation Page

After logging in, the user has access to the system's single-image annotation page. On this page, shown in Figure 1.1, the user can perform actions related to navigation between system pages, visualizing category data and annotating images.

At the top of this annotation page, the navigation bar (Figure 4.2) allows the user to browse pages for editing registration information, editing created categories, visualizing user-saved labels, editing and visualizing management information and performing the log out operation.

Data blocks containing dynamically generated information are displayed just below the main navigation bar.

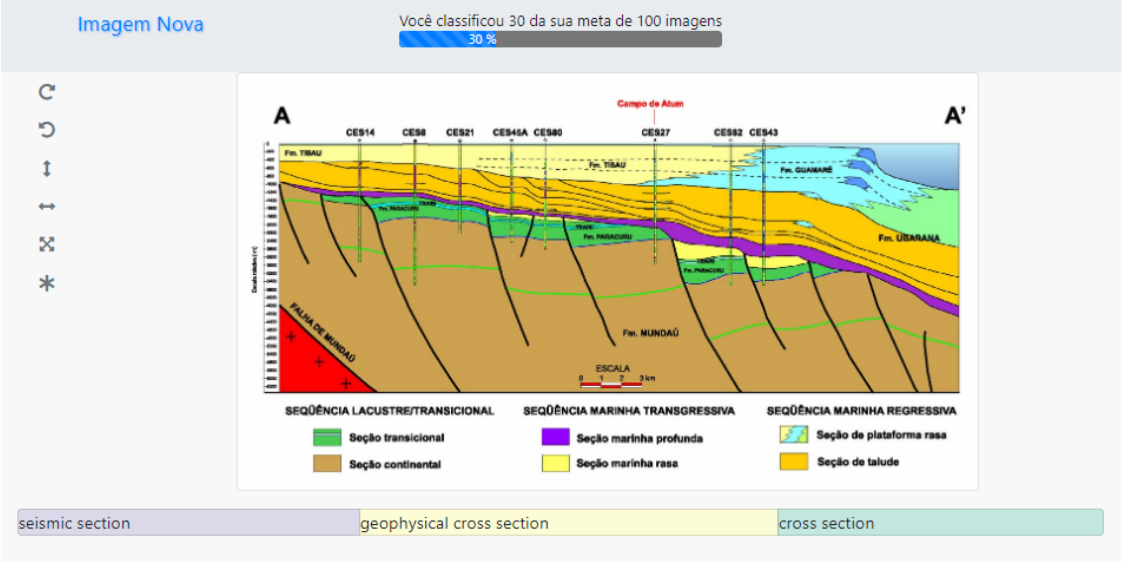
The block on the left (Figure 4.3) contains, at the top, a user progress bar, which counts the number of images that have been annotated by the user. Below, the currently selected image is shown in a reduced scale, being possible to enlarge it with just one click. It is also possible to use the buttons to the left of the image to perform geometric transformations on the current image (rotating 90º in both directions, horizontal and vertical mirroring and undo operations). These transformations are only used to help the user in better analyzing the image, and the image content is not changed in the database. Finally, below the image, an area is reserved for highlighting the categories selected by the user so far for the current image.

The block to the right of the page (Figure 4.4) has, at its top, an annotation help bar containing several functionalities and, just below, information from the semantic hierarchy loaded from the OWL document (ontology).

The annotation help bar (Figure 4.5) allows the user to configure the visualization of the information presented to him. This control can be done by (i) selecting a preferred color palette to highlight the hierarchical level of each category, (ii) searching for a category using the search bar by name in English or Portuguese or by synonym, (iii) changing the language (Portuguese or English) used in system name display and category definition, and (iv) clicking the advance and return buttons to flip through the images in the order that they have been annotated by the user.

Also on the block to the right of the page (Figure 4.4), below the annotation

Figure 4.3: Left Block of the Labelweb Single-image annotation interface displaying a new image for annotation, selected categories, user progress bar and image transformation buttons.



Source: The author

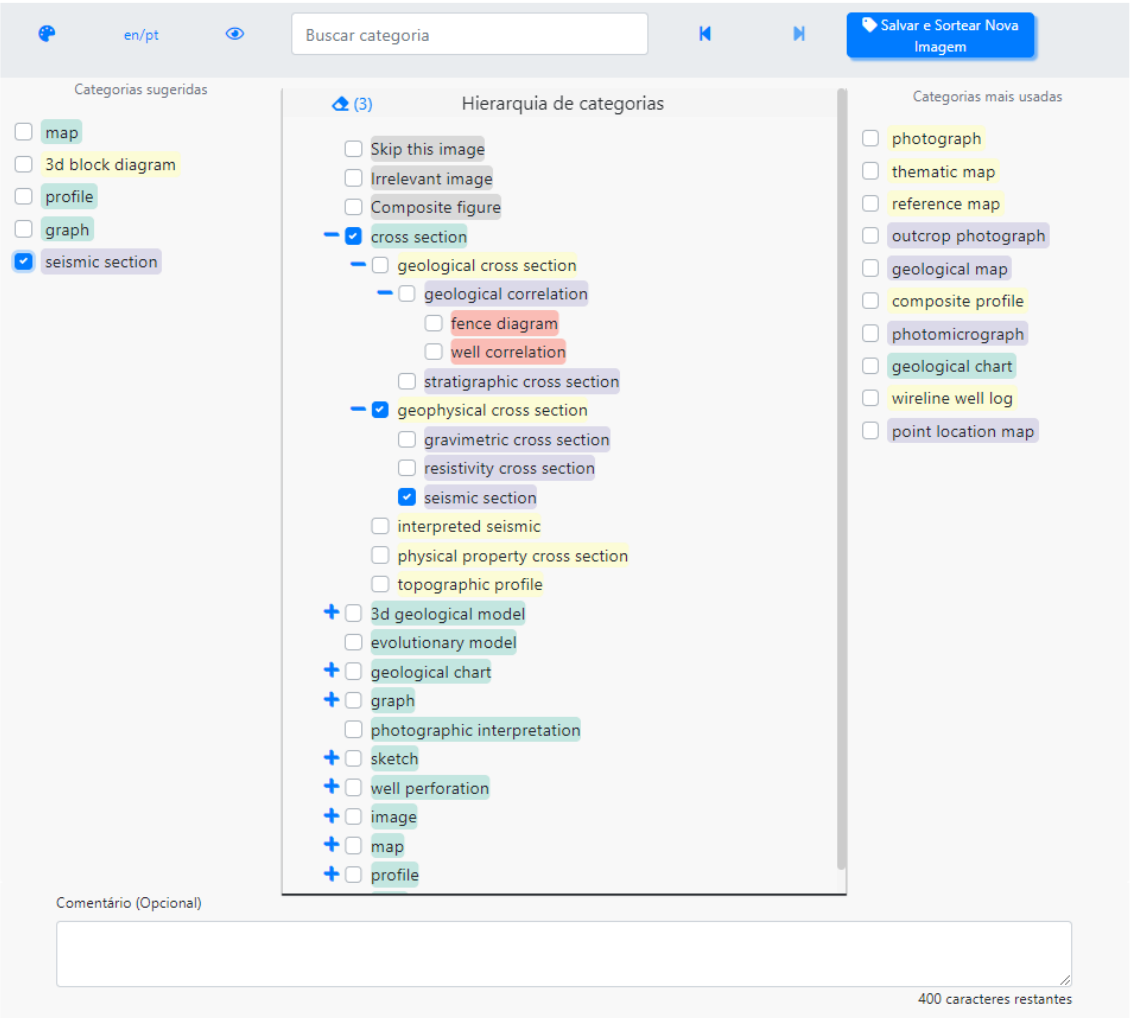
help bar, there are three columns (Figure 4.6) containing non-disjoint lists of selectable categories. On the left are the categories suggested by the Flask category prediction server (Section 5.6), sorted from top to bottom in descending order of probability. In the center is the complete semantic hierarchy of visible categories organized in levels (separated by colors from the color palette chosen by the user). Categories which contain sub-categories in the hierarchy are expandable/maximizable by clicking on the plus "+" signs to the left of the category names. On the right there is a list of the categories most used globally by all users, ordered from top to bottom in descending frequency of usage.

The annotation page has a responsive design which adapts to different window sizes, considering the ease of use in browsers for mobile devices. This is done through the use of breakpoints in CSS language, so that the image and the semantic hierarchy of categories are viewed as vertically-distributed blocks (instead of horizontal).

4.3.3 My Labels Page

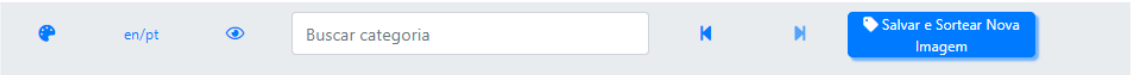
The My Labels page (Figure 4.7) provides an interface for browsing through the annotations made by the user so far. In this interface, the user scrolls through a vertical list of annotations, that is, images associated with the categories saved in the annotation process, which is ordered by the modification date. In addition to the categories selected for

Figure 4.4: Right Block of the Labelweb Single-image annotation interface contains a help bar, three categories lists (recommended, full expandable hierarchy and most used categories) and comment input box.



Source: The author

Figure 4.5: Help bar of the Labelweb Single-image annotation interface contains many operations to facilitate the annotation, including interface language and color palette switching and image traversal buttons.

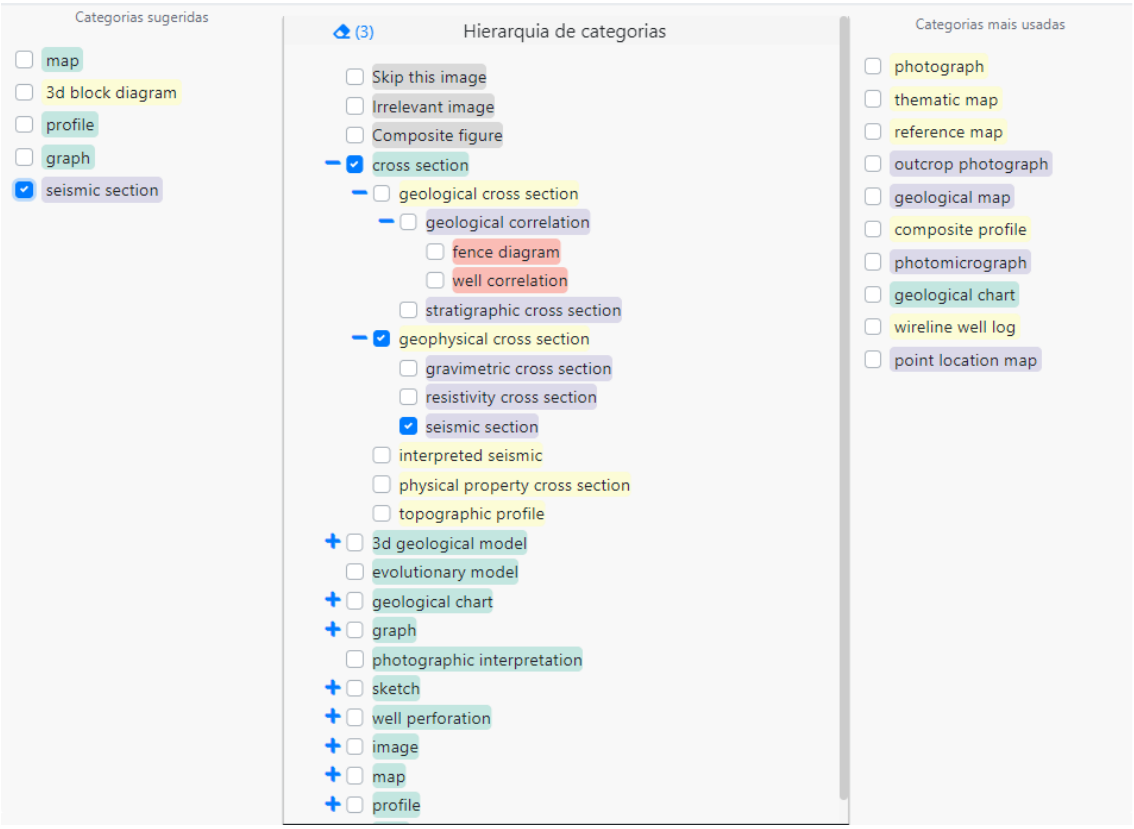


Source: The author

each image, modification date information and annotation comment, if any, are displayed.

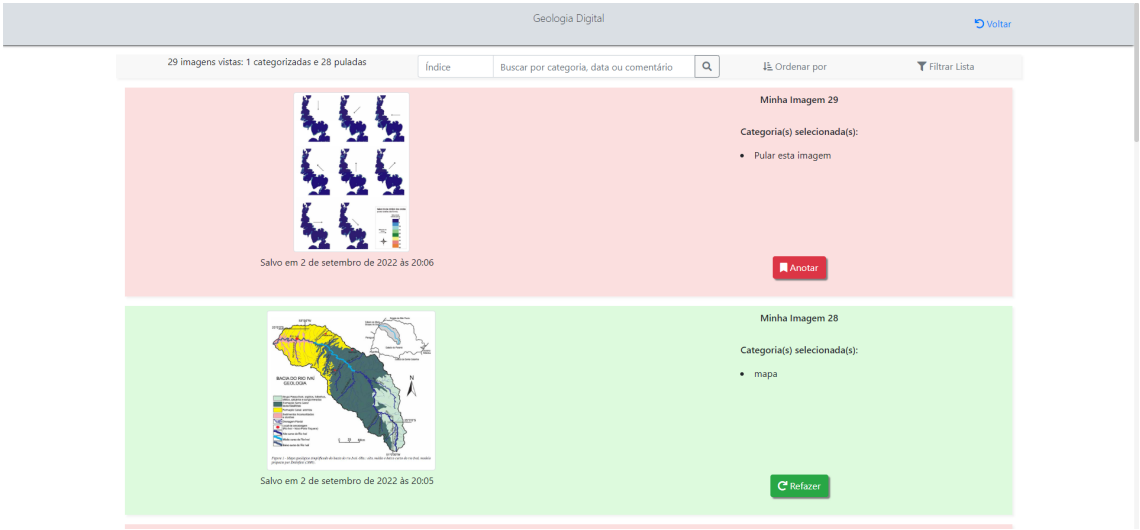
The interface also allows the user to redo an annotation saved in the system. Annotations of images with the special category “I prefer not to comment on this image” are highlighted in red on this page to encourage the user to return to them and redo their annotation.

Figure 4.6: Columns of the Labelweb Single-image annotation interface contains three categories lists: recommended categories for the current image, full expandable hierarchy and most used categories.



Source: The author

Figure 4.7: My Labels Interface for navigating between images annotated by the user.

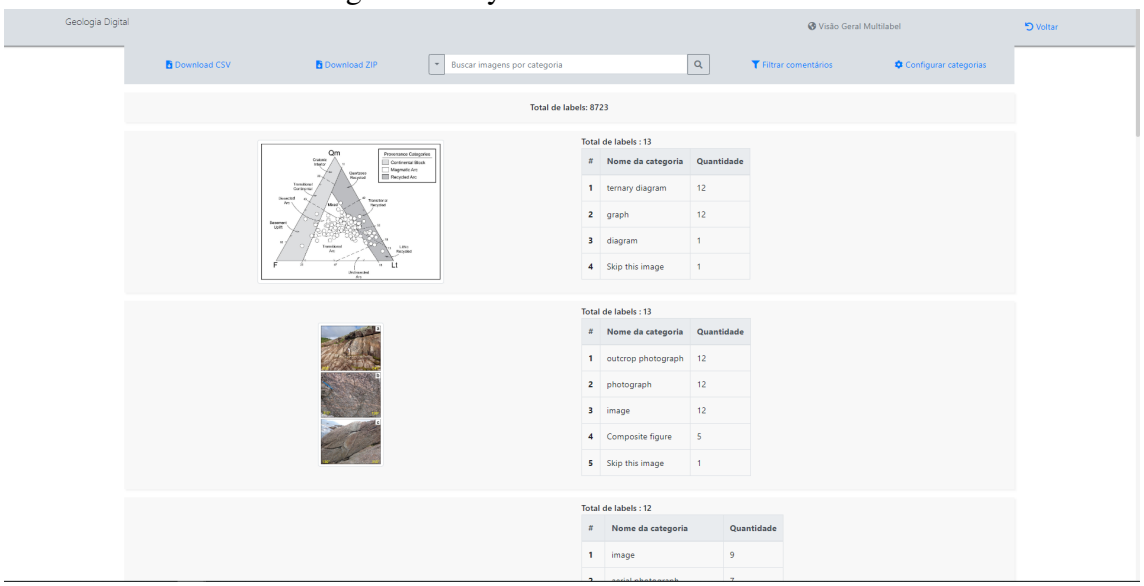


Source: The author

4.3.4 Overview Page

The Overview page is accessed only by superusers and system administrators via the “Visão Geral” (Overview) button available in the navigation bar, by validating a

Figure 4.8: System Overview Interface.



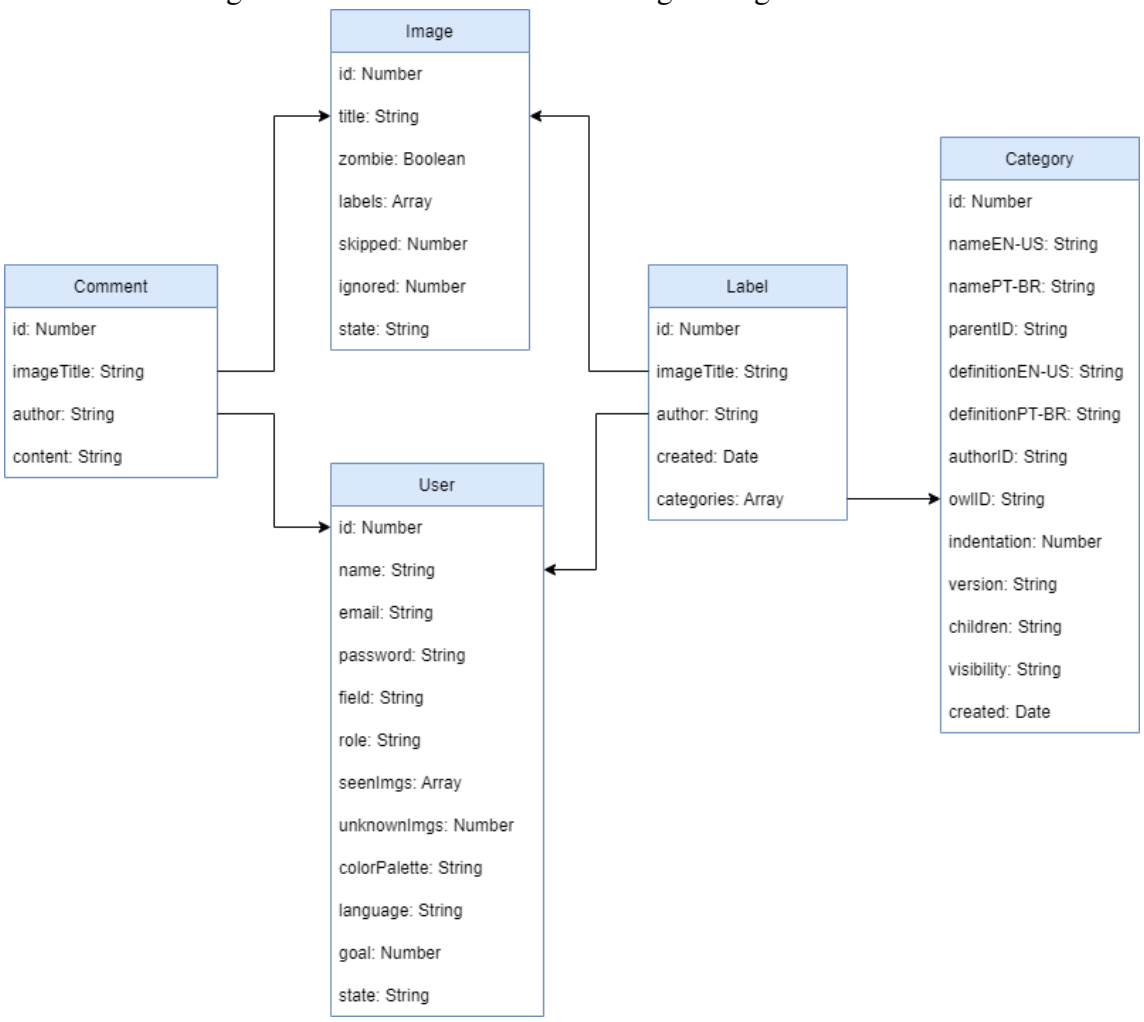
Source: The author

password known to this subgroup of users. The interface of the overview page, as can be seen in Figure 4.8, presents the annotation results of all users of the system, also allowing the extraction of this information in CSV and ZIP format. The superuser can also select the “Configurar categorias” (Configure categories) button to change the categories visible to all users, in addition to searching for specific images by categories.

4.3.5 Multi-image Annotation Page

The multi-image annotation page is composed of three layers of interface elements. As can be seen in Figure 1.2, the first layer of the multi-image annotation page, at the top, is the navigation bar (Figure 4.2). The second layer, just below, permits the modification of the multi-image query type selected by the user, the visualization of the category of the current interaction, a button to skip the current interaction and a single button to save and advance the annotation. Finally, the third and last layer displays a grid of images dynamically selected and created by the system, whose size is set by the administrator.

Figure 5.1: Database models for single-image Annotation.



Source: The author

5 BACK-END LAYER

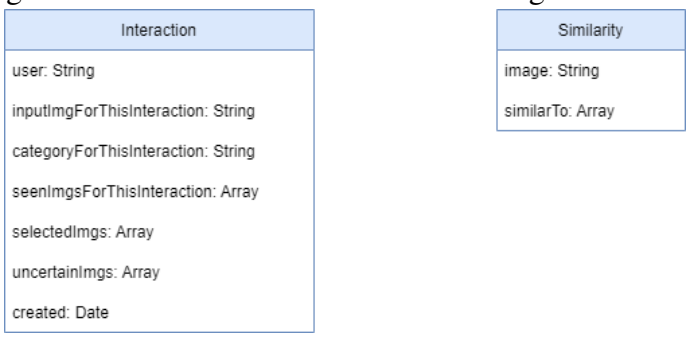
5.1 Database Models

The annotation system uses a non-relational document database, MongoDB. MongoDB is an open-source tool, which provides support for JSON storage and is very commonly used in Node.js back-ends. Hence, the database modeling for single-image annotation utilizes the Mongoose tool, and the database model shown in the Figure 5.1.

Each annotation is stored as a Label document in the database. The Label identifies a list of Categories which are associated with a particular Image. It also includes information o the Author of the annotation. An optional Comment document can be used to store a user-defined comment about a particular Image.

Additionally, in order to develop the multi-image annotation mode, the multi-image

Figure 5.2: Database Models for multi-image annotation.



Source: The author

collections are represented in the models displayed in the Figure 5.2. The Similarity collection allows the importation of precomputed image similarity information into the database and the Interaction collection stores all the Multi-image annotation information.

The similarities collection (right-hand-side of Figure 5.2) stores the pre-computed information that represents the degree of similarity between images served through the multi-image interface. For each image acessible through the multi-image interface, the database stores a similarity vector which contains similar images sorted by the similarity degree in a descending order. Furthermore, the interactions collection (left-hand-side of Figure 5.2) stores the result of the multi-image annotations. So, each user interaction with the multi-image annotation process may contain: an input image (when selecting images based on visual similarity, as discussed in Section 4.2.2), a set of images shown to the user, a category, and the images selected by the user.

5.2 Labelweb Back-end Management Subsystem for Single-image Annotation

In order to guarantee an efficient and dynamic image selection procedure for the annotation of a varied dataset, the system's back-end provides image and user management subsystems. These are modeled through the use of state machines, presented below.

5.2.1 User State Control

The user state control happens through the following states: *warm-up*, *validating*, and *valid* (Figure 5.3). From the starting point of creating a new user, every user goes through every state, sequentially, starting at the warm-up state. This control is performed with the objective of determining the subset of images that will be used to select the

Figure 5.3: State machine defining the user state control.

Source: The author

next image that a user will annotate. Every single-image annotation-page request to the back-end selects a new image, drawn with an uniform probability, from a particular subset of images defined by the user state.

The *warm-up* state is used to make the user familiar with basic annotation features and the system's layout. During this state, the user annotates a finite and small subset of images from the database. The size of this subset is defined by the system administrator in the system settings file, using the key *numberOfImagesBeforeValidation*. After annotating the required number of images, the user is considered "warmed up" and moves on to the *validating* state.

During the *validating* state, the user annotates a hand-selected subset of images, called "special images," that has also been already annotated by a domain specialist. Thus, it is possible to estimate the level of familiarity of the user with the image content and ontology classes in the specific dataset begin annotated, by comparing the user's annotations with the specialist's annotations. As such, the quality of the annotations by a specific user can be estimated, comparing the semantic distance between the annotations.

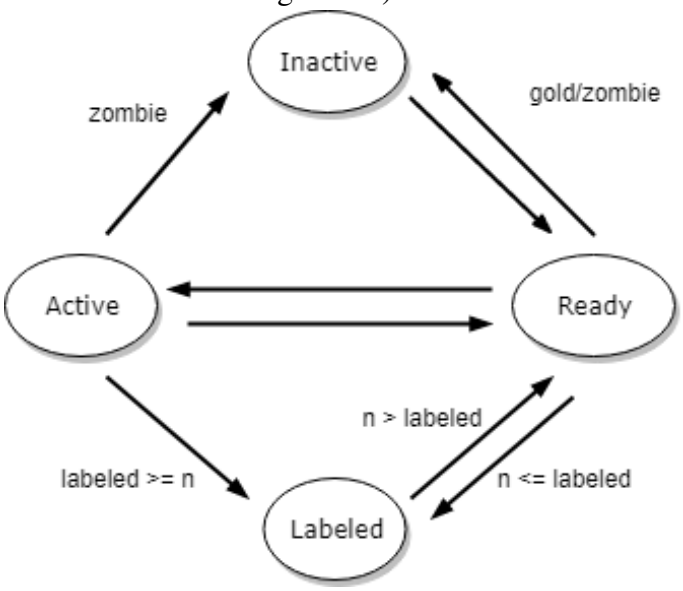
Finally, after annotating all images in the "special images" subset, the user enters the final *valid* state. At this state, any image marked as "active" in the system can be randomly selected for the user to annotate. Hence, they can annotate any image in the database as long as there are non-annotated (active) images available.

5.2.2 Image State Control

The state of each image in the system's database is controlled by the state machine shown in Figure 5.4, containing the following states: *inactive*, *ready*, *labeled*, and *active*. The subset of images at a particular state is used to control the selection of images provided to the user dynamically, for annotation.

The single-image annotation process allow the user to annotate a single image at a time. Furthermore, each image in the dataset is annotated by potentially more than one user, as controlled by the *labelsPerImage* configuration option (Section 5.3). This

Figure 5.4: State machine defining the image state control. An image can assume the states: *active* - it can be randomly drawn for the user annotation interface, *ready* - it can be included in the active state, *labeled* - it was already annotated by *n* users as defined by the administrator in the config.js file (labelsPerImage variable defined in Section 5.3), and *inactive* - it cannot be included in the active state. This subsystem state machine is also defined by the transitions: *zombie* - an existing image in the database that cannot be found in the data folder served by the back-end (imagesDirectory variable defined in Section 5.3), *gold* - an existing image in the database that was previously annotated by a specialist for user validation, and comparison of *n* - constant minimum number of times that an image must be annotated - and *labeled* - number of times the image was annotated by an user (*labels* array size attribute defined in Figure 5.1).



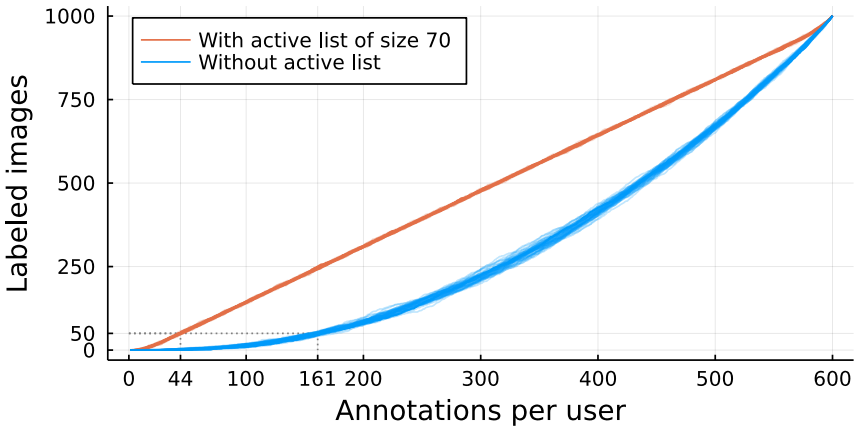
Source: The author

redundancy in the annotations may be used to improve the quality and robustness of the annotated dataset, for example, by discarding images where different users have significant disagreement on the annotated classes. An image is only considered as correctly *labeled* when it has received the required number of annotations (from different users) as defined by the value of *labelsPerImage*.

Note that, if the system randomly selected images for users to annotate from *all* available images, it would take a large number of annotations before reaching a particular target goal of labeled images. For example, in a configuration with 1000 images in the database, being annotated by 5 users simultaneously, and with a requirement of *labelsPerImage* = 3, on average it would take **over 161 annotations per user** before reaching a target goal of 50 labeled images (each annotated by 3 users).

Therefore, in order to speed up the rate at which labeled images are obtained, the annotation system back-end uses a "sliding window" of images that are *active*, such that only images in the active set are considered in the random image selection that occurs

Figure 5.5: Evolution of the number of labeled images with and without the use of an active images list. Graphs obtained through random simulations of the annotation process.



Source: The author

when a user makes an image annotation page request. The size of the active image list is controlled by the *numberOfActiveImages* configuration option.

When the system randomly selects images for users to annotate only from the *active* images list, a particular target goal of labeled images is reached significantly faster. For the same example as before, in a configuration with 1000 images in the database, being annotated by 5 users simultaneously, with a requirement of *labelsPerImage* = 3 and with *numberOfActiveImages* = 70, on average it takes **only 44 annotations per user** before reaching a target goal of 50 labeled images (each annotated by 3 users). Figure 5.5 illustrates the performance gain in using the active image list.

In order to store the current annotation, the image management subsystem verifies if the corresponding image holds the number of annotations above *labelsPerImage*, so that the image becomes a *labeled* image. When this happens, the image management subsystem removes the current image from the sliding window of selectable (*active*) images and adds a new *ready* image to the set.

As well as the control of the active images sliding window, it is necessary to control the images deleted from the images folder, which are kept in the database in the *inactive* state, in order to guarantee that they will not be selected for display. Furthermore, control of the "special images" (Section 5.2.1), also called *gold* images, is necessary during the *validating* user state. During this state, images annotated by specialists are selected aiming to "validate" an user (Section 5.2.1). Meanwhile, during any other user state, the special images set is considered as *inactive*, disabling the system to insert these images in the sliding window.

Another exception scenario controlled by this subsystem is the "*zombie*" transition.

In this exception, during the lifespan of an *active* or a *ready* image from the database, that cannot be found in the image folder and has not achieved the minimum number of annotations required, becomes *inactive*. In this way, the image along with its metadata is still present in the database and subject to be annotated once the image content is available in the directory again.

5.3 Administrator Settings

The following list represents the configuration settings, which are available to the administrator user for customization in the configuration file `config.js`. This customization defines the system behaviour on the back and front-end levels, realized during the software loading time, so any change to the config file must be done before initializing the system.

Config.js File Attributes:

- **imagesDirectory:** Address of the local directory where the images for single-image annotation are located.
- **flaskAddress:** Remote address for communication with Flask server (Section 5.6) for automatic category suggestion.
- **dbname:** MongoDB managed database name.
- **port:** Port number used to serve the annotation system on the server machine.
- **excludeFromPopularList:** Set of OWL identifiers of the categories that should not be included in the popular categories list in the interface.
- **defaultInvisibleCategories:** Set of OWL identifiers of the categories that will be made invisible in the interface by default (the visibility of each category can be changed individually in the Visibility configuration page with restricted super users access (Section 4.3.4)).
- **usersToIgnore:** Set of users ignored during the process of exporting labels in ZIP format (used to ignore “test” users).
- **saveTreeState:** Permission for the system to make the categories hierarchy list state persistent between each Single-image annotation page request during the user session, the categories hierarchy list state defines which dynamic elements are expanded and collapsed.
- **userGoal:** Total number of image annotations requested from each user.

- **numberOfActiveImages:** Number of images in sliding window for dynamic next image selection for user annotation.
- **labelsPerImage:** Number of annotations required for an image to transition from the *active* state to the *labeled* state.
- **specialImages:** Set of filenames of "special images" that are used in user validation.
- **numberOfImagesBeforeValidation:** Number of images selected for a user to annotate before transitioning from *warm-up* to *validating* state.
- **numberOfImagesToValidate:** Number of images annotations for a user to transition from *validating* to *valid* state.
- **adminPassword:** Password used on the frontend to grant restricted superuser access to the System Overview page.
- **allowCategoryEdition:** Permission to allow creating, editing and removing categories (for authorized users only).
- **routeKey:** Key used to restrict access to the new user registration page.

5.4 Ontology Parsing

According to Smith (2012), a Basic Formal Ontology (BFO) is an artifact designed for purposes of ontological engineering. In *Knowledge Organization Systems* (STUDER; BENJAMINS; FENSEL, 1998), an ontology is a formal and explicit specification of a shared conceptualization. It must be explicit, in order to support the application of computational methods to extract new information through reasoning. The conceptualization must be shared because it should represent the community consensus, thus raising the chance of reuse and the capacity of distinct models integration.

The formal ontology in OWL file format used during the Labelweb development integrates data of the specific scientific domain of Geology (Chapter 6), and it defines the categories hierarchy loaded automatically to the front page of the single-image annotation page.

In addition to the Labelweb system images directory, the file `parsed.txt` is also required as input for the system startup. This file is the result of the ontology *parsing* process, carried out through a script in Python language with the support of the *owlready2* module (for reading the ontology document in OWL format) and *hashlib* (to store a version attribute determined through the MD5 checksum of the ontology OWL file).

The OWL file parsing script, `owlpy.py`, receives as input a file in OWL format and a set of root node identifiers, which determine the extracted branches of the formal ontology hierarchy in the resulting `parsed.txt` file.

Each ontology category extracted in the output file contains the following information, line by line:

Category representation in `parsed.txt`:

- **1st line:** Root or Non-root: Indicates whether or not the category is a forest root.
- **2nd line:** identifier: a unique identifier of the OWL document of the parent category of the current category – when it is a root category, the string "None" appears by default.
- **3rd line:** identifier: a unique identifier of the OWL document of the current category.
- **4th line:** ['string']: a string enclosed in square brackets and single quotes that represents the English label of the current category.
- **5th line:** ['string']: a string between square brackets and single quotes that represents the Portuguese label of the current category.
- **6th line:** ['string']: a string representing the definition of the category in English.
- **7th line:** ['string']: a string that represents the definition of the category in Portuguese.
- **8th line:** string: an English synonym of the current category (can be empty).
- **9th line:** string: a synonym in Portuguese of the current category (can be empty).
- **10th line:** identifier1$identifier2$...identifierN$: a sequence of identifiers separated by the dollar sign that represents the set of children of the current category – when it is a leaf category, this line appears empty.

5.5 Importing Similarities for Multi-image Annotation

The Multi-Image annotation system relies on visual similarity data between images, imported by the script from the parser-images directory. This information can be imported using the import-json-similarities-to-mongo script, from a json file containing, line by line, the following object format:

```
1 {
```

```
    "<image_title>":
    {
        "most_similars":
        {[
            { "path": "<image_1_path>" },
            { "path": "<image_2_path>" },
            ...
        ]}
    }
}
```

In this object, an image title indexes an array of its most similar images, in decreasing order of similarity, under the key *most_similars*. The sorted images are ordered in a list containing objects indexed by the keyword *path* and containing the respective image path of the similar image.

From the JSON file containing similarity data, the data is imported into the database through the parser-images/import-json-similarities.to-mongo.js script. This procedure can perform, through the use of the *command-line flags -overwrite* or *-incremental*, a forced import, erasing and rewriting the similarity collection of the database, or incrementally, storing only data of similarities that do not yet exist in the corresponding collection.

5.6 Flask Server for Category Suggestion

In parallel with the Labelweb server, a category prediction module, developed in Python/Flask, is responsible for receiving a remote address from an image and returning a set of suggested ontology category identifiers. After performing the asynchronous request to the prediction module during the loading of the single-image annotation page, the set of categories returned to the system is displayed in the interface as an annotation suggestion for the respective image (Figure 4.6).

For this purpose, the prediction module must be properly deployed and its correct address defined, in the `config.js` configuration file, in the *flaskAddress* settings attribute. This communication uses a standard JSON API and any third-party category-suggestion service could be used. Details of the Flask server used during the development of Labelweb is outside the scope of this work.

6 EXPERIMENTS

The Labelweb system was developed in the context of a Cooperation Agreement and Research Project between Petrobras and the Federal University of Rio Grande do Sul (UFRGS). Development started in April 2019, and the system was continuously improved through the inclusion of new features during the course of the project. This was done based on user feedback and experiments realized by the Institutes of Geosciences (IGeo) and Informatics (INF) of UFRGS, and Petrobras (EXP and CENPES). The development of this tool had a very large intersection with the activity of building the ontology in the domain of the project, considering the images available in the proposed database. Hence, the version of the system deployed for experiments with students and specialists extracted the categories' semantic hierarchy from the GeoImageOntology[1] project.

6.1 Experiments with students

The first large-scale use of the Labelweb system took place during December 2019 and January 2020, when IGeo/UFRGS volunteers annotated 2637 images from the database. A total of 64 registered users generated 5877 annotations for these images (an image can be annotated in more than one category), which served as the basis for the first automatic classification experiments in the project (training Convolutional Neural Networks).

6.2 Experiments with specialists

From July 9th 2020 until September 8th 2020 (approximately 9 weeks), a team of ten Petrobras specialists used the Labelweb tool hosted at UFRGS to annotate 8,717 images from our database. All annotations made by the Petrobras team used the single-image annotation mode. Furthermore, each image was annotated by at least three users, for redundancy, generating 31,075 annotations. More than 4,000 comments were attached to the annotations by the users. Since these were considered higher quality annotations, the specialists' annotations replaced the initial annotations, made by IGeo/UFRGS volunteers, in the project classification experiments. Finally, the Labelweb tool was installed on

[1] <https://github.com/BDI-UFRGS/GeoImageOntology>

CENPES servers in December 2020, with support given by the INF/UFRGS team, for use in Petrobras' confidential image databases.

6.3 Multi-Image Interface Evaluation

The multi-image annotation functionality, where users can categorize many images at the same time (Section 4.2.2), was developed in the last few months of the project. This annotation process, however, is less detailed, as the user only assigns a single pre-defined category to the images. On the other hand, in this way it is possible to annotate a larger number of images in a shorter period of time when compared to the single-image interface. For example, in one of our experiments, a specialist went through 1,720 images in an interval of 1.5 hours using the multi-image interface, categorizing each of these images as belonging or not belonging to the "seismic cube" category. In particular, before this experiment, the dataset had only 36 images annotated as a seismic cube, as this is a category where it is difficult to find publicly available images. After the 1.5 hour experiment, the number of images increased by 50%, to 54 images annotated as a seismic cube. This experiment used the image-selection procedure based on the probabilities of a pre-trained classifier (Section 4.2.2.2).

The multi-image annotation mode based on image similarity (Section 4.2.2.1) was evaluated in an experiment in February 2021, where the definition of visual similarity between images was given by the cosine distance between feature vectors generated by a Neural Network. According to the results of the experiment, the images selected in this way are indeed quite visually similar, however this similarity does not have a good correlation with the image categories. On average, only 2 out of every 8 similar images actually belonged to the same category. Thus, the multi-image annotation mode based on the confidence of a pre-trained classifier proved to be more adequate, as it uses the knowledge of a pre-trained classifier to differentiate between existing categories. The disadvantage of this interface, however, is that the user is limited in choosing only categories that are known to the classifier.

7 CONCLUSION

This work described Labelweb, a customizable web-based image annotation tool developed in the context of a Cooperation Agreement and Research Project between Petrobras and UFRGS, in order to facilitate the generation of the labeled dataset used by the image classification algorithms developed in the same project.

The resulting tool supports many important features gathered through the feedback from multiple deliveries. Many features aim to minimize the cost of manually selecting categories that correspond to the visual content in a randomly picked image , and better attend the users' needs. Furthermore, the entire system design and features took full advantage of other steps of the project. For instance, the categories hierarchy provided by the system is loaded automatically through the ontology crafted by researchers in this domain, and the category recommendation feature leverages the neural network model provided by the team. This set of specialized customized functionalities was necessary to accelerate the annotation process and generate better quality data. During an experiment of nine weeks, a group of ten Petrobras specialists created 31075 annotations and more than 4 thousand comments with those annotations, which were later used to create and polish machine learning models.

Still, looking into speeding up the annotation process and optimizing the user interaction, the multi-image annotation mode was developed. Instead of providing an image and asking for the user to pick a detailed selection of many categories (traditional single-image annotation mode), in the multi-image annotation mode the system asks the user to associate multiple images to a pre-selected category. During an experiment, this annotation mode proved to be much faster than the previous one. However, image selection in this mode relies on the quality of a pre-trained classifier, and this could lead to worse quality data in an extreme case.

Through the design, development and experiment process of the Labelweb system, the goal of generating a labeled dataset also meant streamlining a simple task such as annotating an image, which proved to be much more challenging than expected. In order to optimize quality and quantity of annotations, this process took into account the support of multiple simultaneous remote users while attending a variety of quality of life and customization features, and different modes of interaction. We believe the resulting system met all expectations and represented an essential component in the success of the research project where it was inserted.

Acknowledgements

We would like to thank everyone that made the development of the Labelweb system possible at UFRGS: Prof. Viviane P. Moreira, Prof. Mara Abel, Prof. Cassiana Michelin, Prof. Bruno Castro da Silva, and the students Luiza Maggi, Luan Fonseca Garcia, Felix Eduardo Huaroto Pachas, Francisco Bento, and Bruno Firnkes. We also thank all the collaborators at Petrobras EXP and CENPES, who provided invaluable feedback during the development process.

REFERENCES

DAYLEY, B. **Node. js, MongoDB, and AngularJS web development**. [S.l.]: Addison-Wesley Professional, 2014.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [s.n.], 2009. p. 248–255. Available from Internet: <https://doi.org/10.1109/CVPR.2009.5206848>.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **2009 IEEE conference on computer vision and pattern recognition**. [S.l.], 2009. p. 248–255.

DENG, J. et al. Scalable multi-label annotation. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. [S.l.: s.n.], 2014. p. 3099–3102.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. **International journal of computer vision**, Springer, v. 88, n. 2, p. 303–338, 2010.

HCI Lab, Karlsruhe Institute of Technology. **sloth 1.0 documentation**. 2014. <https://sloth.readthedocs.io/>. Accessed: 2022-09-16.

HEARTEX, I. **LabelMe, the open annotation tool**. 2018. <http://labelme.csail.mit.edu/>. Accessed: 2022-09-16.

KAWAMURA, R. **RectLabel - An image annotation tool to label images for bounding box object detection and segmentation**. 2022. <https://rectlabel.com/>. Accessed: 2022-09-16.

Labelbox, Inc. **Labelbox - The data engine for AI**. 2022. <https://labelbox.com/>. Accessed: 2022-09-16.

MADEYSKI, L.; SOCHMIALEK, M. Architectural design of modern web applications. **Foundations of Computing and Decision Sciences**, Poznan, Poland: Institute of Computing Science, Technical University of . . . , v. 30, n. 1, p. 49–60, 2005.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv**, 2018.

RUSSELL, B. C. et al. Labelme: a database and web-based tool for image annotation. **International journal of computer vision**, Springer, v. 77, n. 1, p. 157–173, 2008.

SMITH, B. On classifying material entities in basic formal ontology. In: **Interdisciplinary Ontology: Proceedings of the Third Interdisciplinary Ontology Meeting**. [S.l.]: Keio University Press, 2012. p. 1–13.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. Knowledge engineering: principles and methods. **Data & knowledge engineering**, Elsevier, v. 25, n. 1-2, p. 161–197, 1998.

TZUTALIN. **LabelImg**. 2018. <https://github.com/heartexlabs/labelImg>. Accessed: 2022-09-16.

W3C, O. W. G. **OWL 2 Web Ontology Language Document Overview (Second Edition)**. 2012. <https://www.w3.org/TR/owl2-overview/>. Accessed: 2022-09-13.

WONG, K. H. **OpenLabeler**. 2022. <https://github.com/kinhong/OpenLabeler>. Accessed: 2022-09-16.