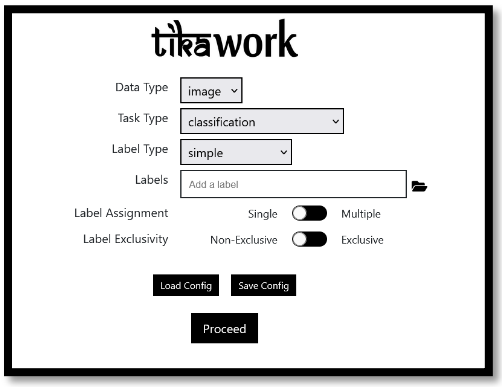
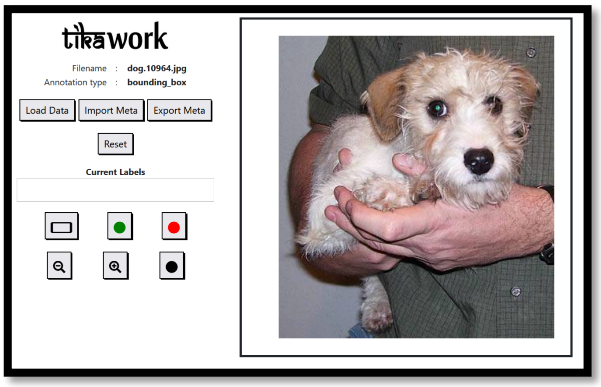
tika-work Annotation Tool

Introduction

tika-work is designed as a lightweight, modular open-source software that can support data annotations in computer vision and NLP domains. It is released as a web-app as well as for multiple plaforms like Mac OS and Microsoft Windows. tika-work is built as a pure javascript application, more specifically reactjs. This allows it to be truly portable by being entirly based in the web browser.

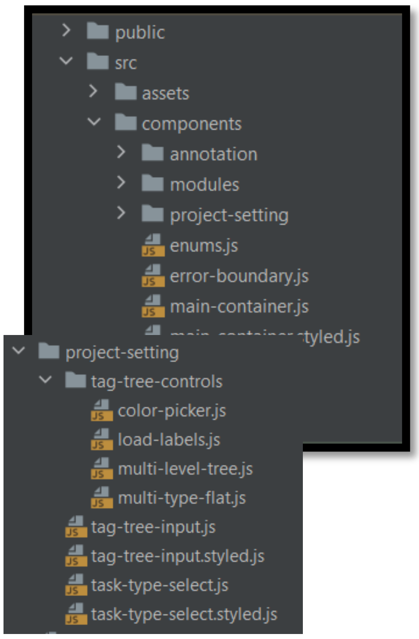
It has two main screens: annotation task config initial params and annotation process. The first screen allows you to select the task type and configure the labels options dinamically depending on the selected task and levels. The second has the controls and user interface for make the annotations, also it depends of data and task types.


Data Type: image
Task Type: classification
Label Type: simple
Labels: Add a label
Label Assignment: Single / Multiple
Label Exclusivity: Non-Exclusive / Exclusive


Filename : dog.10964.jpg
Annotation type : bounding_box

The project has a module folder with an independent component for every existen annotation. The user interface of both screens load its options from the components inside the folder. Global settings is create with the config files inside every module that contains the priority, data-type, task's name, mode's list and level's list.

annosmart tikawork

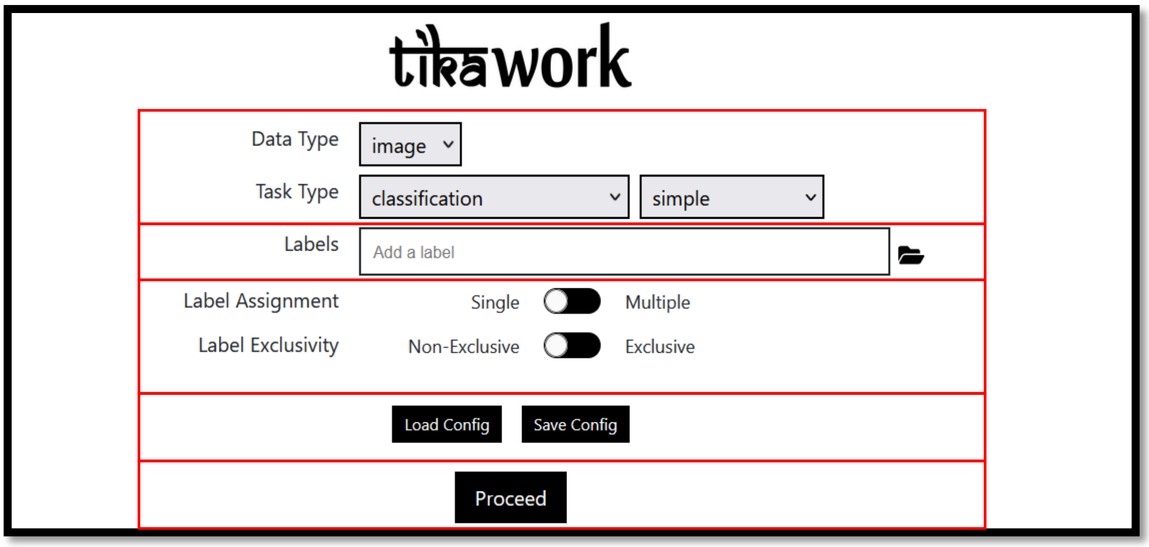

```
> public
v src
  > assets
  v components
    > annotation
    > modules
    > project-setting
      enums.js
      error-boundary.js
      main-container.js
      main-container.styled.js
v project-setting
  v tag-tree-controls
      color-picker.js
      load-labels.js
      multi-level-tree.js
      multi-type-flat.js
    tag-tree-input.js
    tag-tree-input.styled.js
    task-type-select.js
    task-type-select.styled.js
```

The source code structure has three main folders: **"modules"** contains all compoments that will be loaded, **"project-settings"** with the controls for the first screen, **"annotation"** with menus panel, data views and common controls of second screen.

The App.js is in charge of scanning the config files on modules folder, create the whole data structure and pass it to the setting screen

Selection of annotation task type

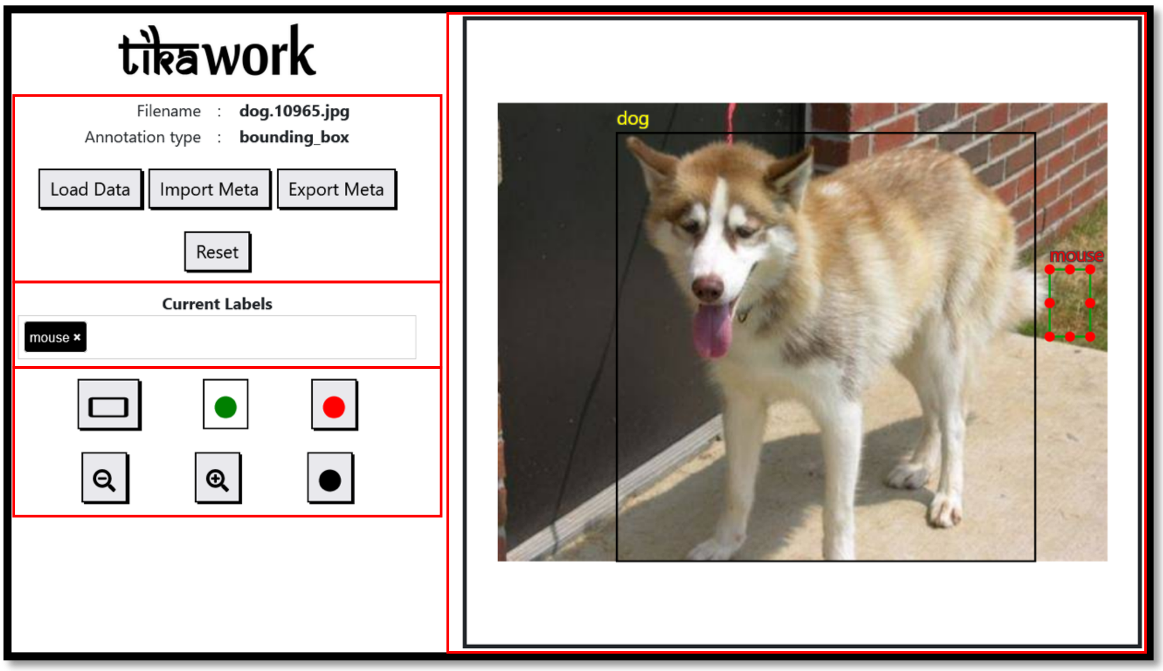
Data structure from configuration files has a tree format used to fill the select boxes. The options for Data type, Task type, Modes and Levels of labels are generate dynamically from the tree. **"TaskTypeSelect"** is the main component of the screen and contains rest of child components. **"TagTreeInput"** manage the label input component according to selected mode and level: **"LoadLabel"** is an input to add labels for single level, **"MultiLevelType"** is a recursive component that allows a tree of Attributes with a label input control in every last layer, and **"MultiTypeFlat"** that can be use with single or multi-level plus to select colors for annotations.


Data Type	image	
Task Type	classification	simple
Labels	Add a label	
Label Assignment	Single / Multiple	
Label Exclusivity	Non-Exclusive / Exclusive	

The screen has extra options: Label Assignment and Label Exclusivity according to the selected task type. Also allows load and save the current configuration on JSON format. All information collected from select boxes and label inputs are pass to the next screen when it is process.

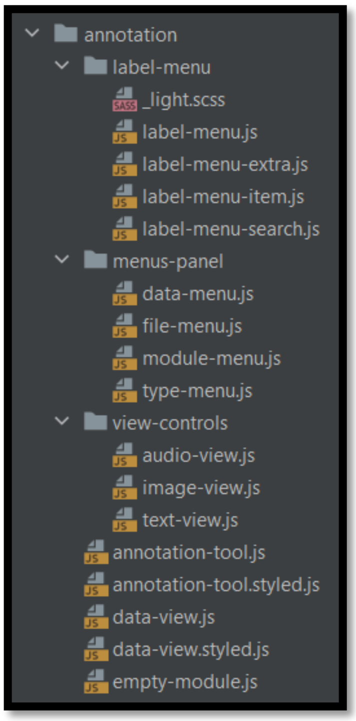
Annotation task

This screen is responsible for loading the specific module for the selected annotation type. It also allows loading the data file to be processed and displaying it according to the data type defined for the module. It has predefined controls and tools for each data type and depending on the declared configurations. It also has a panel that is defined and generated



Filename : dog.10965.jpg
Annotation type : bounding_box

by the loaded module. The screen allows you to import and export the annotations made in different formats as defined by the module.



```
annotation
  label-menu
    _light.scss
    label-menu.js
    label-menu-extra.js
    label-menu-item.js
    label-menu-search.js
  menus-panel
    data-menu.js
    file-menu.js
    module-menu.js
    type-menu.js
  view-controls
    audio-view.js
    image-view.js
    text-view.js
  annotation-tool.js
  annotation-tool.styled.js
  data-view.js
  data-view.styled.js
  empty-module.js
```

The main component is “**AnnotationTool**” handling all the general data and events between the rests of the components. It receives the data collected in the first screen, including the address of the annotation module, which is loaded and validated, binding it with the processed information and the expected events.

“**DataView**” component is in charge of loading and relating the data viewer according to the type of data. There are currently three data viewers, for images, text and audio-video. Each have their own worker methods, custom tools and fire their events accordingly.

Image Viewer is based on SGV, in addition to displaying the image it handles the zoom and its movement on the screen. Supports the active module to add a layer to the SVG that allows drawing over the image. The majority of events that are trigged are those related to the mouse.

Text Viewer is based on HTML, allows to format the text and create more complex structures on top of the text.

Audio Viewer combine several audio and video players, it also includes some JS libraries such as “WaveSurfer” for audio segmentation and visualization.

“**LabelMenu**” implements a context menu that can be displayed on any of the viewers. This menu handles the labels loaded on the first screen, and nests them based on the selected levels and types. It is generated recursively using as base “**LabelMenuItem**” and allows searching with “**LabelMenuSearch**”. More custom options can also be added from the active module with “**LabelMenuExtra**”.

Left panel of the screen has four menus that are generated and displayed according to the active module and the corresponding data viewer.

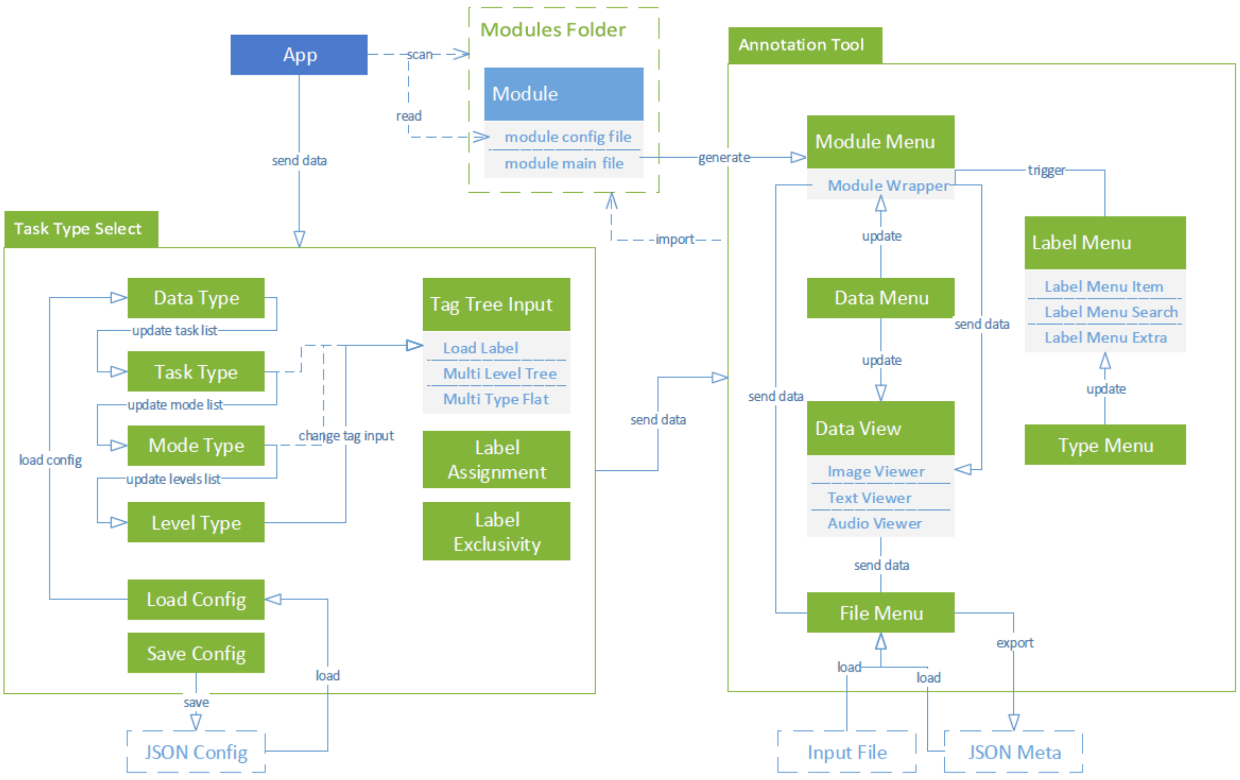
“**FileMenu**” handles loading the data file to annotate, which is passed to the viewer and active module. It allows importing previous annotations and exporting the current one in the formats available by the current module. It also has a button to reset the annotation in process.

“**TypeMenu**” is only available when the multi-type option is selected. Allows you to select the type and updates the context menu with the corresponding options.

“**MuduleMenu**” is a wrapper for display the current module menu. Allows catch any error trough by the loaded module.

“**DataMenu**” has the tools inherent to the type of data selected and depending on the configuration established by the active module. Fired events are primarily handled by data viewers.

App Architecture



App
scan
read
send data
Modules Folder
Module
module config file
module main file
generate
import
Annotation Tool
Module Menu
Module Wrapper
trigger
update
Label Menu
Label Menu Item
Label Menu Search
Label Menu Extra
Data Menu
send data
update
Type Menu
Data View
Image Viewer
Text Viewer
Audio Viewer
send data
File Menu
export
load
load
Input File
JSON Meta
Task Type Select
Data Type
update task list
Task Type
update mode list
Mode Type
update levels list
Level Type
change tag input
Tag Tree Input
Load Label
Multi Level Tree
Multi Type Flat
Label Assignment
Label Exclusivity
load config
Load Config
Save Config
load
save
JSON Config