Grazioso Salvare

Animal Shelter Dashboard

# Purpose

The purpose of this project is to provide an interactive dashboard for the AAC animal shelter database with a focus on finding service animals appropriate for jobs such as Water, Mountain, Wilderness or Disaster Rescue as well as Individual Tracking.

To address this the dashboard includes a paginated, sortable data table that displays animals in the AAC animal shelter database with an option to filter the selection to only animals fitting the criteria from one of the above jobs: Water Rescue, Mountain or Wilderness Rescue, or Disaster Rescue and Individual Tracking.

Also available is a histogram that displays a histogram showcasing the population counts of each breed that fits the filter criteria and an interactive map that shows the location of the selected animal as well as its name and breed.

# Screen Shots

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

# Tools and Libraries Used

## Model

The underlying database was implemented using the NoSQL database library MongoDB because of it does not require strict schemas and can as a result, new fields can be dynamically added without large rewrites.

# View and Controller

The view and controller for this project were implemented in Python 3 because of its extensive documentation and the ease of fast prototyping. They were developed using the PyMongo to interface with the MongoDB database and Dash to implement the web app. Dash was used to develop the web app front end because of its ability to dynamically generate HTML and JavaScript components that interact appropriately using just Python code. This functionality led to components such as the header, the data table, the histogram, and the map interacting with each other and the user seamlessly without having to write any HTML or JavaScript code. This allows a better cohesion between the Controller logic and the View input and output components.

# Implementation

The process of implementing this dashboard began with the model which was imported from a comma-separated values (CSV) into a MongoDB server. Authentication was then implemented with the MongoDB instance such that only server administrators and the webapp user were able to access the data.

A middleware API was then developed in Python 3 with PyMongo to simplify how the Controller would interact with the Model and abstract away PyMongo functionalities that are not relevant to the Controller.

Finally, the Controller and View were developed using Python 3 and the Dash library. This was done by dynamically generating then linking together HTML components into one cohesive HTML document that makes up the web app. Callbacks were then written in Python to provide interactivity between components.

# Challenges

Most challenges encountered during development involve Jupyter Notebook suppressing warnings and the loose typing done by the Python language. One example of such an issue is when neither the histogram nor interactive map were loading with no runtime errors reported by Juptyer Notebook. This had to be diagnosed by inspecting the generated HTML itself, in which the div tags that were to contain both components had a field whose value said the Dash callbacks were not loading. This was remedied by fixing the callback for the histogram which was calling a member function of the Plotly Express module that did not exist. Warnings about such logic errors as this are normally reported by the interpreter or linter however Jupyter Notebook has no such linter extension for Python and needs to suppress all warnings about Dash callbacks to run properly. These issues have apparently been remedied in the new Notebook interface JupyterLab.