Technical Report for FindADogForMe

Matthew Zhao, Samarth Desai, Bryan Yang, Keven Chen, Daniel Ho

Motivation

As shelters become increasingly crowded at alarming rates, it's become clear that the adoption process must be optimized. Our intent in building Find a Dog for Me is to go beyond simple data aggregation in facilitating the adoption of dogs from shelters. As such, we have begun work on creating a unified system tying together dogs, dog breeds, shelters, and dog-friendly activities in an attempt to simplify finding available dogs and establish motivation for their adoption.

User Stories

We were tasked to complete the following user stories:

1. Finish the breeds page
2. Finish the activities page
3. Finish the shelters page
4. Finish the about page
5. Add more non-white dogs of pictures

In order for the first three issues to be resolved, the backend team needed to scrape data from a variety of APIs listed in our about page. The backend team used the requests library for Python and wrote code to interact with and obtain data from each API. From there, the backend team used sample calls relevant to the Austin area in order to get useful data from each API and provided it to the frontend team. Their python code can be found in the /backend folder of the GitLab repository.

Using the scraped data, the frontend team inserted the information into their respective model and instance pages. In addition, the frontend team created carousels for each instance page. The carousels were filled with pictures from both the scraped data and third-party sources. The fourth user story involved updating the basic static .html that was formatted in Bootstrap. The biographies were hard-coded in, but the issues/commits are dynamically pulled from a JavaScript script interacting with the GitLab API. The last user story was simply a matter of replacing the placeholder images that were in place at the launch of the site with different images.

We tasked our developers with the following user stories:

1. Implement an accessible navigation menu for the mobile version of the website.
2. Add a section for holding related model instances to each model instance.
3. Make disaster model information more readable (spacing, line breaks, highlights).
4. Add a carousel to the landing page to switch between different cover images.
5. Create a table holding total GitLab statistics in the About page.

See https://gitlab.com/oceanwall/findadogforme/issues/14 for additional details.

RESTful API

The sources that we relied on to obtain data for our static website were the PetFinder API, TheDogAPI, Dog API, Eventbrite API, Meetup API, and National Park Service API.

We obtained shelter and dog data using the PetFinder API, breed data using TheDogAPI and Dog API, and activity data using the EventBrite API, Meetup API, and National Park Service API.

Our RESTful API, designed using Postman and accessible here (https://documenter.getpostman.com/view/6754951/S11KQJxc), provides four API calls for each model; one API call to obtain data about instances of that model, and then three API calls for collecting data about related instances of the three other models. All API calls are GET requests, and the idea is that they will eventually be validated through the use of a bearer token.

Endpoints were designed such that, building off the base URL of (https://findadogfor.me/api), you would first add the name of the primary model (/breeds) to receive general information about instances of that model, and then if seeking information about related instances of other models, you would append the name of that secondary model behind the name of the primary model (for instance, /breeds/shelters returns a list of shelter items that host dogs of a breed parameter).

For the general API calls to obtain overall data about instances of some model, no parameters are required, but for the API calls related to collecting data about related instances of the three other models, parameters to identify a specific instance of the main model (generally an id string) are necessary.

Models

Dogs

The Dog model has multiple attributes, these are: name, images, the shelter they're located at, size, age, breed, and description.

The model has connections to breeds, shelters, and activities. The model will link to the breed instance corresponding to the current dog's breed, the shelter instance corresponding to the shelter the current dog is located at, and various activity instances located near the shelter that the current dog is located at.

The public RESTful API source supplying the data for this model is the PetFinder API.

Breeds

The Breed model has multiple attributes, these are: name, images, group, temperament, lifespan, height, and origin.

The model has connections to dogs, shelters, and activities. The model will link to dogs that are adoptable of a given breed, shelters that contain the relevant breed, and activities that are suitable for the breed.

The public RESTful API sources supplying the data for this model are TheDogAPI and Dog API.

Shelters

The Shelter model has multiple attributes, these are: name, images, contact information, location, and contained breeds.

The model has connections to dogs, breeds, and activities. The model will link to dogs that the shelter contains, the breeds of those dogs, and dog-friendly activities near the shelter.

The public RESTful API source supplying the data for this model is the PetFinder API.

Activities

The Activities model has multiple attributes, these are: name, images, URL, location, description.

The model has connections to dogs and shelters. The model is linked to dogs that are suitable for this activity, and shelters near each activity.

The public RESTful API sources supplying the data for this model are the National Park Service API, the EventBrite API, and the Meetup API.

Tools

GitLab

GitLab was used as a version control tool, as well as a tracker for various issues. We were able to track the progress of our project based on issue completion and assign various sub-parts of the project to different team members. In the future, we will use this for continuous integration.

Postman

We used Postman to design the RESTful API that we will eventually implement and consume to fill our frontend. It allowed us to specify parameters for the GET requests as well as expected return values, and dictates specifically how all four models are related to one another.

Bootstrap

Bootstrap was used primarily for its CSS in styling the HTML files. We relied on its grid layout system and margin / padding CSS shortcuts to format the majority of our static HTML files. We also utilized its JavaScript components to power the navigation bar and landing page carousel.

Slack

Slack was the main communication channel used by the team. We utilized a WebHooks integration to receive GitLab event notifications about our repository.

jQuery

jQuery was used by BootStrap to power the carousel and navigation bar.

Hosting

Namecheap

Namecheap was used to obtain the domain name and was instrumental in verifying domain ownership to receive an Amazon-issued certificate for HTTPS.

AWS

AWS was used to host the website. We used an S3 bucket to hold our static files and host our static website, CloudFront to serve our website over https and act as a content delivery network, and Route 53 as our DNS web service to direct requests to our website.