Ryan Lindner, Evan Shields

OCAdopt

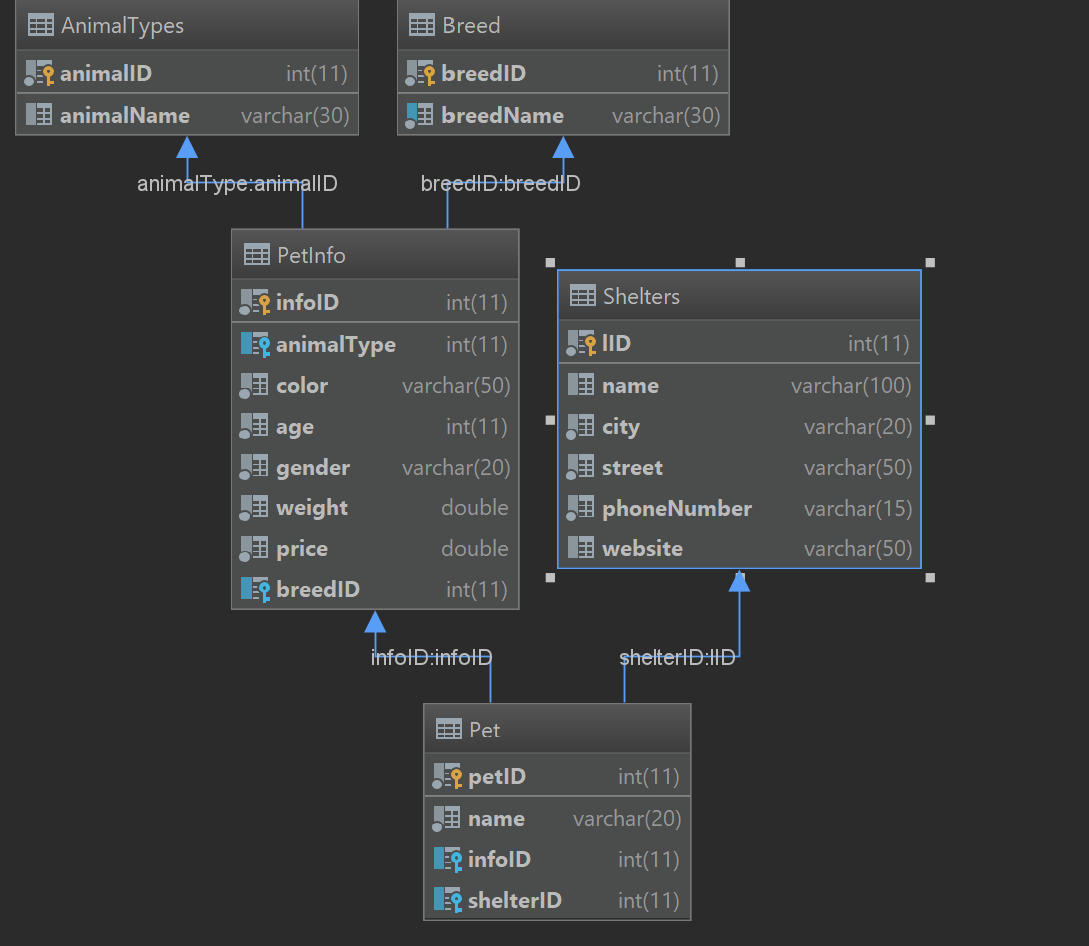
Recently, we were in the market to adopt a new pet. Specifically, we wanted to find the perfect cat to adopt. Naturally, the first action was to Google local adoptions shelters in our area to see our options. We were met with tons of different adoption shelters, such as OC Animal Care, OC Small Paws, MeoowzResQ, and many more. Instead of being able to view all the local cats up for adoption in the area, we had to navigate between several different sites just to see what was available for us.

Because of our experience, we wanted to make this entire searching process much easier for those who are willing to help a pet in need. Our goal is to eliminate the headache of searching through multiple websites by creating our own website which aggregates data from all local adoption shelters and hosts them on one site. Specifically, our project aims to collect data on the adoptable pets in the OC area and display all local pets to the user on the website and give them basic information about the pets themselves and the shelters they are located in.

To implement our project, we utilized several different types of technologies. For our backend, we utilized Google Cloud to host a MySQL microarchitecture server. This allowed us to not have to worry about server size or durability issues. For our frontend, we built the website using a combination of HTML, CSS, and the Bootstrap Framework. HTML is standard hypertext markup language to develop web applications. CSS is a cascading style sheet which allows for easy styling of the HTML web parts. Bootstrap is a framework that is built on top of CSS which provides out-of-the-box, easy to work with configurations of several common web objects. To connect the front-end to the back-end, we utilized PHP, which made MySQL connections to the database and executed basic CRUD statements.

We didn’t publish the website to be publicly available via DNS due to financial reasons. Instead, we ran the program off of a local Apache server via a free program called XAMPP. While the Apache server is running, you can easily connect to the server via your browser and going to localhost/OCAdopt. Consult the README for installation instructions.

For our backend, we created five separate tables to normalize our database into 3NF. The 5 tables are: a Pet table, a PetInfo table, an AnimalTypes table, a Breed table, and a Shelter table. The Shelter table consists of a primary key id for each record as well as basic information about all given shelters. The Breed table consists of all of the possible breeds for our animals in the DB. The AnimalTypes table consists of all types of animals in the DB, such as dogs, cats, hamsters, etc. The PetInfo table consists of the infoID for each given pet as well as their relevant characteristics, like color, age, and gender, as well as foreign keys to the AnimalTypes and Breed tables. Finally, our Pet table consists of petID, name, foreign key to their infoID, and the shelterID of the shelter they are currently located at. Our schema diagram is shown below:



The functionality of our website works as intended. The home page consists of a table that displays the count of all the different types of pets currently in our database. At the top, we have buttons to display different animal tables for Dogs, Cats, and Others. Clicking on any of these buttons redirects the user to a page which shows all of the available pets in the area of that type with pertinent information about them. There is also a Shelters button at the top of the home page, which lists all the shelters and important details about them like address and website. On each animal table, there is a button to add a new pet, update a pet in the table, or delete a pet from the table (probably only needed for the administrator). At the bottom of each page is a button that generates a CSV report for the table.

On the other hand, there was some unexpected complexity regarding basic CRUD functionality in the backend due to normalization. For example, to create a dog, you must first check that the entered breed exists and fetch its ID, else create a new entry in the Breed table and retrieve the associated ID. Next, you must check the shelter that is associated with the dog and fetch its ID, or insert a completely new shelter if its not found in the DB. We thought about limiting these options by creating a drop-down bar, but we felt that would be too restraining for the user. Finally you have to insert all the dog’s info into the PetInfo table, get that ID and insert the dog’s name, infoID, and shelterID into the overall Pet table. Normalization forced the creation of new entries to start from top down (breed to pet) rather than bottom up (pet to breed).

Unfortunately, we were not able to implement everything that could have made the website complete. One useful addition would be a search bar at the top of each table that would have been great to give users a way to query for specific data. Another optional feature would be to add a timestamp on the generated csv file for version control. Most importantly, we would love to have a picture of all the pets in our database because knowing what an animal looks like is possibly the most important factor when people decide which pet to adopt. We have made plans to implement all of these features onto the website but simply ran out of time. Finally, all the data currently in our database is fake data generated in Python by the library Faker. In order to give our project real world sustenance and relevance, we would substitute the fake data we used to populate our database with real data of pets in the OC area pulled from online.