Countries & Capitals App

Estimated Time: 240 minutes

To close out this lesson, we're assigning you the most difficult project so far. In this challenge, you have to code up an Angular app that allows users to search and visualize data about countries and capital cities. This project will put your new knowledge of routes and services to the test, while forcing you to dive deeper into animation, promises and XHR requests.

API Endpoints and Access

To complete this project, you'll need to sign up for a free account with geonames.org. You can do that here. You can find the official docs here and a list of all of the endpoints available here.

You'll use use the following geonames api endpoints in your app:

countryInfoJSON

neighbours

search: You'll use this endpoint to pull in info about country capitals.

In addition to these API endpoints, we'll also get flag images for individual countries using the following url schema for img sources:

`<img src="http://www.geonames.org/flags/x/??.gif" />` , where `??` stands for two letter, lowercase country code.

Similarly, we'll get map images for countries like this

`<img id='myImage' src="http://www.geonames.org/img/country/250/??.png" />` , but here (don't ask us why!) the two letter country code must be in upper case.

Requirements

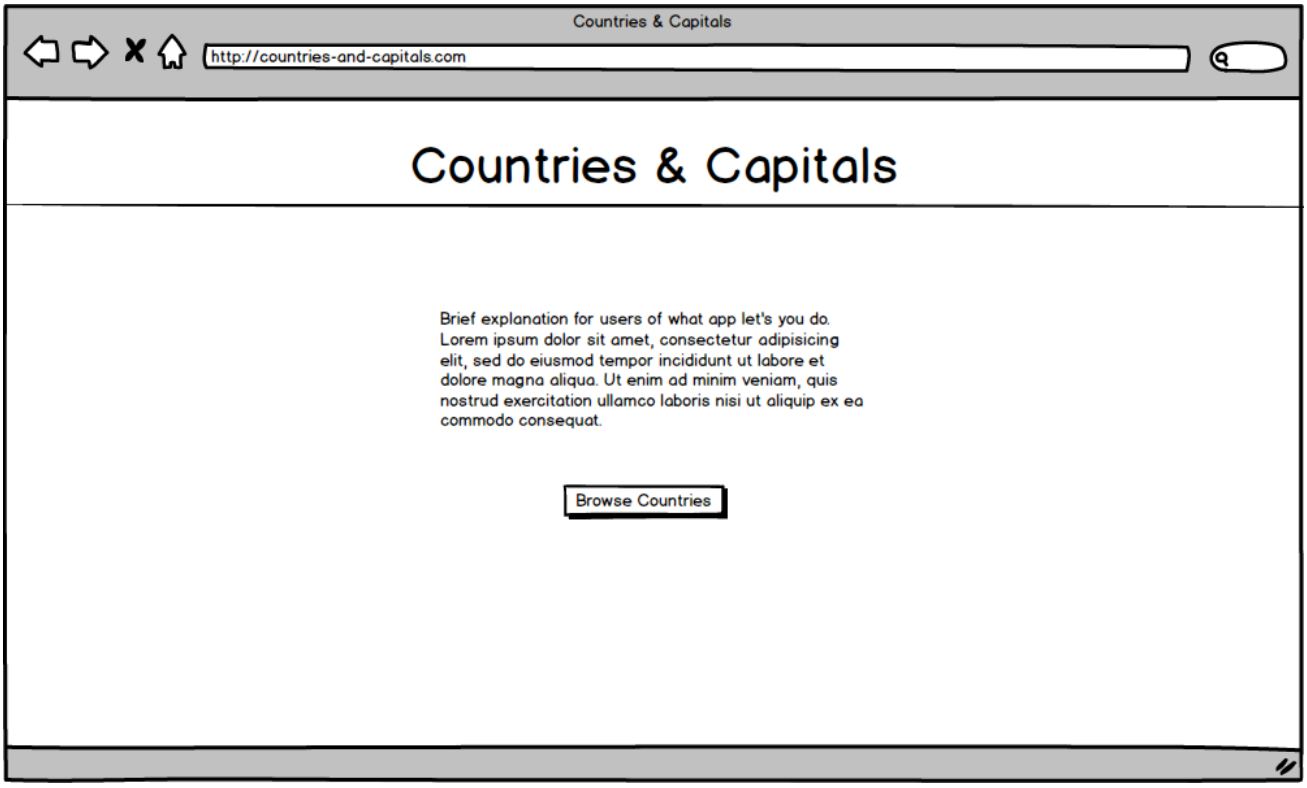
For this app, we provide minimal sketches for the different views along with some specs for the behavior for each view. It's not necessary that you produce a pixel perfect implementation of the sketches. Instead, the sketches indicate the components and functionality each view requires. You are free to

adjust the layout and appearance of any of the views as long as you implement the required functionality.

Views

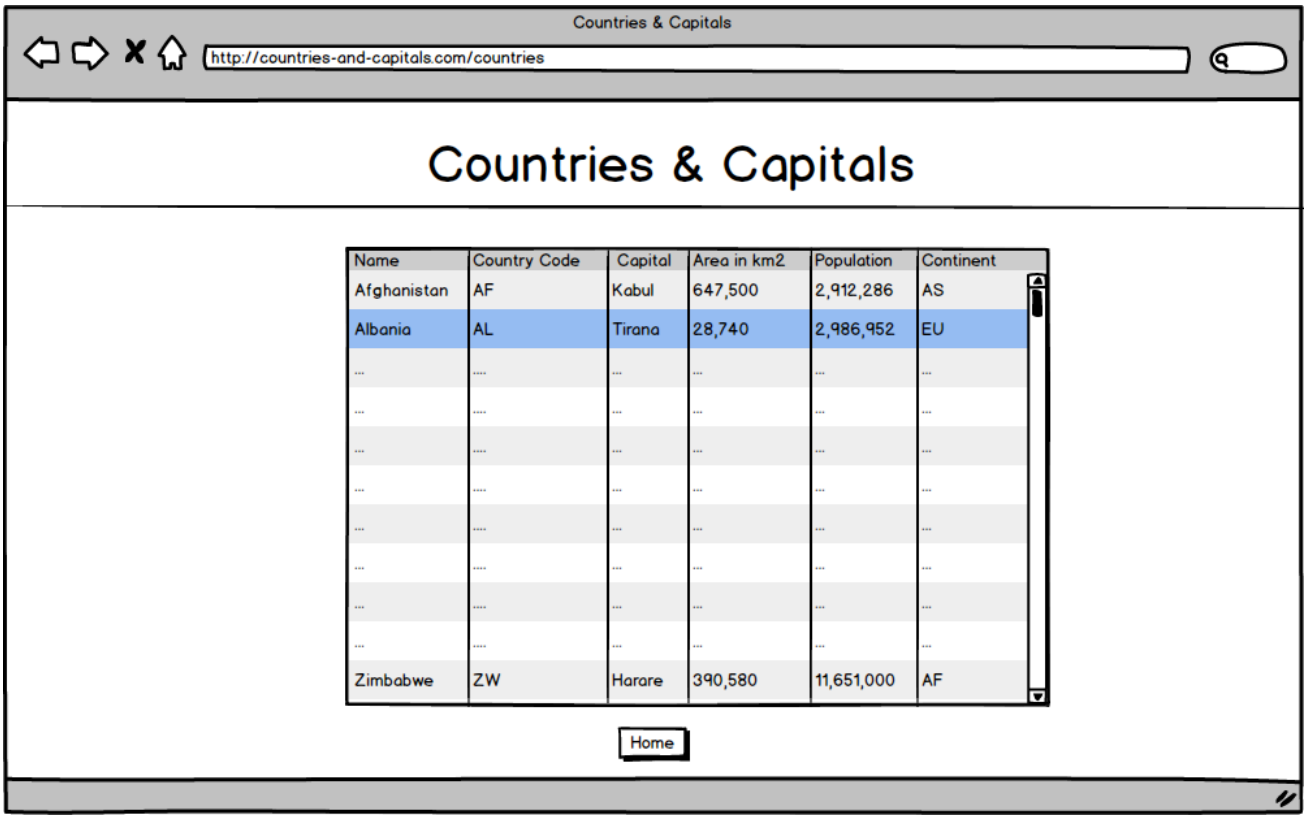
Your app will consist of three views, described below. Note that all three views share some page components, and you should avoid repeating this shared code.

Index View



This view is just responsible for displaying some explanatory text about our app. It's perfectly fine to use lorem filler, but if you're looking to create a more polished experience, you might include an image, graphic, or other visual embellishment.

Countries List



This view is responsible for letting the user see all of the countries they can view more detailed information about. Each row entry has the following summary information: name, country code, capital, area in square kilometers, population, and continent code.

The URL for this view should be `/countries`

To get the data for this view, you'll need to use this API endpoint.

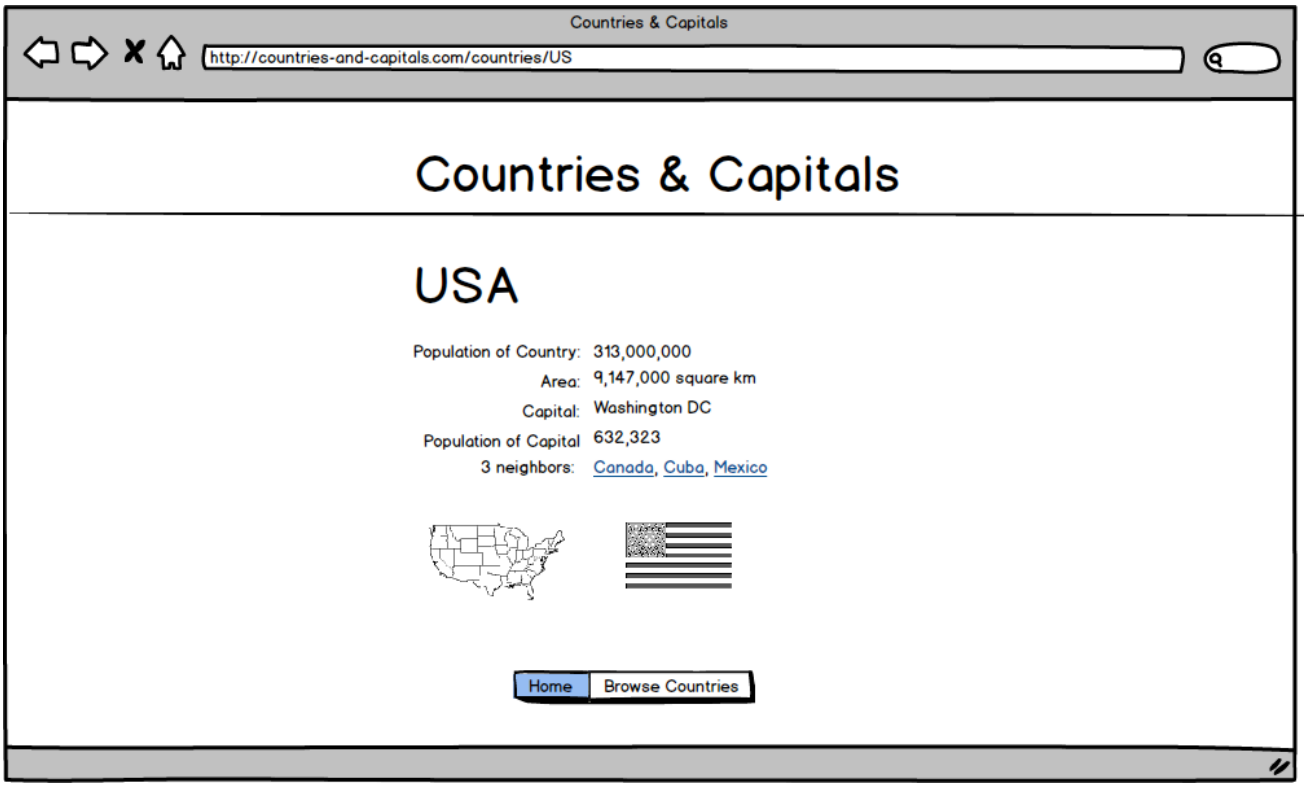
Also, your app should cache the data it gets back from the server. The countries list should only be gotten from the server once.

Note that both area and population should be formatted with commas after every third digit place (e.g., 1,000 not 1000).

There should be a button that takes the user back to the home screen.

When you click on a row entry, you should be taken to the country detail view for that country.

Country Detail



This view shows users detailed information about a country. The following data should be displayed: country name, population of country, area, capital, population of capital, continent, timezone, and number and names of neighbors.

This view should also display an image of the flag and map for the country, as explained above in "API Endpoints and Access".

The URL for this view should be `/countries/:country/capital`

There should be two buttons in this view, one to go back to the home view and one to go to the country list view.

The countries listed as neighbors should link to respective country detail view.

To get the population information for country capitals, you'll want to use the search endpoint. Have a look at the parameters you can supply in your requests to this endpoint. Hint: you'll probably want to use: q, country, name_equals, and isNameRequired. You'll also need to plan for the fact that this API endpoint may return more than one item.

To get the neighbors information, you'll need to use the neighbors API endpoint.

Animations and Loading

To get more practice with animations, we'd like you to create a loading state that gets displayed when AJAX calls are being made when the user visits a country or city detail view. You should use a timeout to ensure that the animation animations for waiting for ajax call and processing, with minimum return time, timeout

Additionally, you should use transitions to smooth the transition between all of the views in the app.

Project Layout and Build Process

For this project, follow the best practices for app layout we discussed in the previous assignment. Also, create a Gulp task that minifies and builds your files for live deployment, like we did earlier in this unit.

Project Completion

When you complete this project, be sure to save your work using Git and Github. Publish it using Github Pages, and share a link to your repo and the Github Page with your mentor and fellow students.