NPOLink

CS 373 – Software Engineering
IDB Project – Phase 1
10am Class- Group #6
npolink.me

By Georgina Garza, Gerardo Mares, Edgar Marroquin, Paul Purifoy, and Jacob Zillifro

---

Motivation

NPOLink is a web application which allows users to attain information about various non-profit organizations. Using this website, users are able to find out about non-profits in their area, non-profits by category, and even find ways to volunteer for and help these non-profits. We noticed that often websites either just gave direct information about a non-profit's name and tax-exempt code/non-profit category, or simply listed events to volunteer for. We wanted users to be able to find non-profits that they could actively be passionate about based on either location, category, or both, and find out various information about it.

User Stories

1. As a user, I should be able to get all of the organizations within one category so that I may find one that suits what I'm looking for.
    - Our RESTful API has been designed with a call that will directly allow for this to occur (GET Specific Category). This call will data will be originally scraped from the Nonprofit Explorer API, and then be held in our database. Our GET call will specifically access our database.
    - Estimated Implementation Time: 10hrs
2. As a user, I should be able to extract the location of an organization so I can know if it is relevant to me or not.
    - Our RESTful API has been designed with a call that will directly allow for this to occur (GET NPO by Name). This will provide various data about the organization, which will include the location.
    - Estimated Implementation Time: 10hrs
3. As a user, I should be able to extract the codes for any organization and find out what the code means so I can learn about the operations of the organization.
    - Our RESTful API has been designed with a call that will directly allow for this to occur (GET NPO by Name). This will provide various data about the organization, which will include the tax-exempt code.
    - Estimated Implementation Time: 10hrs
4. As a user, I should be able to get volunteer opportunities connected to an organization so that I can know what is available to do for an organization.
    - Our RESTful API has been designed with a call that will directly allow for this to occur (GET NPO by Name). This will provide various data about the organization, which will include the volunteer opportunities.
    - Estimated Implementation Time: 10hrs
5. As a user, I should be able to extract the description of every organization so that I can be informed about what each non-profit does.
    - Our RESTful API has been designed with a call that will directly allow for this to occur (GET NPO by Name). This will provide various data about the organization, which will include the description.
    - Estimated Implementation Time: 10hrs

RESTful API

The overall documentation for out RESTful API can be found at the following link:
https://documenter.getpostman.com/view/5491513/RWgm3gY7

We expect that all our calls will be formatted in the way the documentation has specified, and return a JSON object. Here is some information about the various GET calls that we plan to implement:

GET NPOs

- Retrieves a list of non-profit organizations within our database

GET NPO by Name

- Retrieves a non-profit organization based on a name; This will also include various information about a non-profit organization including name, description, location, volunteer opportunities, and more. The specific fields returned can be seen in the documentation under 'Return Format'.

GET NPO by Keyword

- Retrieves a non-profit organization based on a keyword found in either the description or mission statement

GET NPO Locations

- Retrieves non-profit organizations based on a state id

GET NPOs by State

- Retrieves a list of non-profit organizations based on a state id

GET NPOs by City

- Retrieves non-profit organizations based on a city id

GET Category

- Return a list of categories across our non-profit organizations

GET Specific Category

- Retrieves a list of non-profit organizations based on a category id

GET By Location and Category

- Returns results for every NPO within a specified location and category

<u>Models</u>

1. Non-profit Organizations
    - This model shows various non-profit organizations.
    - Attributes so far:
        i. Name
        ii. Description
        iii. Image
        iv. Location
        v. Category
        vi. Volunteer Opportunities
2. Locations
    - This model shows various locations that have non-profit organizations.
    - Attributes so far:
        i. Name
        ii. Description
        iii. Image
        iv. Organizations
        v. Categories

3. Categories
    - This model shows various categories for non-profit organizations.
    - Attributes so far:
        i. Name
        ii. Description
        iii. Image
        iv. Organizations
        v. Locations

__Tools__

1. Namecheap
    - We used Namecheap in order to obtain our URL (npolink.me), and as a way to help redirect our AWS Elastic Beanstalk application to the correct URL (see more information under Hosting).
2. Amazon Webservices – S3
    - Currently, we are taking a zip of our repository files and holding them in an S3 bucket so that our Elastic Beanstalk can have access to the code. We hope to further implement this in a more dynamic way so that when the repository updates, the website does as well.
3. Amazon Webservices – Elastic Beanstalk
    - We are using AWS's Elastic Beanstalk service to host our website. By grabbing the code from the S3 bucket, and then redirecting to our URL, this application hosts our web application.
4. Docker
    - We are using Docker as a simple way to run the application.
5. Docker-Compose
    - We are using Docker-Compose in order to create a way to work with multiple containers to run both the backend and the frontend at the same time. This will allow for greater efficiency in our program and will be further implemented as the website becomes dynamic.
6. Gitlab
    - We currently use Gitlab as a way to hold our code and work interactively. We create issues within our repository in order to keep track of what still needs to be completed and make various branches so that different aspects of the project may be worked on concurrently without merge conflicts.
7. Postman
    - To use Postman, one of our team members created a workspace and then downloaded the Postman app (available for download online) in order to begin.
    - We used Postman to scrape the data that is currently being represented statically on our website. To do this, we looked into the details of our API's and discovered the type of ‘GET' calls we would need to make in order for our team to get the information we needed.

- We also used Postman to design our API, and create the documentation for it. Refer to RESTful API portion for more details. This was created by considering what users may be interested in being able to access using our API, and designing various ‘GET' requests to meet these needs.

8. React-Bootstrap
    - React-Bootstrap was used across the entirety of our website in order to provide quality layouts and styling for our web application's frontend. By using react in general, we were able to break up our application into components, allowing for faster designing of each page. For example, a component could be made which represented the layout for each of the model pages.
9. ReactStrap
    - In addition to React-Bootstrap, in a few instances, we used ReactStrap for our frontend. This included the home page, navigation bar, and pages for the instances of the various models. ReactStrap furthered our possibilities for our layout and provided ways in which we could easily add interactive tiles.
10. Flask
    - Flask is a python framework we are using for the backend of our application. As our web application is not dynamic yet, we have not gotten in too deep to using Flask.
11. Grammarly
    - Grammarly was used in order to clean up and polish the writing within the technical report.
12. Slack
    - Slack was a simple tool the group used to interact and communicate throughout the project.
    - Slack was directly integrated with our GitLab repository in order for automatic messages to be sent about recent updates about code and issues.

Hosting

We are currently using both Namecheap and Amazon Web Services to host our website. We used Namecheap in order to obtain our URL, npolink.me, through their education account options. Once we had the URL, we worked on the website itself and put that code into a zip, and placed it into an AWS S3 bucket. Following this, we set up the Elastic Beanstalk application's environment, which was dependent on the code in the S3 bucket, and obtained the automatic URL this application was being hosted on. From here, we used AWS's Route 53 Management Console to create a way for the Elastic Beanstalk application to automatically redirect the given URL to redirect to our custom one. To do this, we created a Hosted Zone, with the name of our application, and within that created a CNAME Record Set with the value set to the automatic URL from the Elastic Beanstalk application. This set up the AWS side. We then had to go into Namecheap and create redirect URL's from our website to that of the automatic URL for the application. This can be done by creating CNAME records and URL redirect records through Namecheap. At the end of these steps, the web application was then properly hosted on our custom URL.