Inspiration

Do you need an API? Are you tired of AWS and its complicated UI? API design tools can be confusing for beginners, so we set out to create the most simple backend deployment tool on the market - !Backend(pronounced not backend)

What it does

!Backend takes a URL to a public GitHub repository, and gives you a link to run the functions as API requests.

How it works

The service is built using Angular front-end web framework and Python Flask back-end framework. !Backend runs in the following steps:

1. Ask for URL input for GitHub public repository
2. Send URL to a Python web server
3. Backend server converts the `main.py` from the GitHub repo to a Python Flask application
4. Web server returns API URL that can now call the functions of `main.py`

The result is an API that looks like this: `http://notbacken.tec/r/<unique_uuid>/<your_python_function>`

When a request is sent, if uses the UUID generated from step 4 to access the correct flask server, and runs the method with the parameters specified

Challenges we ran into

Overall we where not very experienced with web development and javascript, so some simple tasks like updating a field from the result of an API call to take a long time. Additionally, we initially wanted to deploy each API to a docker container, or run them all together, and had a difficult time deciding on a design that would enable us to do that (specifically most of these tools only allow one instance bound to a specificity).

Accomplishments that we're proud of

- Created an incredibly simple way to deploy Python APIs
- Created an advanced backend with multiple stages and parts

What's next for !Backend

- API generation depends on many factors - such as return types in the Python functions and structure of program
- Security has to be handled very carefully whenever arbitrary code is being run on a server. Right now, security is not handled that way.

Developer docs

/frontend

Run `npm install` for dependencies and `ng serve` to start the website.

/backend

Run `python3 service.py` to start the Flask web server.