KuraLabs Microservices Lab #2 (Project Moneda)

Andrew Dass - Leader
Craig Celestin - Speaker
Jepson Saint-Pierre - Documenter
Zach Cyrus

Our MultiMedia Service Application: Project Moneda

- Moneda can perform the following tasks:
    - Direct Messaging, Sharing, Analytics
    - Forum creation, suggestion of forums to join
    - Blogging, posts integrated with various forms of curated content
    - Podcast creation, podcast suggestions, non-penetrated podcast topics
    - Music suggestions, music-making platform service
- A new fast social media that is entirely about curators and curator support
- Our app performs many tasks, therefore a microservice architecture will be implemented to run these tasks more efficiently

The Websites’ Multimedia MicroService Architecture

- To implement a microservice architecture, we need assistance from a third party
- We have to make sure Moneda can compute, store, and network efficiently
- Many of Amazons’ services fulfill these tasks, and we considered to use the following:
    - Computing : EC2’s Virtual Servers
    - Storage and Databases:
        - Amazon S3 - Primary data storage
        - Amazon RDS - Recover lost data
        - EC2 - Machines, Scaling
        - Elasticache - Monitor tasks, analytics, performance
    - Networking : EC2
- Connect services by using Python, Javascript, Github, Jenkins and Amazon’s EC2
    - (Have services communicate to one another through HTTP following REST principles)
- Our app requires this much technology integration our app’s features

Features

- Moneda provides many services and rules to customize it to your preference:
- Integrated Messaging via an internal secure service, external messaging app integration:
    - Text freely to people in your or outside of your network
- Forums:
    - Join several communities
- Blogging:
    - Create your own blog
- Podcasts:
    - Start or join a podcast
- Music
    - Upload music tracks from other platforms and save playlists
- Each of our features can be customized for a person’s liking while they follow every rule

Additional Feature Rules and Guidelines

- Messaging
  - Real time communication with others users
  - Can enable feature to send text to prevent accidental mistakes
- Forums
  - Enter any community, gives two warnings before entering explaining what kind of community it is
- Blogs
  - Users will be able to create and manage their own blogs
  - Prewrite blogs and schedule release dates
  - Ability to easily share blogs to other platforms
- Podcasts
  - Listen in or join podcasts from forums or blogs
- Analytics
  - Collect and analyze data reports from our provided services
  - Send surveys to users for feedback to implement further improvement

Connecting to the Github Repo

- Github account: https://github.com/andrewdass49/micromonolab
- The following steps are needed to connect to a Github account:
    - **git init** - Initialize the GitHub repo
    - **git config --global.user.name** - Enter your name
    - **git config --global.user.name** - Enter your email
    - **git config --list** - See if you name, email and other information are entered correctly
    - **git remote add origin …** - Used to connect to a Github account. The … should be a Github html or SSH
    - **cat .git/config** -Configure the global accounts with the repo
    - **ssh-keygen -t rsa -b 4096 -C "email"** - Generate the ssh key for your specified email
    - **cd ~/.ssh** - Go back to the home directory then to your new ssh folder
    - **ls** - Check the ssh folder for files
    - **cat id_rsa.pub** - Should see a ssh key. Login into your Github to insert the ssh key
    - **exec ssh-agent bash** - To see if the ssh receives no errors to ensure it is working properly
    - **git push** - Connect to a Github account
    - **git add "file location name"** - First line to add a file to Github
    - **git commit -m "file location name"** - Now commit the same file that was once added
    - **git push -f origin main:secondbranch** - Choose the branch for the file to be added in