Rutgers Course API

David Parsons Spring 2018 Independent Study Proposal

Object

Create a server-less and developer friendly version of the Rutgers Schedule of Classes (SOC) API.

Abstract

There currently exists one API for accessing Rutgers Universty Schedule of Classes (SOC) Data. This API is not easy to find, is undocumented, and provides only one endpoint.

The currenty SOC API does not provide allow for any querying and the data returned contains lots of empty fields, in general is not the easiest to understand, and is some cases uneccesarily duplicates information.

Link to current SOC API Endpoint which provides data for Spring 2018 Undergraduate CS Courses: http://sis.rutgers.edu/oldsoc/courses.json?subject=198&semester=12018&campus=NB&level=UG

This makes it very difficult for Developers to find, understand, and use this API, and provides a tremendous barrier to anyone who wants to use this information to create a program or tool for themselves or for the University.

The aim of my project (currently named the Rutgers Course API), is to overcome all of these issues by providing a publicily available, well documented, and rich versions of the SOC API.

Design

To achieve this goal, I plan to leverage Amazon Web Services (AWS), specifically the Lambda, API Gateway, and RDS modules, in order to create a easy to maintain server-less API.

Server-less refers to a modern API framework that leverages "FaaS" (Function as a Service) Services to provide easily develop scalable APIs with less Infrastrucuture overhead.

My use of this framework and the AWS technology can be demonstrated here which follows the follow of a API Call which can be followed top down.

[User] API Call (Developer via URL)

- AWS [API Gateway] (Handles Request)
- [API Gateway] Calls AWS Lambda Function passing query params parsed from request
  - AWS [Lambda] (Performs Query)
  - [Lambda] Runs either JavaScript or Python Code to collect data from backend.
  - [Lambda] Interacts with AWS RDS
    - AWD RDS (Performs Query in native Query Language)
    - [RDS] Returns Query Data to AWS Lambda
  - [Lambda] Formats and Returns Data from RDS to API Gateway
- [API Gateway] Packages and Returns Data from Lambda Call to API Call (API Response)

[User] API Data Recived

This project and design yeild itself to the following areas to study throughout its design and development.

Study Areas

- API Design & Development
  - Endpoint Design
    - What Endpoint should be provided to the developer?
  - Query Design
    - What queries should the developer be allowed to perform
    - What Data Should be exposed to the developer?
- Scripting & Automation (Data Collection)
  - Data Collection and Scrapping
    - What are reliable data sources?
    - How to Reliably Collect and Update API Data?
    - Creating a Framework to Automate AWS Development
    - What Scripts are needed to automatically maintain endpoints and services?
- Database Design (Data Storage)
  - What data needs to be stored?
  - What's the best way to store the Data?
- Amazon Web Services (Cloud Infrastructure)
  - Depolying Database and Qeury Logic
    - Lambda
    - RDS
  - Exposing the Logic to Developers Safely
    - API Gateway

- Server-less Web Services
  - See AWS
    - Lambda
    - API Gateway
    - RDS

Conclusion

This project provides me with an oppourtunity to achieve two goals. First to complete a projec that I have spent almost my entire University tenure tying to complete, that exposes me to technologies and area's of Software Design that I was not able to touch upon in my course work. And second, to provide a easy to use, well documented, endpoint rich, Course API for other Rutgers Developers and Students to use in their projects, greatly reducing work required for any projec that requires the use of this data.