REST Application Services

Final Report

Ajay Pandit
pandit.aj@husky.neu.edu

Harsh Raval
raval.ha@husky.neu.edu

Kushmi Dedhia
dedhia.k@husky.neu.edu

Problem Statement

The world right now has undergone the transition from accessing data from computers/laptops to their mobile devices which has caused a lot of institutions to change their focus from Web Applications to Mobile Apps. Also with the availability of large number of APIs we can request data directly from those services rather than creating a new database replicating the information that is already available on the web.

Developers write Applications from scratch, which consume web services and provides data available from scratch. The process of fetching information from web services is more or less the same, only the path parameters, query parameters and values change based on domain and scope. This process requires lots of coding plus it does not allow the usage of the same application for multiple users. Hence the main focus for the developer should be to provide better solution rather than thinking of how to achieve it.

Also, here we are giving importance that the developer of the Application needs to have prior knowledge of the technology, but if another developer understanding the architecture would like to create a similar application needs to spend time on learning these technologies to be on par and then spending time in writing down the entire application from scratch which is time consuming.

Even though there are websites like IBuildApp which allows creating Applications easily but they are on the Web, but ideas can come anytime and what can be done if you just had your mobile with you.

Is there an App which can create another App on the go, allows Developers to share APIs?

This is the problem that we are trying to solve.

Proposed Solution

For the above problems we propose the solution of creating an Application which is capable of creating other applications making use of the Restful API Services.

Since almost all of the Restful API applications work on the same fundamental architecture wherein the API provided is associated with path parameters and request values for query parameters on which search is performed allows us to make an application which is capable of creating applications.

This allows Developers to create Applications by click of a few buttons without thinking of the technology involved behind it; thereby erasing the constraint of having to know the technology.

This enables Developers to focus on ideas that could solve the problem rather than worrying of how it could be implemented.

This Application as a whole allows Developers to reduce redundancy of implementing the same APIs as it acts as store of APIs which has been used by previous Developers but also acts as an aggregating agent for new APIs allowing constant growth and easy access.

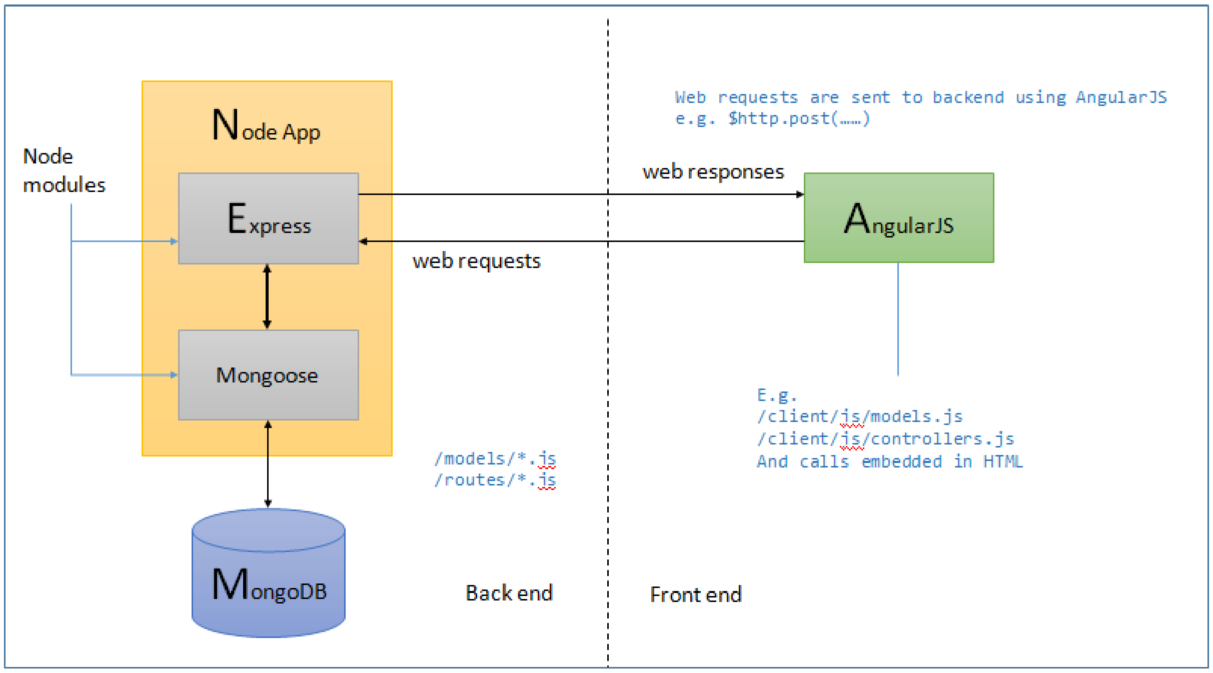
It also allows easy access for configuring on how the developer wishes to show the results and editing them after saving. It changes the focus of constructing the output on the basis of the result; thereby requiring that the Developer be API knowledgeable than technology knowledgeable.

Since it would be a mobile application, it is helpful on the go, as ideas can pop up any time.

The application can currently be accessed from http://v2-restappv1.rhcloud.com

For easy access, we have created the user 'dbms' with password 'dbms'.

Architecture



NodeJS provides the fundamental platform for development. The backend services and server-side scripts are all written in Node.js. MongoDB provides the data store for our project but is accessed via a MongoDB driver Node.js module Mongoose.

Express acts a webserver for Node.js extending to provide several key components for handling web requests. It allows us to implement a running webserver in Node.js with only a few lines of code.

AngularJS framework handles the view in the browser. It allows the MVC framework wherein the model - made up of JSON or JavaScript objects, the view - HTML/CSS and the controller - AngularJS JavaScript.

MongoDB on the other hand helps for a scalable NoSQL database where the data is stored in the form of JSON Objects rather than the traditional column, row allowing the user to access data in an Object Oriented Programming way.

APIs

There are currently using four APIs in our application:

Category	**API Description**	**Link**
Movie	Rotten Tomato	api.rottentomatoes.com
Weather	World Weather Online	api.worldweatheronline.com
Food	YUMMLY API (Food)	api.yummly.com
Articles	NYTimes Article Search API	api.nytimes.com

An example of the usage of the Weather API:

- Web Service URL: api.worldweatheronline.com
- Path Parameters: /free/v2/weather.ashx
- Query Parameters:

  q=London&format=json&num_of_days=5&key=%2073cffeae313423bf618a6dce02b1d

An example of the usage of the Movie API:

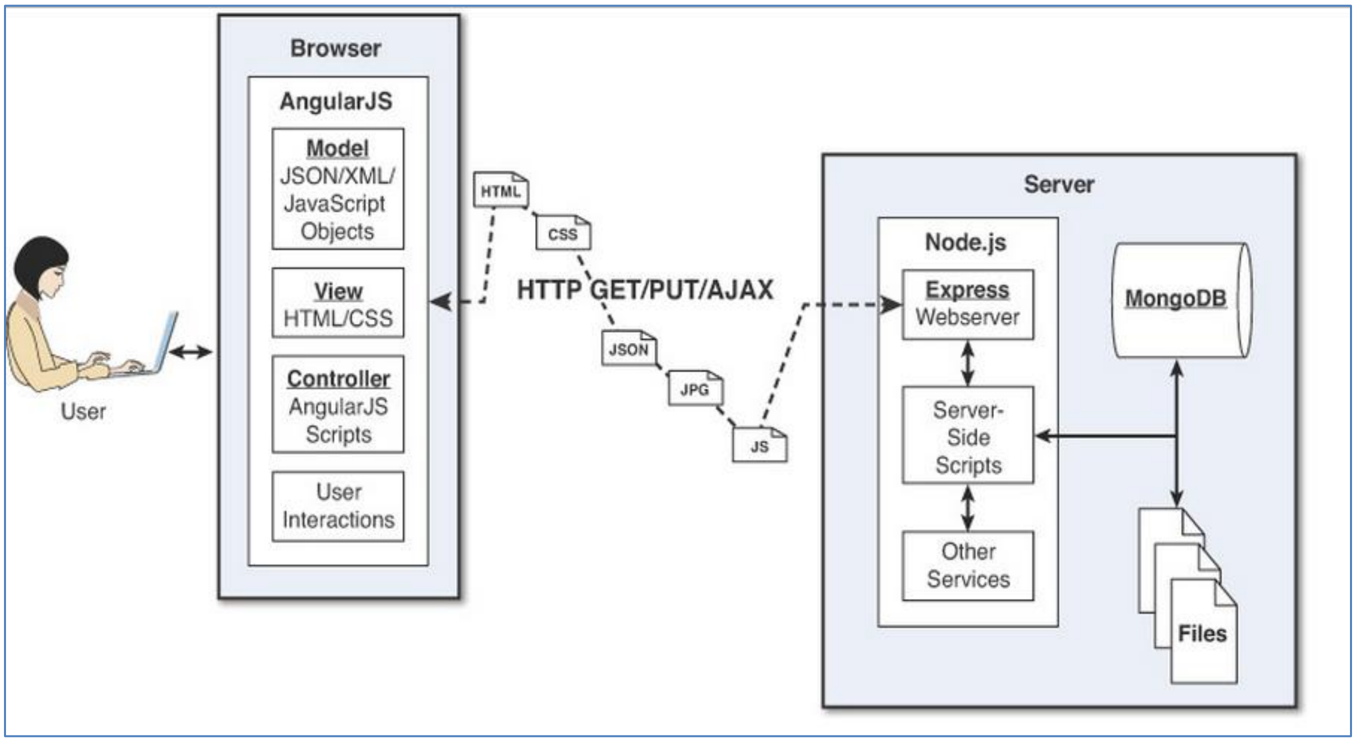
- Web Service URL: api.rottentomatoes.com
- Path Parameters: / api/public/v1.0/movies.json
- Query Parameters:

  apikey=9cceau4cw9hrskbfpee23x7j&q=Harry+Potter&page_limit=1

Technology Stack

We are using the Technology Stack of MEAN JS wherein:

- M – Mongo DB (database)
- E – Express (server)
- A – Angular JS (front end)
- N – Node JS (back end)

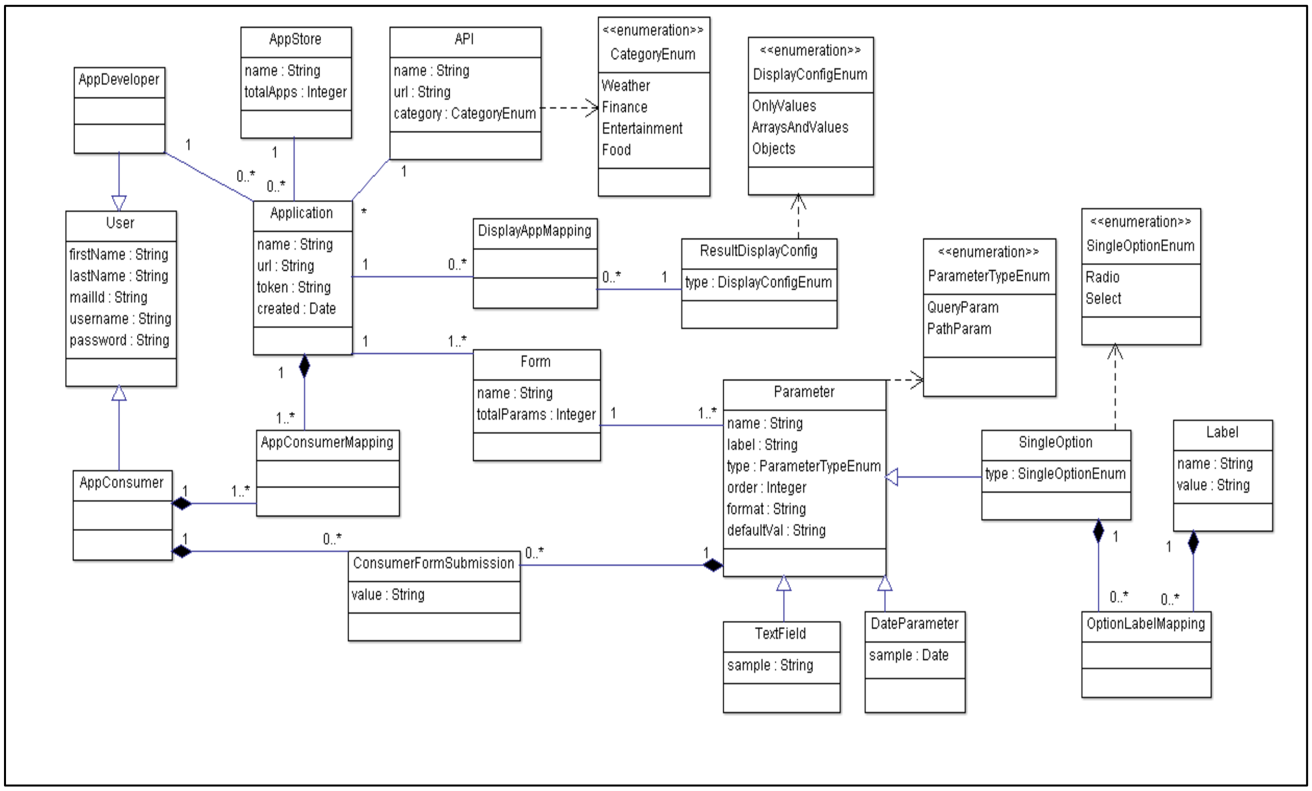


Use Cases

Use Case	Description	Status
Create Account	User wants to create an account for either Developing or Consuming applications	**Completed**
Login into Application	User wants to login into the application to access the account details	**Completed**
Create an application	Developer creates a new application by providing the name, and the API service to be used	**Completed**
Delete an application	Developer deletes the application which in turn deletes all the created parameters and notifies all the Consumers that the application has been deleted by the Developer	**Completed**
Modify an application	Developer modifies the application by changing the information like name, category and the API service used	**Completed**
View an application	Developer wants to view an application which they have already created	**Completed**
Add Parameter to Application	Developer adds a Query or Path Parameter which is appended in the API	**Completed**
Modify Application Parameters	Developer updates/modifies the parameter associated with the API service	**Completed**
Delete Application Parameters	Developer deletes any of the Application Parameters	**Completed**
Create New Service	Developer adds a new service not present in the provided list of services	**Completed**
Configure Application Result	Developer configures the output displayed to the user according to the attributes obtained from the JSON	**Completed**
Publish Application	Developer to publish the Application on the Application Store	**Completed**
UnPublish Application	Developer unpublished his Application on the Application Store	**Completed**
Test Application	Developer can view 'user-view' for his application to test	**Completed**
Execute Application	User executes the application by entering the query parameters	**Completed**
View Result	User views the result of the query submitted	**Completed**
View Application	Consumer can view all his downloaded applications	**Completed**
Search Application	Consumer can search for a desired application	**Completed**
Add Application	Consumer adds/downloads applications that he wants to use	**Completed**
Delete Application	Consumer deletes applications which is no longer used	**Completed**

Data Model

Initial UML Diagram Proposed:



For the previous UML diagram, our idea relating to the Service was related to how the entire system worked.

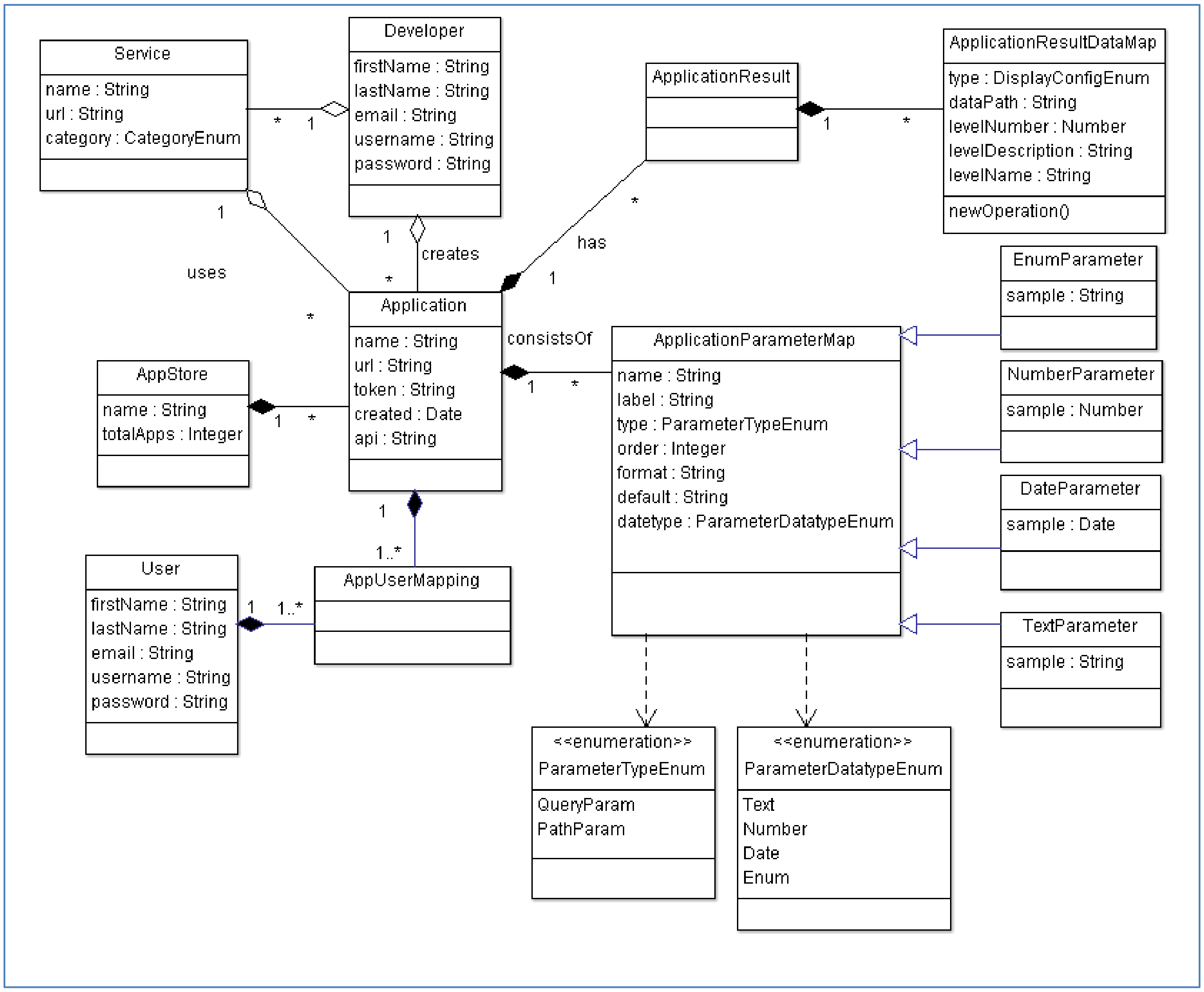
The AppStore will many Applications created by the Developer and consumed by the Consumer. Since, the Developer can also be a consumer of some other Application, it would logically correct to create a single class holding the information and then we can map, who is a Developer and who is a Consumer, for both entries will be made in both the classes.

The Application would be associated with API and using a form the Developer will decide the type of parameters that would be required. The Parameters could be of different natures like TextField, DataParameter, Options, etc.

Also the Developer has the capability of selecting the view configuration of the Consumer, thereby creates a map of the view.

The Consumer on the other hand, after logging in with input values to the fields required for the search and thereby view the output in the format that was configured.

Final UML Diagram:



There are some changes in the database schema which was proposed and what was finally implemented:

- The ENUM parameter type does not provide option to add values rather we can directly enter values as text separated by a special character "/"
- We are not storing the runs made by the user for the applications, which means we would not have any history of the queries that user submitted to the application
- We did not add category parameter for the API Services, since we were testing on single Service under each category, we can say this can be an enhancement
- User and Developer are separate entities

Future Scope

Currently there are four future enhancements that we have thought which can be implemented, they are:

1. **Result Configuration Enhancements**
   We wish to enhance the result configuration capability of our application wherein the Developer can save multiple types of configurations for showcasing results to the User

2. **Client Side Validation**
   We have currently not performed any client-side validation as it is out of bounds for the course, but having validation will improve the way user inputs are handled.

3. **Advanced Search**
   The current implementation uses plain text search, we can enhance the User experience by providing additional search options like by category, by date etc.

4. **Ratings, Reviews and Error Reporting for Applications**
   This would allow the User to submit reviews/ratings for a particular application and even report bugs in case they find one

References

- http://bootswatch.com/spacelab/
- http://daniellam.me/blog/getting-started-on-node-js-and-mean-stack/
- http://www.stackoverflow.com
- http://mongoosejs.com/docs/api.html
- http://expressjs.com
- https://www.codeschool.com/courses/shaping-up-with-angular-js
- https://egghead.io