CS3300: Introduction to Software Engineering

Design Document for RESTful API

Version 1.0: 06/07/15
Team: RESTful API

Contents

1 Introduction

1.1 Overview (Executive Summary)

This document describes the software being used in order to build an API service for a messaging application. In addition, it explains how clients can use the API to easily communicate between devices and manipulate resources. The API will adhere to the architectural principles and constraints of REST. These properties and how they will be implemented to make a RESTful API are explained in section three. The way in which software components and objects interact are represented by diagrams in section four.

1.2 Definitions and Acronyms

HTTP: Hypertext transfer Protocol
RESTful: (Representational State Transfer) refers to software that adheres to certain architectural constraints and principles.
Resource: the intended conceptual target of a hypertext reference
Resource Identifier: URL, URN
Representation: A representation is a sequence of bytes, plus representation metadata to describe those bytes.
Object: an instance of a class.
Class: a representation of an object that reflects its structure and behavior within the system.
GET: Requests a resource from the server.
POST: Similar to get but places additional info is sent in the body of the request.
PUT: Upload a resource to the server
DELETE: Delete a resource from the server
Spring: Java Developmental Framework
Maven: Software project management tool
URI: Uniform Resource Identifier
JSON: Javascript Object Notation.
API: Application programming interface.
Jackson https://github.com/FasterXML/jackson : Java JSON parsing library

IntelliJ: Interactive development environment

1.3 References

N/A

2 Design Considerations

2.1 Assumptions

The system was designed with the assumption that clients will be able to use HTTP requests with the JSON mime-type. This format will be used in the manipulation of resources. In order for clients to interact with our service, they will be provided with the information to do so. Client's will have access to class and sequence diagrams. This will define the classes, and inform the clients of class attributes. The class attributes will define the resource representation to be used by its JSON format. If a team cannot make requests with the JSON format, a different data format will need to be agreed upon.

2.2 Constraints

The constraints of the API being designed are:

- The system must provide service to a web client, and to an android phone, tablet and smart-watch client
- The system must allow for client communication
- The system must allow clients to upload/access resources
- The system must allow clients to manipulate resources
- The system must allow clients to transfer resources
- The system must able to handle requests from multiple clients at the same time
- The system must always be running

2.3 System Environment

MySQL Server 5
MySQL
Version 5.6.22
The system must use SQL Server as its database component. Communication with the DB is through JDBC connections. The system must provide SQL data table definitions to be supplied to the other teams (tablet,watch,web...etc.) for setup.

Amazon Web Service
AWS
The API being developed will use AWS to host it's web service and SQL database. Amazon will securely store all of our data, as well as deploy our application.

Spring
Version 1.1.2
Spring is a collection of libraries used in the source code of the project, so it has intimate knowledge of our application's data and source code.

Maven
Version 3.3.3
Maven will use an XML file to build the project. The XML file will have information about the project's dependencies, build order, package architecture, and required plug-ins.

3 Architectural (High-Level) Design

3.1 Overview

A web service is RESTful when its structure and functionality adhere to certain guidelines. REST focuses on components, the constraints of component interactions, and the interpretation of data. The basis of REST provides a uniform interface for clients to make stateless requests of resources. This stateless communication between the client and server allows the architecture to be made of layers. Each request contains all the information necessary to service it. Furthermore, cache constraints require that the data within a response to a request define its cahcheablity. If a response is cacheable, then a client can store that response data for later requests. The session state is held in the client. When the client is ready to transition to another state, a request is sent to the server.

The uniform interface of REST encompasses four constraints:

Identification of resources
A resource is any information that might be the target of a hypertext reference. This means a document, image, collection of other resources, application state, functionality and so on, can be a resource. Every resource is uniquely addressable using a URI.

Manipulation of resources through representations
When a client holds a representation of a resource, it can make requests to manipulate it. Standard HTTP methods such as GET, PUT, POST, and DELETE are used.

Self-descriptive messages
Clients send HTTP requests to a Resource Request Handler. Each resource includes the information required to process it. The method will be mapped to

corresponding operations of the resources in the Resource Request Handler. All resources are identified and a unique URI is assigned to each of them.

Hypermedia as the engine of application state
Clients make state transitions through the actions identified by the hypermedia. The representation of each state contains links that may be used in the client's next state transition. By linking with URIs, links can point to resources that are provided by a different application, a different server, because the naming scheme is a global standard.

Multiple software components are required to build the RESTful API. The main framework will be provided by Spring. The use of Spring's libraries will be used in the project's source code. Spring will be used with Maven in order to build the project and deploy it to the server. The server will be hosted by Amazon web service and by using the RESTful API, clients can manipulate resources through our database. MySQL will be used to provide the database component of the system. By using HTTP protocol, clients will be able to manipulate resources hosted in the MySQL database. The Resource Request Handler is implemented using Jackson and will be represented by JSON.

3.2 Rationale

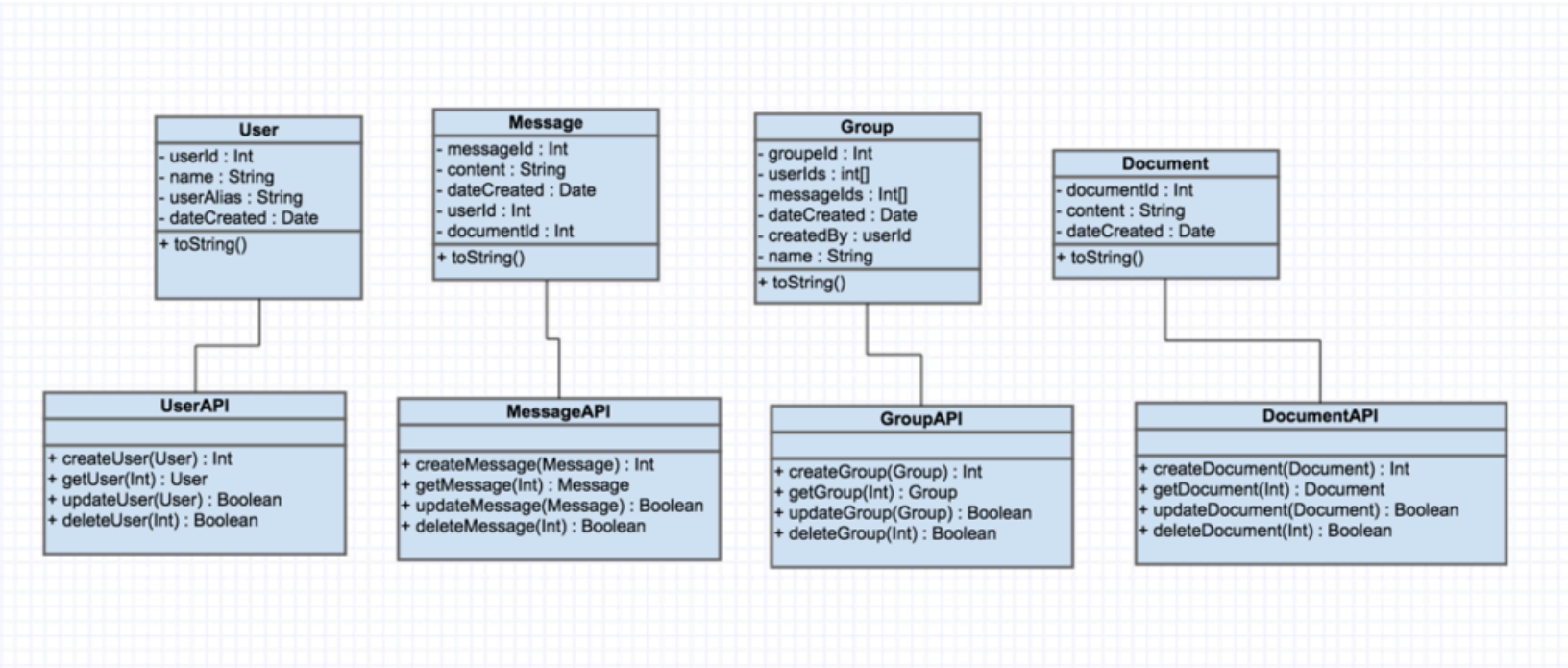
In order to build a service that can handle our client's needs, the architectural design of the system must be considered. A RESTful API allows clients to construct a GET URL with their Web browser. The content of the GET request can then be easily read and returned. This provides a simple method for clients to manipulate resources and transition to different states. Since it is assumed that the client developers are familiar with JSON, our service will use this as its format. JSON will provide a simple way to read, write and parse the resources. The stateless constraint between clients and server means the server will not process two consecutive requests from the same server. This will improve scalability by eliminating the need for the server to maintain multiple client states. The cache constraints make the system more efficient by only allowing explicitly

stated resources to be stored. Through this implementation, the system as whole is simplified.

3.3 Conceptual (or Logical) View

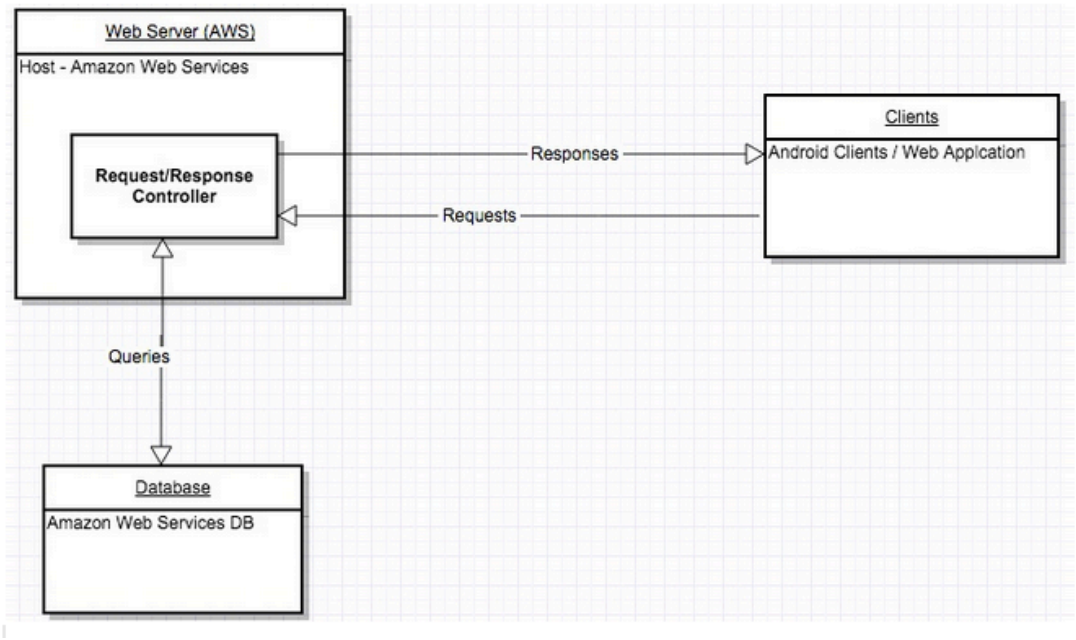
4 Low Level Design

4.1 Class Diagram



4.2 Sequence Diagram

4.3 Component Diagram



5 User Interface Design

User Interface is the API Documentation located in our README.md
URL : https://github.gatech.edu/cs3300/p1server/blob/master/README.md