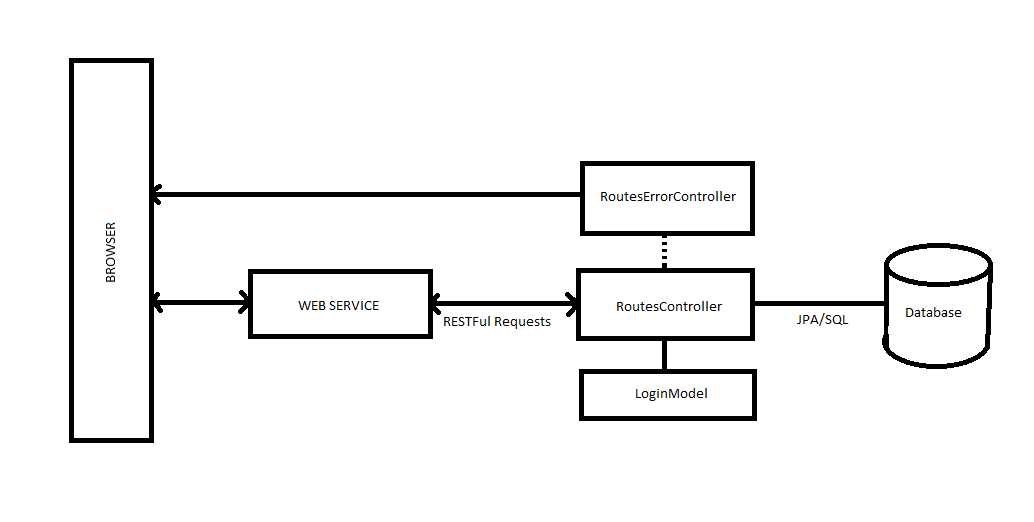
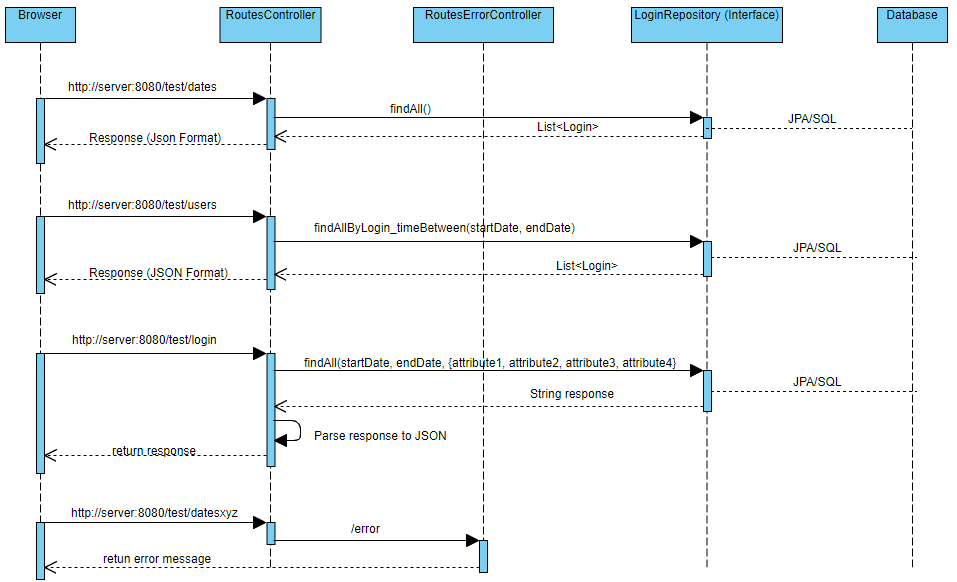
|  |  |
| --- | --- |
| **Implementation Details** | |
| **Technology** | **Java 8** |
| **Framework** | **Spring/Spring Boot** |
| **Project Management Tool** | **Maven 3.2** |
| **Database** | **PostgresSQL** |
| **IDE** | **Intellij, PGAdmin** |

Decisions Made and Assumptions:

* Used Spring Boot for simplified and reduced effort on configuration, easy database integration and since it provides RESTful services.
* Assumed database system would be RDBMS
* Used Hibernate as ORM for easy data manipulation and query handling
* Added data.sql on *resources* directory for creating dummy data on spring boot startup. This is the mechanism to provision database with 100000 data
* On error handling, just included status codes: 400, 404 and 500 as the common errors on resource access.
* Assumed on URI <http://server/test/users> that start date will default to beginning of epoch (Jan 1, 1970) if no value is passed and end date will have a default value of date today if no value has been passed.
* Assumed on URI [http://server/test/logins](http://server/test/logins?attribute1=AA1&attribute1=AA2&attribute1=AA3) attributes(1-4) that the next attribute will not be populated without the previous one but all are implemented as optional. Example: Attribute 1 should have a value when attribute 2 has a value.
* Assumed that date format should always be YYYY-MM-DD

**System Design**

**Sequence Diagram**



**Project Flow**

<http://server/test/dates>

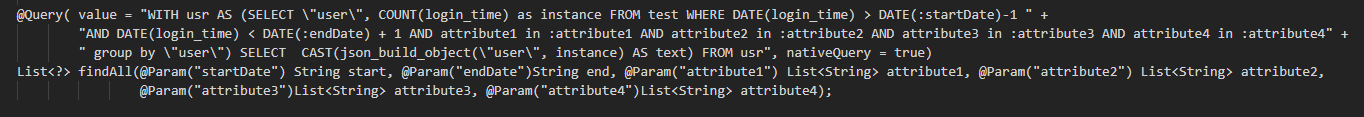
* Request will be mapped to /test/dates and will trigger query below.
* This query will select the unique login time in format of (YYYY-MM-DD), user, attributes1-4 and sort results by login time ascending.
* Result will be mapped to Login object and send it as a response in JSON format.

<http://server/test/users?start=YYYYMMDD&end=YYYYMMDD>

* Request will be mapped to /test/users and will check parameters attached to URI. one or both parameters are missing, a default value will be set and will execute the query below
* Query will select all unique user, format the login\_time from timestamp to date format and will pull rows that has login\_time in between the start date and end date ordered ascending
* Results will be mapped to Login object and will be sent as a response in JSON format

<http://server/test/logins?start=YYYYMMDD&end=YYYYMMDD&attribute1=AAA&attribute2=BBB&attribute3=CCC&attribute4=DDD>

* Request will be mapped to /test/logins
* Controller will check for start, end date and attributes. Attributes are stored in a List to cater multiple values.
* The query below will be executed



* Query will select all user and count the login instances of users having login\_time between start and end date and check attributes if attributes exists from the required attribute sets. Results will be casted to a text representing a JSON string.
* Resultset will then be processed again and will format strings to JSON object having user as key and value as the instances of login.