**A small twitter like message service in REST**

**User Story:**

* Build a samll Twitter like solution for users, where users can tweet , read and have followers.

**Implementation**

To build a RESTful service for the message exchanges by all the users

**Rest Service resource identification**

1. Users
2. Follower
3. Messages

**Rest Service identification**

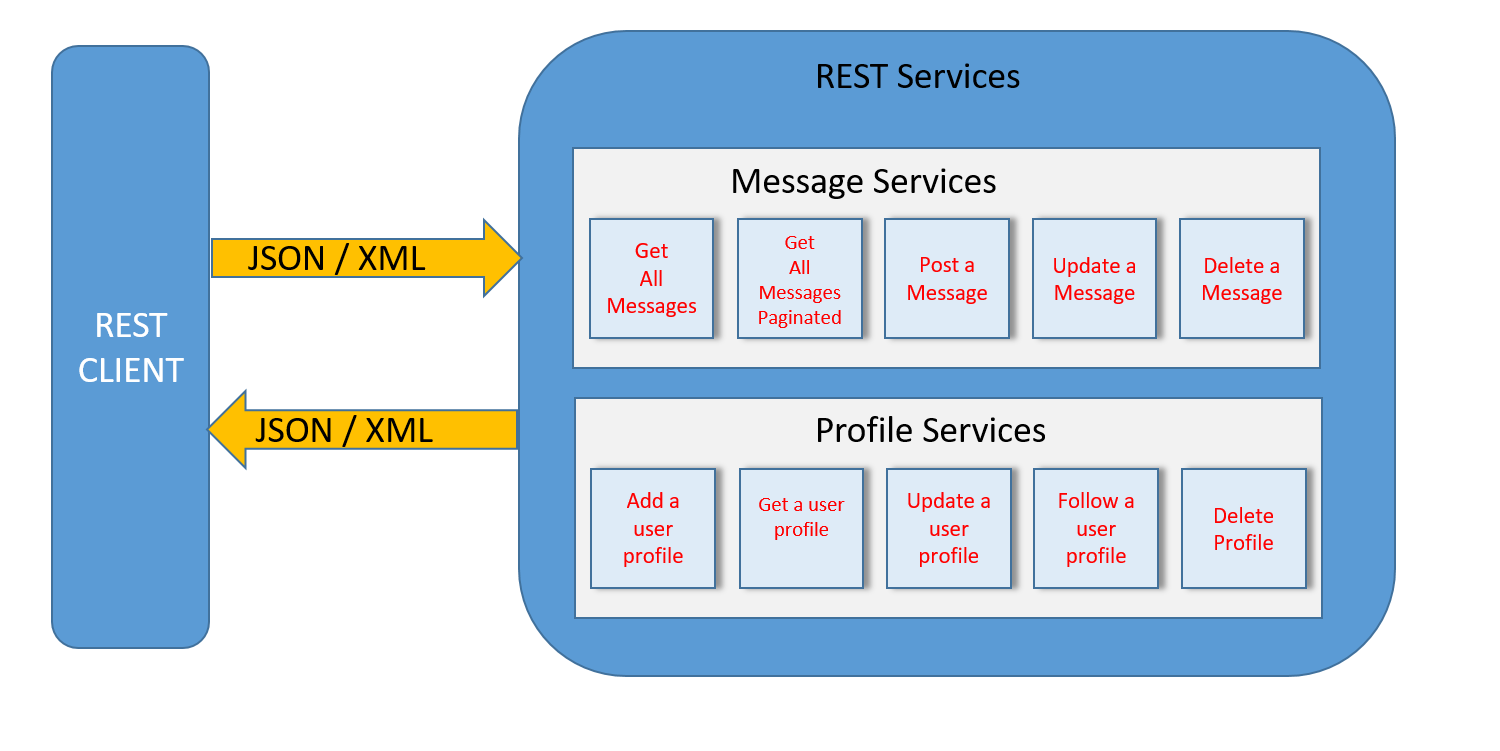
1. Registration
2. Login / logout
3. Search / suggest
4. Following a user / users
5. Accept the follower request
6. Sending message to a user / users

**Assumptions**

1. Users are already registered and are able to do login and logout
2. Users will be able to search and discover other users
3. Persistence used is in memory, but space provided to be scalable
4. Content type used is JSON , but can be made scalable to negotiate XML , Text etc.

**Architecture Diagram**

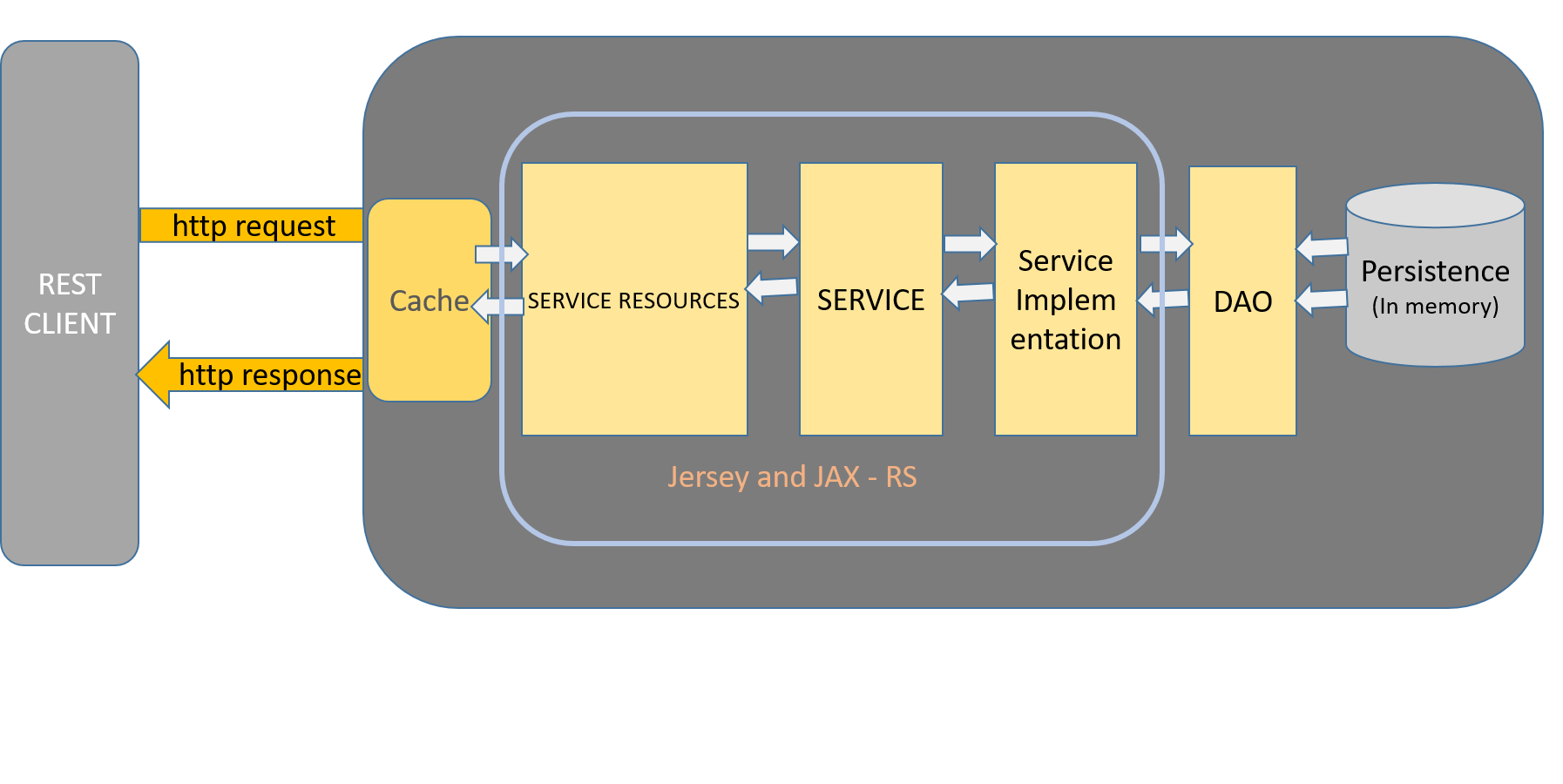
1. **Services**



1. **Technology stack**

Technology used are Java 8 , Jersey , JAX RS , Maven , Junit and Tomcat 8 and Postman used as client

No permanent data storage is used but the DAO layer can be extended further to ORM (Object Relational mapping) or database solutions. Current data stored are in memory.



**Class Diagram**

C:\Users\swain\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Untitled Diagram.png

**Data Modeling**

Current data stored and accessed are in memory using List and Map

**UserProfiles Map**

|  |  |
| --- | --- |
| User MailID (Key) | User Object (value) |
| Test1@test.com | Test1 |

**Messages Map**

|  |  |
| --- | --- |
| Message id (Key) | Message (value) |
| 1L | Message |

**UserMessages Map**

|  |  |
| --- | --- |
| User MailID (Key) | List of message Ids (value) |
| Test1@test.com | MessageId List {1L,2L,3L} |

**Factors considered or to be considered**

1. Designed as per OOAD ( object-oriented analysis and design) and HATEOS (Hypermedia as the Engine of Application State) concepts
2. The service is currently built upon the regular HTTP protocol . Additional security can be wrapped by implementing OAuth , JWT ,SSO or other technologies
3. A data caching option (ehcache) can be provided based on the user transaction statistics
4. An auto follower functionality can be implemented when use registers , by verifying location or mail ids or other hierarchy
5. The implementation has options to be scaled for data persistency to data based or other ORM solutions

**Implemented services and access**

User / profile

1. get all user profiles @GET

[http://localhost:8080/messageserv/feed/userprofiles/](http://localhost:8080/craftdemo/feed/userprofiles/)

1. Follow a user profile @PUT

<http://localhost:8080/messageserv/feed/userprofiles/follow/test1@test.com?followerMailId=test7@test.com>

1. Update a profile @PUT

<http://localhost:8080/messageserv/feed/userprofiles/follow/test1@test.com?followerMailId=test7@test.com>

Message

1. Get all messages for a user profile @GET

<http://localhost:8080/messageserv/feed/messages/test1@test.com>

1. Post a message to a following profile @POST

<http://localhost:8080/messageserv/feed/messages/test1@test.com>

1. Update a message @PUT

<http://localhost:8080/messageserv/feed/messages/test1@test.com>

1. Delete a message @DELETE

<http://localhost:8080/messageserv/feed/messages/test1@test.com?messageId=7>

**Build and Deploy**

Use maven command build and install to build the project and get the war file (messageserv.war).

The war file can be deployed into tomcat server and access the services.