REST API Server Assignment

Table of Contents

[Technology 2](#_Toc514719323)

[Database Tables and Columns 3](#_Toc514719324)

[Documentation of REST API 4](#_Toc514719325)

[User Stories 4](#_Toc514719326)

[Testing 8](#_Toc514719327)

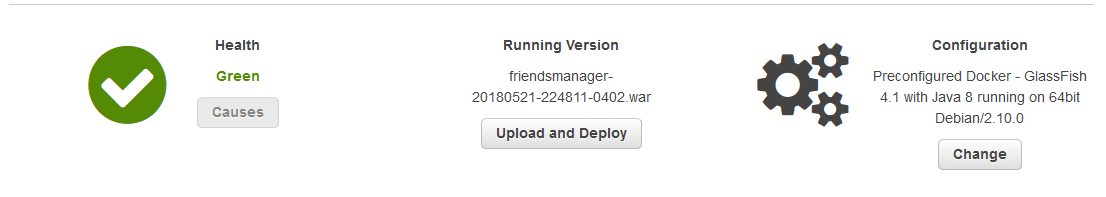
[Postman 8](#_Toc514719328)

## Technology

* **REST API** - The application is build using **Java JAX-RS API and Jersey** to write the REST calls.
  + It provides support in creating web services using REST architectural pattern, simplifying development and deployment of web services.
  + It comes with various annotations that eases the creation of services and makes it more readable.
* **JPA (Java Persistence API)** - Collection of classes and methods to persistently store the vast amounts of data into a database
  + Reduce the burden of writing codes for relational object management, follows the ‘JPA Provider’ framework, which allows easy interaction with database instance.
* **Database – MySQL**
  + Open source, stable and easy to debug and modify.
  + Being relational DB, provides support for one to one and one to many relationships which is very useful for social media applications to find connections.
  + Can also implement social graphs.
* **Server – Glassfish Payara**
  + Provides easy deployment using Netbeans
  + Easy to use admin console to form connection and thread pools.
* **Deployment Information – Amazon Elastic Beanstalk**
  + Configured an AWS Elastic Beanstalk environment with pre-configured glassfish-4.1 server
  + Connected the EB to AWS RDS (MySql instance)
  + Web application created using the above environment and RDA connection
  + **WebUrl -**

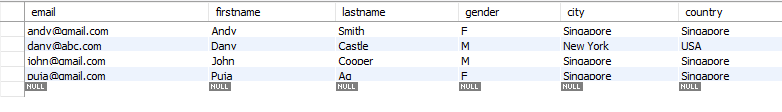
http://awseb-e-j-awsebloa-1n1rmejtxly17-1026695416.ap-southeast-1.elb.amazonaws.com/friendsmanager/<API>

***\*\*Note: API can be referred from API documentation section.***

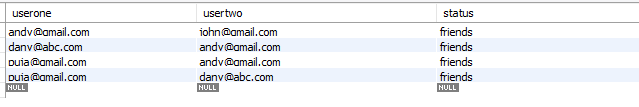


## Database Tables and Columns

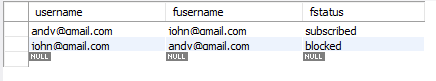
* **Users**



* + Email (Primary key)
  + Firstname
  + Lastname
  + Gender
  + City
  + Country
* **Relationships**



* + Userone (Foreign key – reference email(users))
  + Usertwo (Foreign key – reference email(users))
  + Status – describes the status of two users eg: friends
  + Primary key (userone + usertwo)
* **Followings**



* Username (Foreign key – reference email(users))
  + Fusername (Foreign key – reference email(users))
  + Fstatus – describes the status of users (enum value : subscribed or blocked)

***\*\*assumption: if a user has blocked another user, then he/she should not be able to listen to listen to its subscriptions***

* + Primary Key (username + fusername)

## Documentation of REST API

### User Stories

**1. As a user, I need an API to create a friend connection between two email addresses.**

**API End point: /api/socialapp/connect**

*API JSON Request:*

{

friends:

[

'andy@example.com',

'john@example.com'

]

}

*JSON Response:*

{

"success": true

}

*Response Error:*

* NOT\_FOUND – If user with email not found
* CONFLICT – If users are already connected as friends

**2. As a user, I need an API to retrieve the friends list for an email address.**

**API End point: /api/socialapp/friends**

*API JSON Request:*

{

email: 'andy@example.com'

}

*API JSON Response:*

{

"success": true,

"friends" :

[

'john@example.com'

],

"count" : 1

}

*Response Error:*

* NOT\_FOUND – If user with email not found
* NO\_CONTENT – If user has no friends

**3. As a user, I need an API to retrieve the common friends list between two email addresses.**

**API End point: /api/socialapp/common**

*API JSON Request:*

{

friends:

[

'andy@example.com',

'john@example.com'

]

}

*API JSON Response:*

{

"success": true,

"friends" :

[

'common@example.com'

],

"count" : 1

}

*Response Error:*

* NOT\_FOUND – If user with email not found
* NO\_CONTENT – If users have no common friends

**4. As a user, I need an API to subscribe to updates from an email address.**

**API End point: /api/socialapp/subscribe**

*API JSON Request:*

{

"requestor": "lisa@example.com",

"target": "john@example.com"

}

*API JSON Response:*

{

"success": true

}

*Response Error:*

* NOT\_FOUND – If user with email not found
* CONFLICT – Is requestor has already subscribed to the target

**5. As a user, I need an API to block updates from an email address.**

**API End point: /api/socialapp/block**

*API JSON Request:*

{

"requestor": "andy@example.com",

"target": "john@example.com"

}

*API JSON Response:*

{

"success": true

}

*Response Error:*

* NOT\_FOUND – If user with email not found
* CONFLICT – If requestor has already blocked the target

**6. As a user, I need an API to retrieve all email addresses that can receive updates from an email address.**

**API End point: /api/socialapp/retrieve**

*API JSON Request:*

{

"sender": "john@example.com",

"text": "Hello World! kate@example.com"

}

*API JSON Response:*

{

"success": true

"recipients":

[

"lisa@example.com",

"kate@example.com"

]

}

*Response Error:*

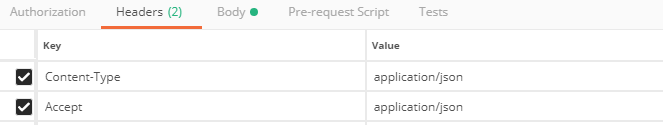
* NOT\_FOUND – If user with email not found
* NO\_CONTENT – If user has no valid list of subscribers

## Testing

### Postman

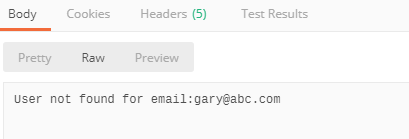
Postman (HTTP client) is used to test the REST services.

1. Setting up request headers:

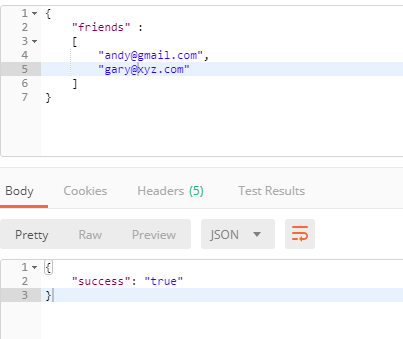


1. User story 1:

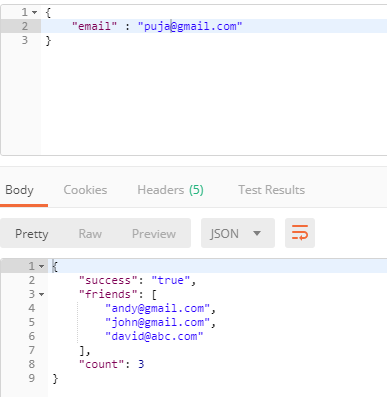
Error when user does not exists:



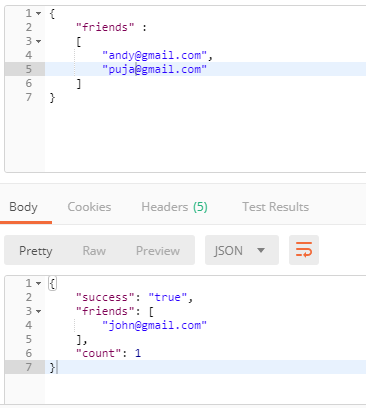
Success:



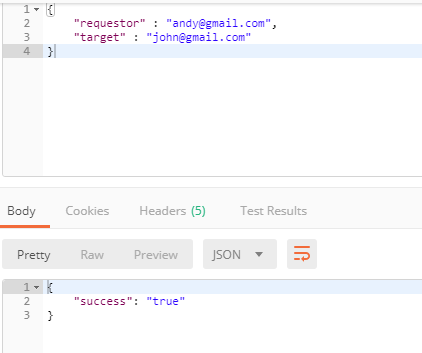
1. User story 2:



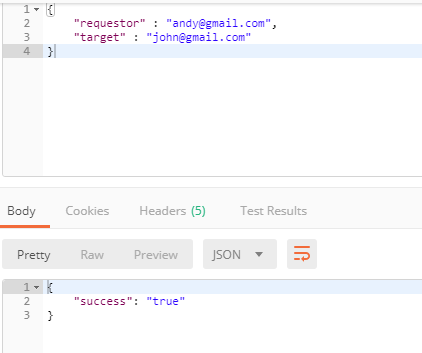
1. User story 3:



1. User story 4:



1. User story 5:



1. User story 6:

