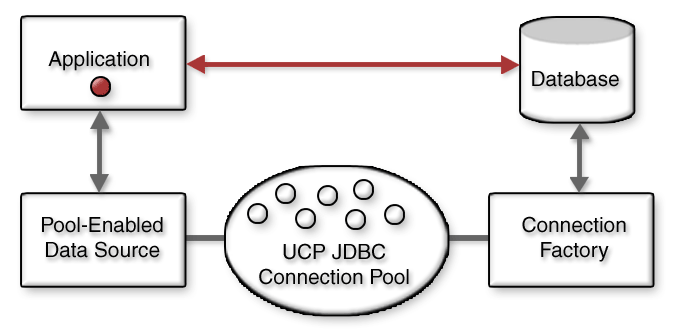
REST API Server Assignment

## Technology

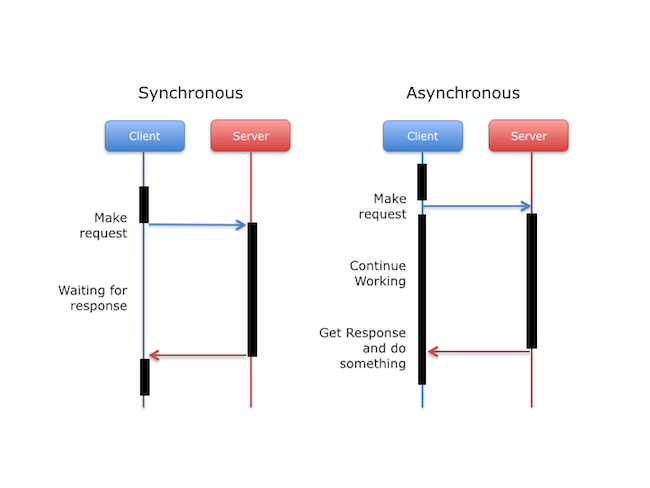
* **REST API** - The application is build using **Java JAX-RS API and Jersey** to write the REST calls.
  + It provides support in creating web services using REST architectural pattern, simplifying development and deployment of web services.
  + It comes with various annotations that eases the creation of services and makes it more readable.
* **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.

## Application Architecture

### Using connection pool



### Asynchronous Requests



## 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**
  + Following\_id (primary key)
  + 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

* + Date\_created – timestamp when the relation was created

## Documentation of REST API

* **/api** – Entry point for REST api calls to be used after application context
* **/api/socialapp** – Entry point for all REST calls
* **/api/socialapp/friends** – To fetch all friends of the provided email address
  + Returns list of all friends if user exists (Response code: 200)
  + Returns NOT FOUND is users with email address not exists
* **/api/socialapp/connect** – To connect given two users as friends
  + Returns OK if the users exists and not connected as friends
  + Returns CONFLICT if users already exists as friends
  + Returns NOT FOUND is users with email address not exists
* **/api/socialapp/common** – To fetch all common users between two given users
  + Returns OK with a list of common friends is the users exists
  + Returns NO CONTENT if no common users
  + Returns NOT FOUND is users with email address not exists
* **/api/socialapp/subscribe** - To enable requesting user to be able to subscribe to target user
  + Returns OK if users exists and able to subscribe
  + Returns NOT FOUND is users with email address not exists
  + Returns CONFLICT is already subscribed
* **/api/socialapp/block** – To enable requesting user to be able to block the target user
  + Returns OK if users exists and able to block
  + Returns NOT FOUND is users with email address not exists
  + Returns CONFLICT is already blocked
* **/api/socialapp/retrieve** - To retrieve all users who can listen to a topic published by the sender user
  + Returns OK with a list of all users who can listen to the sender (either friends/subscribed/not blocked or mentioned in @)
  + Returns NO CONTENT is no users available for listening
  + Returns NOT FOUND is sender user does not exists

## Steps to run the application

* Clone the project from the gitbub repository (<https://github.com/puja3001/Assignment_SPGroup>)
* Setup MySQL Server and run the script (friends\_management.sql) to set the initial database, tables and some dummy values.