**# mars server**

**## setup**

1. nodejs install

-> https://nodejs.org/ko/download

2. mysql install

-> https://dev.mysql.com/downloads/installer/

3. mysql workbench install(optional, but recommend)

-> https://dev.mysql.com/downloads/workbench/

4. mysql database create as "mars"

1) set mysql account/password in mysql install

-> root / q1w2e3r4

2) create database in shell

-> mysql -h localhost -u root -p (in cmd or mac shell)

-> q1w2e3r4

-> create database mars;

-> exit;

5. nodejs library installi n project

1) open mars folder in visual studio code

2) Terminal menu -> New terminal

3) "npm install" in terminal // library install

4) "npm start" in terminal // server start(local)

6. call rest api

-> use http://localhost/~ or http://{local ip address}/~

**## database**

\*It may be updated.

|  |  |
| --- | --- |
| Type | Data |
| User | |  |  |  | | --- | --- | --- | | Field | Type | Description | | id | String | PRIMARY KEY | | email | String |  | | firstname | String |  | | lastname | String |  | | address | String |  | | ipAddress | String |  | | isEnable | Boolean |  | | status | String | login  logout | | token | String |  | | lastLoginTime | String |  | |
| Conference | |  |  |  | | --- | --- | --- | | Field | Type | Description | | id | String | PRIMARY KEY | | topic | String |  | | start | String | YY-MM-DD-HH-MM-SS | | duration | String |  | | participant | String[] |  | | joinUsers | String[] | users in conference | |
| Contact | |  |  |  | | --- | --- | --- | | Field | Type | Description | | id | String | PRIMARY KEY | | users | String[] | user id array | |

**## rest api**

http status code

|  |  |
| --- | --- |
| Status code | Contents |
| 200 | success |
| 201 | successful resource creation |
| 202 | successful start of a request that will be handled asynchronously |
| 204 | success but the response body is empty (e.g., zero rows for a query) |
| 400 | data in the request is not valid |
| 401 | invalid user credentials provided for authentication |
| 403 | access denied (authorization failure) |
| 404 | the requested resource was not found |
| 415 | the request Content-Type header is an unsupported media type |
| 422 | business logic or business validation error |
| 500 | unexpected error processing the request (e.g., out of memory, NPE) |

Rest api

\*It may be updated.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Category | Method | Operation | path | Body(json) | return | etc |
| user | POST | create user | /user | |  |  | | --- | --- | | field | type | | id | String | | password | String | | email | String | | firstname | String | | lastname | String | | address | String | | ipAddress | String | | http status code |  |
| GET | get user | /user/:id | - | http status code  user data json |  |
| GET | search user | /user/:id/search?key={} | - | http status code  user data list | key => search keyword |
| PUT | update user | /user/:id/update | |  |  | | --- | --- | | field | type | | password | String | | email | String | | firstname | String | | lastname | String | | address | String | | http status code |  |
| PUT | login | /user/:id | |  |  | | --- | --- | | field | type | | password | String | | http status code  user token |  |
| PUT | logout | /user/:id/logout | |  |  | | --- | --- | | field | type | | token | String | | http status code |  |
| DELETE | reset user password | /user/:id/password | |  |  | | --- | --- | | field | type | | email | String | | http status code |  |
| DELETE | delete user | /user/:id/delete | - | http status code |  |
| Conference | POST | create conference | /conference | |  |  | | --- | --- | | field | type | | topic | String | | start | String | | duration | String | | participant | String[] | | conference id |  |
| POST | call request | /conference/:id/callRequest | |  |  | | --- | --- | | field | type | | userId | String | | participant | String[] | | http status code |  |
| POST | call reject | /conference/:id/callReject | |  |  | | --- | --- | | field | type | | userId | String | | http status code |  |
| GET | get conference | /conference/:id | - | http status code  conference data json | :id => conference id |
| GET | get conference list | /conference/:userId/list | - | http status code  conference data list json |  |
| PUT | update conference | /conference/:id/update | |  |  | | --- | --- | | field | type | | topic | String | | start | String | | duration | String | | participant | String[] | | http status code | :id => conference id |
| PUT | update join users | /conference/:id/join | |  |  | | --- | --- | | field | Type | | userId | String | | isJoin | Boolean | | http status code | :id => conference id |
| DELETE | delete conference | /conference/:id/delete | - | http status code |  |
| Contact | ~~POST~~ | ~~create contact~~ | ~~/contact~~ | |  |  | | --- | --- | | ~~field~~ | ~~type~~ | | ~~Id~~ | ~~String~~ | | ~~users~~ | ~~String[]~~ | | ~~http status code~~ |  |
| GET | get contact list | /contact/:id/list | - | http status code  contact data list json | :id => user id |
| PUT | update contact | /contact/:id/update | |  |  | | --- | --- | | field | type | | Id | String | | users | String[] | | http status code | :id => user id |
| DELETE | delete contact | /contact/:id/delete | - | http status code | :id => user id |

**### socket api**

\*It may be updated.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| API | Port | Event Name | Param | Return message | How it works |
| Ping/Echo | 3001 | pingEcho | id | success  -> “ok”  missed id(userId)  -> “please input a id(userId)” | 1. When the server starts, the socket opens  2. The client try connection to 3001 port after login  3. The client send event with event name “pingEcho” and id(userId) in parameter.  4. The server return “ok” if normal case  5. The client send event every xx seconds  6. The server update every ping/echo message time. And check the gap of message time |
| Conference Call Notification | 3002 | callNotification | - | conferenceId | 1. When the server starts, the socket opens  2. The client try connection to 3002 port after login  3. The server send event to all clients when one of conferences join user data is update.  4. All connected clients receive message from the server. The message is conference id  5. Each client check if need to conference join or not |

\*Ping/Echo example(POST MAN)

