

This manual describes the setting up the application and usage of the tasks implemented.

Pre requisities

- Java 8 should be installed
- Postgres Sql. Please use the default port 5432 for postgres. Please provide the user credentials as below to avoid editing application.properties and rebuild of jar.

User Name: postgres

Password : test@1234

To run the Spring Boot application execute the below command from the project services folder location in command line

java -jar target/services-0.0.1-SNAPSHOT.jar

INFO : *The postgres connection configuration are mentioned in the application.properties in the location
\\services\\src\\main\\resources\\application.properties.*

INFO: *To build the jar, execute the command **mvnw clean package** from the project services folder location in command line. This is not required if the postgres credentials are given as mentioned above. Please comment the test class ServiceApplicationTests if jar build is required.*

INFO : *Postman client was used for testing the API. Junit Test cases were implemented for ensuring the proper working of API .*

Please find the APIS of the task described below :

Task 1: Implement backend for saving, updating, listing and deleting connection details to you favourite relational database.

- *Create a user name, password and a database for the user*

--request POST "http://localhost:8080/api/v1/users/"

--parameter

```
{  
    "databaseName":"taskdbdemo",  
    "userName":"userdemo",  
    "password":"pas1234"  
}
```

--response

```
{"created":true}
```

- *Get All user - Gets the list of username and password*

--request GET "http://localhost:8080/api/v1/listusers"

--response [{"databaseName":null,"userName":"dbuser12new123",
"password":"md595a4f36f3de5619b68f6d93b3e143aad",
"databases":["taskdbtest"]},
{"databaseName":null,"userName":"testermanu1",
"password":"md5574b6b41820bf588ba7918dd559cf279",
"databases":["testingdb2"]}]

- *List a user - Gets an specific user name and the password details*

```
--request GET "http://localhost:8080/api/v1/listusers/{username}"

--parameter {username} - User Name

--response  {"databaseName":null,"userName":"meera",
            "password":"md59183783ed042b7eb1baabd8847014373",
            "databases":["meeradb"]}
```

- *Update a user name and password*

```
--request PUT "http://localhost:8080/api/v1/users/{name}"

--parameter {name} - User Name which needs to be updated

--parameter - Details to be updated with

    {
        "userName":"userdbmodify",
        "password":"pas1234"
    }

--response

    {"updated":true}
```

- *Delete a user with an username --request DELETE*

```
"http://localhost:8080/api/v1/deletuser/{userName}"

--parameter {userName} - name of the user
```

`--response {"deleted": true}`

- *Gets the databases owned by a user* --request GET

`http://localhost:8080/api/v1/ database/{userName}"`

`--parameter {userName} - name of the user`

`--response ["meeradb"]`

- *Rename the database* --request PUT

`"http://localhost:8080/api/v1/ database {oldDbName}/{newdDbName}`

`--parameter { oldDbName } – old database name`

`{newdDbName} – new database name`

`--response {"renamed": true}`

Task 2: Design and implement REST API for browsing structure and data using your stored database connections from Task 1.

- *Listing schemas in database*

`--request POST "http://localhost:8080/api/v2/schemas"`

`--parameter`

`{`

`"databaseName":"taskdb",`

`"userName":"postgres",`

`"password":"test@1234"`

```
}
```

--response

```
["information_schema","myschema","pg_catalog","public"]
```

- *Listing tables in database*

--request POST "http://localhost:8080/api/v2/tables"

--parameter

```
{
```

```
    "databaseName":"taskdb",
```

```
    "userName":"postgres",
```

```
    "password":"test@1234"
```

```
}
```

--response

```
["company","dbuser"]
```

- *Listing columns in database*

--request POST "http://localhost:8080/api/v2/columns/{tableName}"

--parameter

```
{
```

```
    "databaseName":"taskdb",
```

```
    "userName":"postgres",
```

```
    "password":"test@1234"
```

```
}
```

{tableName} - name of table

--response

```
["id","name","hostname","port",  
"databasename","username","password"]
```

- *Data preview of the table*

--request POST

"http://localhost:8080/api/v2/data/{tableName}"

--Parameter

{tableName} - name of table

-- response

```
[{"columnType":"text","columnName":"hostname","isPrimaryKey":false},  
{"columnType":"bpchar","columnName":"password","isPrimaryKey":false},  
{"columnType":"int4","columnName":"port","isPrimaryKey":false},  
{"columnType":"bpchar","columnName":"databasename","isPrimaryKey":f  
alse},  
{"columnType":"bpchar","columnName":"name","isPrimaryKey":false},  
{"columnType":"int4","columnName":"id","isPrimaryKey":true},  
{"columnType":"bpchar","columnName":"username","isPrimaryKey":false}]
```

Task 3: Design and implement REST API endpoints for statistics:

- *Single endpoint for statistics about each column: min, max, avg, median value of the column.*

--request POST

"http://localhost:8080/api/v3/statistics/{tableName}/{columnName}"

--Parameter

{tableName} - name of the table

{columnName} - name of the column

```
{  
    "databaseName":"taskdb",  
    "userName":"postgres",  
    "password":"test@1234"  
}
```

-- response

```
{"minValue":"8","maxValue":"37",  
  "avgValue":"22.6206896551724138","medianValue":"23"}
```

- Single endpoint for statistics about each table: number of records, number of attributes.

--request POST

"http://localhost:8080/api/v3/statistics/table/{tableName}"

-Parameter

{tableName} - name of the table

{

"databaseName":"taskdb",

"userName":"postgres",

"password":"test@1234"

}

--response

{"recordCount":"29","attributeCount":7}