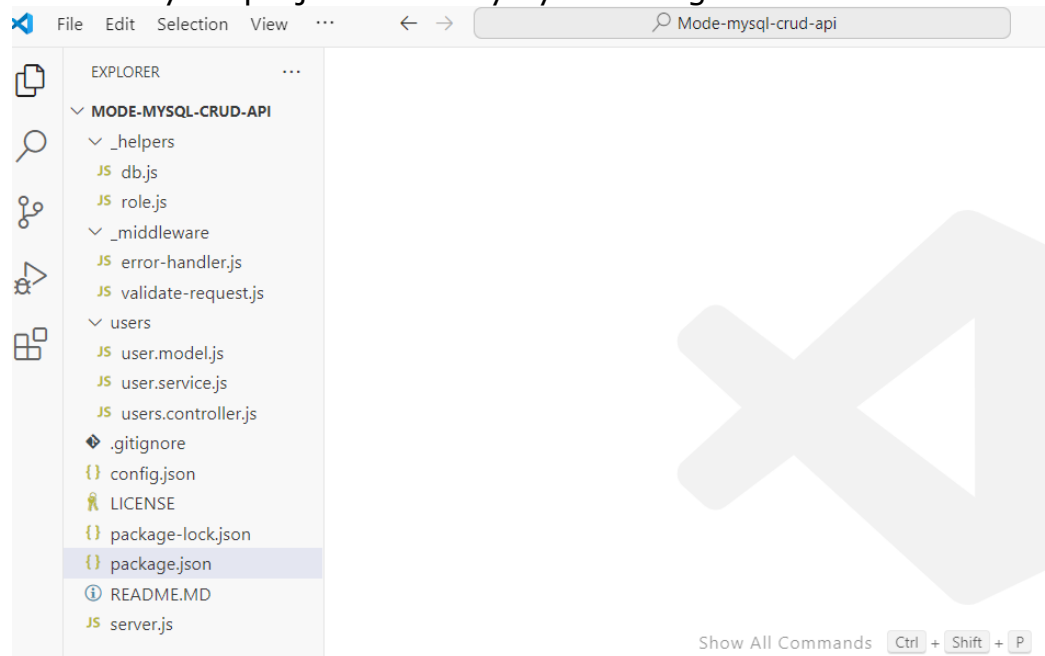


TypeScript, Node.js, Sequelize (ORM) and MySQL - CRUD API.

Step 1: Getting started with NodeJS.

Note: to use the commands within the node package manager you need to have the NodeJS installed on your system. For reference click the link as follows: <https://nodejs.org/>.

Kickstart your project directory by following the same structure below:



Install and initialize the node package manager by typing the following commands on your terminal just within your project folder.

```
C:\Windows\System32\cmd.e  x  +  v

Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\krist\Desktop\Mode-mysql-crud-api>npm init -y
Wrote to C:\Users\krist\Desktop\Mode-mysql-crud-api\package.json:

{
  "name": "mode-mysql-crud-api",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Edit your package.json file to look like this.

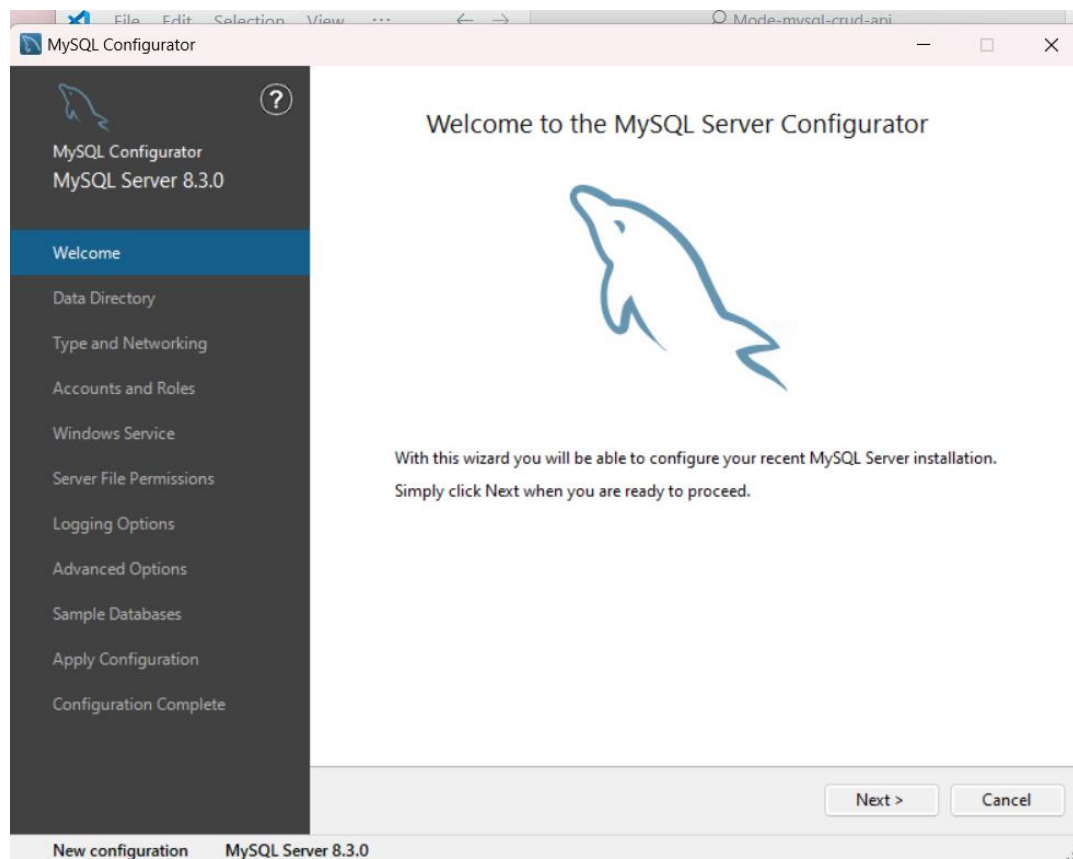
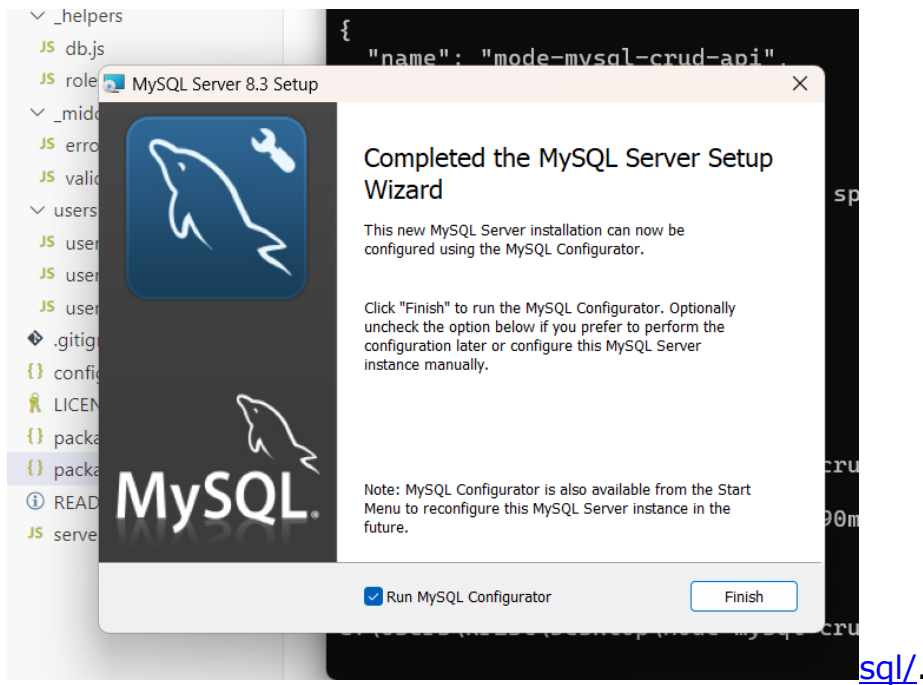
```
{ } package.json X
{ } package.json > ...
1  {
2    "name": "mode-mysql-crud-api",
3    "version": "1.0.0",
4    "license": "MIT",
5    "scripts": {
6      "start": "node ./server.js",
7      "start:dev": "nodemon ./server.js"
8    }
9  }
10 }
```

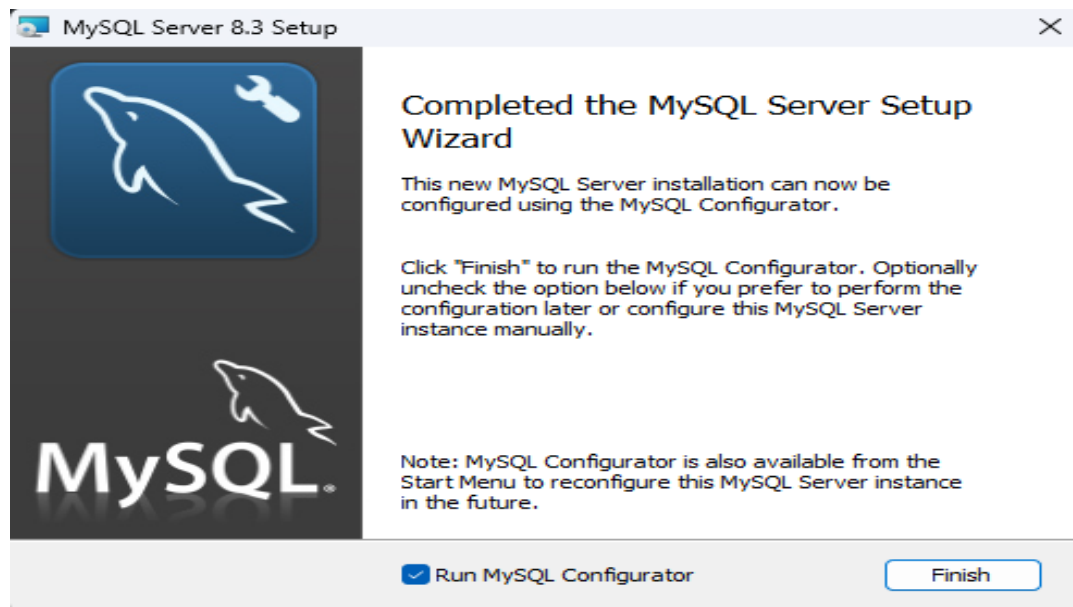
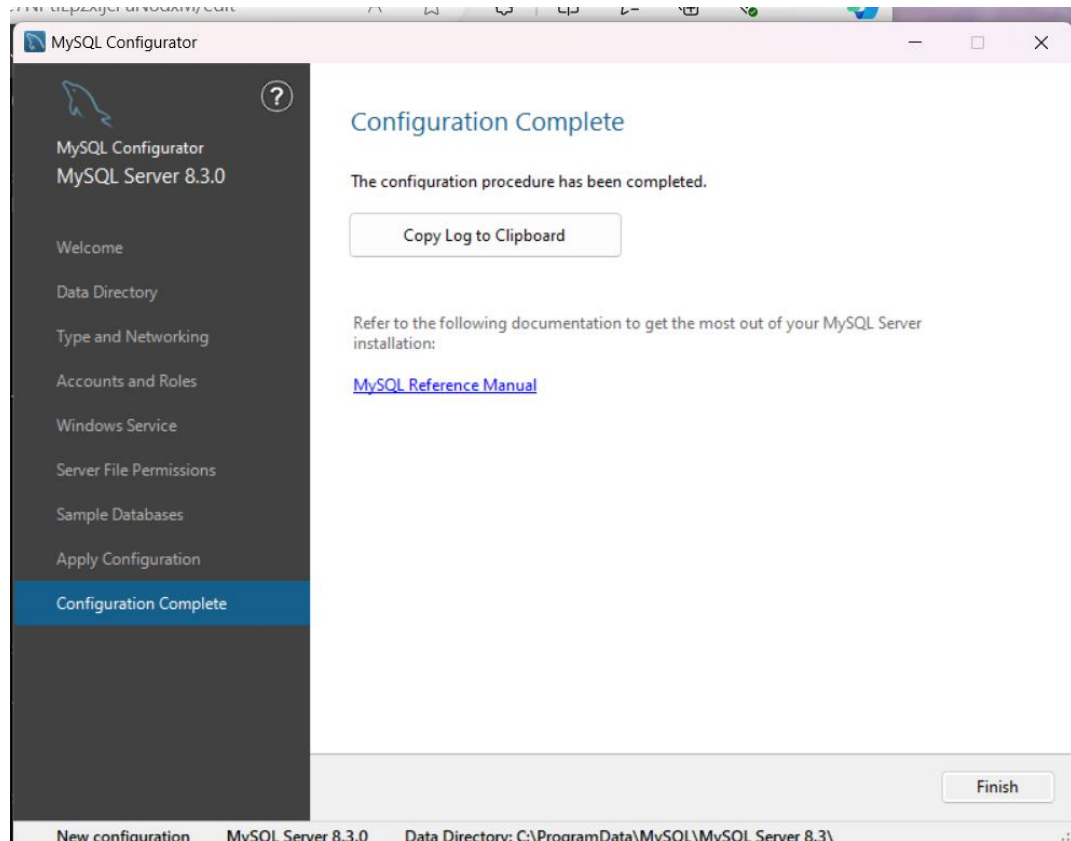
And then afterwards run npm install.

```
C:\Users\krist\Desktop\Mode-mysql-crud-api>npm install
up to date, audited 1 package in 290ms
found 0 vulnerabilities
C:\Users\krist\Desktop\Mode-mysql-crud-api>
```

Step 2: Setting up MySQL

You must have a MySQL server instance for the API to connect to. The Community Server version is available to download for free from <https://dev.mysql.com/downloads/my>





Further Post-Installation instructions available at
<https://dev.mysql.com/doc/refman/8.0/en/installing.html>.

Step 3: Installing project dependencies.

Your Node.js project requires dependencies to use Sequelize and be able to interact with MySQL. On your terminal, install them like so:

- `npm i sequelize mysql2`

You will also need the help of other dependencies for your project to work:

- `npm i express cors bcryptjs joi rootpath`

Install this package to power up your development workflow:

- `npm i -D nodemon`



```
{  
  "name": "mode-mysql-crud-api",  
  "version": "1.0.0",  
  "license": "MIT",  
  "scripts": {  
    "start": "node ./server.js",  
    "start:dev": "nodemon ./server.js"  
  },  
  "dependencies": {  
    "bcryptjs": "^2.4.3",  
    "cors": "^2.8.5",  
    "express": "^4.18.3",  
    "joi": "^17.12.2",  
    "mysql2": "^3.9.2",  
    "rootpath": "^0.1.2",  
    "sequelize": "^6.37.1"  
  },  
  "devDependencies": {  
    "nodemon": "^3.1.0"  
  }  
}
```

Step 4: Filling up the contents of your JavaScript and JSON files.

Helpers Folder

Path: `/_helpers/db.js`

```
{ } package.json JS db.js •
_helpers > JS db.js > initialize
1  const config = require('config.json');
2  const mysql = require('mysql2/promise');
3  const { Sequelize } = require('sequelize');
4
5  module.exports = db = {};
6
7  initialize();
8
9  async function initialize() {
10     const { host, port, user, password, database } = config.database;
11     const connection = await mysql.createConnection({host, port, user, password,});
12     await connection.query(`CREATE DATABASE IF NOT EXISTS \`${database}\`;`);
13
14     const sequelize = new Sequelize(database, user, password, { dialect: 'mysql'});
15
16     db.user = require('../users/user.model')(sequelize);
17
18     await sequelize.sync({ alter:true });
19 }
```

Path: /_helpers/role.js

```
JS db.js JS role.js X
_helpers > JS role.js > <unknown>
1  module.exports = {
2      Admin: 'Admin',
3      User: 'User'
4  }
```

Express.js Middleware Folder

Path: /_middleware/error-handler.js

Path: /_middleware/validate-request.js

```
JS error-handler.js X
_middleware > JS error-handler.js > errorHandler
1  module.exports = errorHandler;
2
3  function errorHandler(err, req, next) {
4    switch (true) {
5      case typeof err == 'string':
6        const is404 = err.toLowerCase().endsWith('not found');
7        const statusCode = is404 ? 404 : 400;
8        return res.status(statusCode).json({ message: err });
9      default:
10       return res.status(500).json({ message: err.message});
11    }
12  }
```

```
JS validate-request.js X
_middleware > JS validate-request.js > validateRequest
1  module.exports = validateRequest;
2
3  function validateRequest(req, next, schema) {
4    const options = {
5      abortEarly: false,
6      allowUnknown: true,
7      stripUnknown : true,
8    };
9    const { error, value } = schema.validate(req.body, options);
10    if (error) {
11      next(`Validation error: ${error.details.map(x => x.message).join(',')}`);
12    }
13    else{
14      req.body = value;
15      next();
16    }
17  }
```

Users Feature Folder

Path: /users/user.model.js

```
JS user.model.js X
users > JS user.model.js > model
1  const { DataTypes } = require('sequelize');
2
3  module.exports = model;
4
5  function model(sequelize) {
6    const attributes = {
7      email: { type: DataTypes.STRING, allowNull: false },
8      passwordHash: { type: DataTypes.STRING, allowNull: false },
9      title: { type: DataTypes.STRING, allowNull: false },
10     firstName: { type: DataTypes.STRING, allowNull: false },
11     lastName: { type: DataTypes.STRING, allowNull: false },
12     role: { type: DataTypes.STRING, allowNull: false }
13   };
14
15   const options = {
16     defaultScope: {
17       attributes: { excluded: ['passwordHash'] }
18     },
19     scopes: {
20       withHash: { attributes: {}, }
21     }
22   };
23
24   return sequelize.define('user', attributes, options);
25 }
```






Path: /users/user.service.js

```
File Edit Selection View Go ... Mode-mysql-crud-api
EXPLORER
MODE-MYSQL-CRUD-API
  _helpers
    db.js
    role.js
  _middleware
    error-handler.js
    validate-request.js
  node_modules
  users
    user.model.js
    user.service.js
    users.controller.js
  .gitignore
  config.json
  LICENSE
  package-lock.json
  package.json
  README.MD
  server.js
JS user.service.js X
users > JS user.service.js > getUser
1  const bcrypt = require('bcryptjs');
2  const db = require('_helpers/db');
3
4
5  module.exports = {
6    getAll,
7    getById,
8    create,
9    update,
10   delete: _delete
11 };
12
13 async function getAll(){
14   return await db.User.findAll();
15 }
16
17 async function create(params) {
18   if (await db.User.findOne({ where: { email: params.email } })) {
19     throw 'Email "' + params.email + '" is already registered';
20   }
21
22   const user = new db.User(params);
23
24   user.passwordHash = await bcrypt.hash(params.password, 10);
25
26   await user.save();
27 }
```




```
27 }
28
29 async function update(id, params) {
30   const user = await getUser(id);
31
32   const usernameChanged = params.username && user.username !== params.username;
33   if (usernameChanged && await db.User.findOne({ where: { username: params.username } })) {
34     throw 'Username "' + params.username + '" is already taken';
35   }
36
37   if (params.password) {
38     params.passwordHash = await bcrypt.hash(params.password, 10);
39   }
40
41   Object.assign(user, params);
42   await user.save();
43 }
44
45 async function _delete(id) {
46   const user = await getUser(id);
47   await user.destroy();
48 }
49
50
51
52 async function getUser(id) {
53   const user = await db.user.findByPk(id);
54   if (!user) throw 'User not found';
55   return user;
56 }
```

Path: /users/users.controller.js



EXPLORER

...

MODE-MYSQL-CRUD-API

✓ _helpers

JS db.js

JS role.js

✓ _middleware

JS error-handler.js

JS validate-request.js


> node_modules

✓ users


JS user.model.js

JS user.service.js


JS users.controller.js




.gitignore




config.json




LICENSE



package-lock.json



package.json



README.MD


JS server.js

JS users.controller.js

X

users > JS users.controller.js > ...

```
1  const express = require('express');
2  const router = express.Router();
3  const joi = require('joi');
4  const validateRequest = require('_middleware/validate-request');
5  const Role = require('._helpers/role');
6  const userService = require('./user.service');
7  const { validate } = require('uuid');
8
9  router.get('/', getAll);
10 router.get('/:id', getById);
11 router.post('/', createSchema, create);
12 router.put('/:id', updateSchema, update);
13 router.delete('/:id', _delete);
14
15 module.exports = router;
16
17 function getAll(req, res, next) {
18   userService.getAll()
19     .then(users => res.json(users))
20     .catch(next);
21 }
22
23
24 function getById(req, res, next) {
25   userService.getById()
26     .then(user => res.json(user))
27     .catch(next);
28 }
29
30 function create(req, res, next) {
31   userService.create(req.body)
32     .then(() => res.json({ message: 'User created'}))
33     .catch(next);
34 }
```



```

35
36 ✓ function update(req, res, next) {
37     userService.update(req.params.id, req.body)
38     .then(() => res.json({message: 'User updated'}))
39     .catch(next);
40 }
41
42
43 ✓ function _delete(req, res, next) {
44     userService.delete(req.params.id)
45     .then(() => res.json({message: 'User updated'}))
46     .catch(next);
47 }
48
49 ✓ function createSchema(req, res, next) {
50     ✓ const schema = Joi.object( {
51         title: Joi.string().required(),
52         firstName: Joi.string().required(),
53         lastName: Joi.string().required(),
54         role: Joi.string().valid(Role.Admin, Role.User).required(),
55         email: Joi.string().email().required(),
56         password: Joi.string().min(6).required(),
57         confirmPassword: Joi.string().valid(Joi.ref('password')).required(),
58     });
59
60     validateRequest(req, res, next)
61 }
62
63
64 ✓ function updateSchema(req, res, next) {
65     ✓ const schema = Joi.object({
66         title: Joi.string().empty(''),
67         firstName: Joi.string().empty(''),
68         lastNameName: Joi.string().empty(''),
69         role: Joi.string().valid(Role.Admin, Role.User).empty(''),
70         email: Joi.string().email().empty(''),
71         password: Joi.string().min(6).empty(''),
72         confirmPassword: Joi.string().valid(('password')).empty('')
73     }).with('password', 'confirmPassword');
74
75     validateRequest(req, req, next);
76 }
77

```

6pi Conffib

Dath: /config.json



The image shows a VS Code editor window. On the left, the file explorer is open, showing a project structure for 'MODE-MYSQL-CRUD-API'. The files listed are: `_helpers` (containing `db.js` and `role.js`), `_middleware` (containing `error-handler.js` and `validate-request.js`), `node_modules`, `users` (containing `user.model.js`), and `config.json`. The `config.json` file is selected and its content is displayed in the main editor area. The content of `config.json` is a JSON object with a `database` property containing an object with `host`, `port`, `user`, `password`, and `database` fields.

```
{
  "database": {
    "host": "localhost",
    "port": 3306,
    "user": "root",
    "password": "",
    "database": "node-mysql-crud-api"
  }
}
```

Server Startup File

Dath: /server.js



The image shows a VS Code editor window with the `server.js` file open. The file content is as follows:

```
1 require('rootpath')();
2 const express = require('express');
3 express();
4 const cors = require('cors');
5 const errorHandler = require('_middleware/error-handler');
6
7
8 app.use(express.json());
9 app.use(express.urlencoded({ extended: true }));
10 app.use(cors());
11
12 app.use('/users', require('./users/users.controller'));
13
14 app.use(errorHandler);
15
16 const port = process.env.MODE_ENV === 'production' ? (process.env.PORT || 80): 4000;
17 app.listen(port, () => console.log('Server listening on port' + port));
18
19
```

Step 4: Testinb the Node + MySQL CRUD 6DI

- Make sure that all required npm packages are installed.
- Update the database credentials in `/config.json` to connect to your MySQL server instance, and ensure MySQL server is running.
- Start the API by running `npm start` (or `npm run start:dev` to start with nodemon) from the command line in the project root folder, you should see the message `Server listening on port 4000`.

```

Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\krist\Desktop\Mode-mysql-crud-api>npm run start

> mode-mysql-crud-api@1.0.0 start
> node ./server.js

Server listening on port4000
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = '
users' AND TABLE_SCHEMA = 'node-mysql-crud-api'
Executing (default): CREATE TABLE IF NOT EXISTS `users` (`id` INTEGER NOT NULL auto_increment , `email` VARCHAR(255) NOT
NULL, `passwordHash` VARCHAR(255) NOT NULL, `title` VARCHAR(255) NOT NULL, `firstName` VARCHAR(255) NOT NULL, `lastName`
VARCHAR(255) NOT NULL, `role` VARCHAR(255) NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PRI
MARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `users`
|

```

MySQL Command Line Interface (Showing the database and table automatically created)

```

Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| node-mysql-crud-api |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> use node-mysql-crud-api;
Database changed
mysql> show tables;
+-----+
| Tables_in_node-mysql-crud-api |
+-----+
| users |
+-----+
1 row in set (0.00 sec)

```

To show available databases, type in “show databases;”
And “show tables;” to look for the tables created.

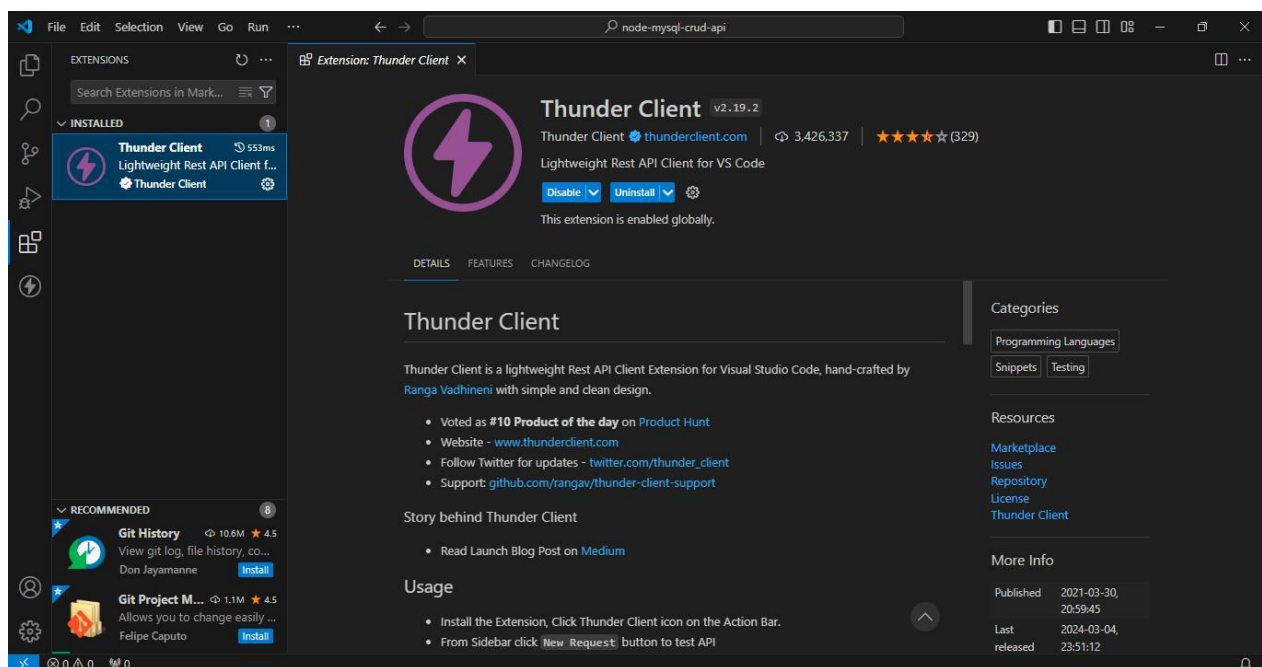
“desc users;” is use to describe table definition

```
mysql> desc users;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
email	varchar(255)	NO		NULL	
passwordHash	varchar(255)	NO		NULL	
title	varchar(255)	NO		NULL	
firstName	varchar(255)	NO		NULL	
lastName	varchar(255)	NO		NULL	
role	varchar(255)	NO		NULL	
createdAt	datetime	NO		NULL	
updatedAt	datetime	NO		NULL	

```
9 rows in set (0.01 sec)
```

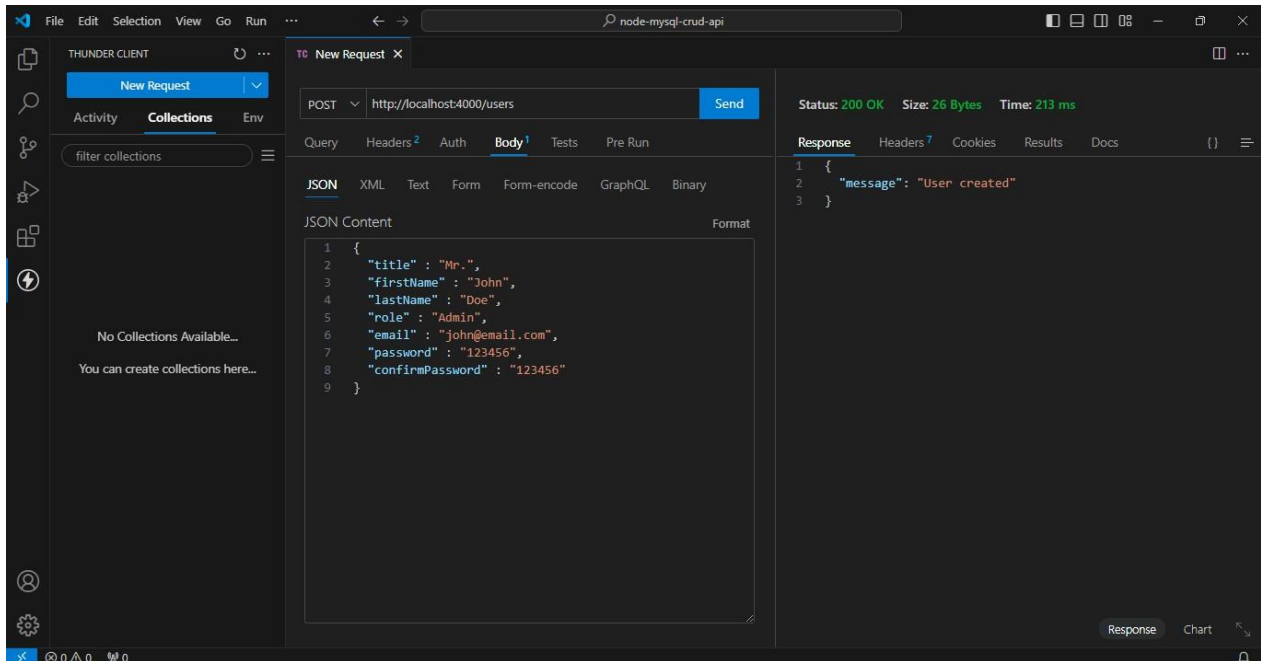
- You can test the API directly with a tool such as Postman or VSCode extension ThunderClient. This time around we use Thunder Client.



Making Requests in Thunder Client

- Click on the new request button
- You can configure the request url and what type of request you want to send on the page next to the activity/collections/env tab.
- If you are done with the configurations then click the send button.

Creating a new user



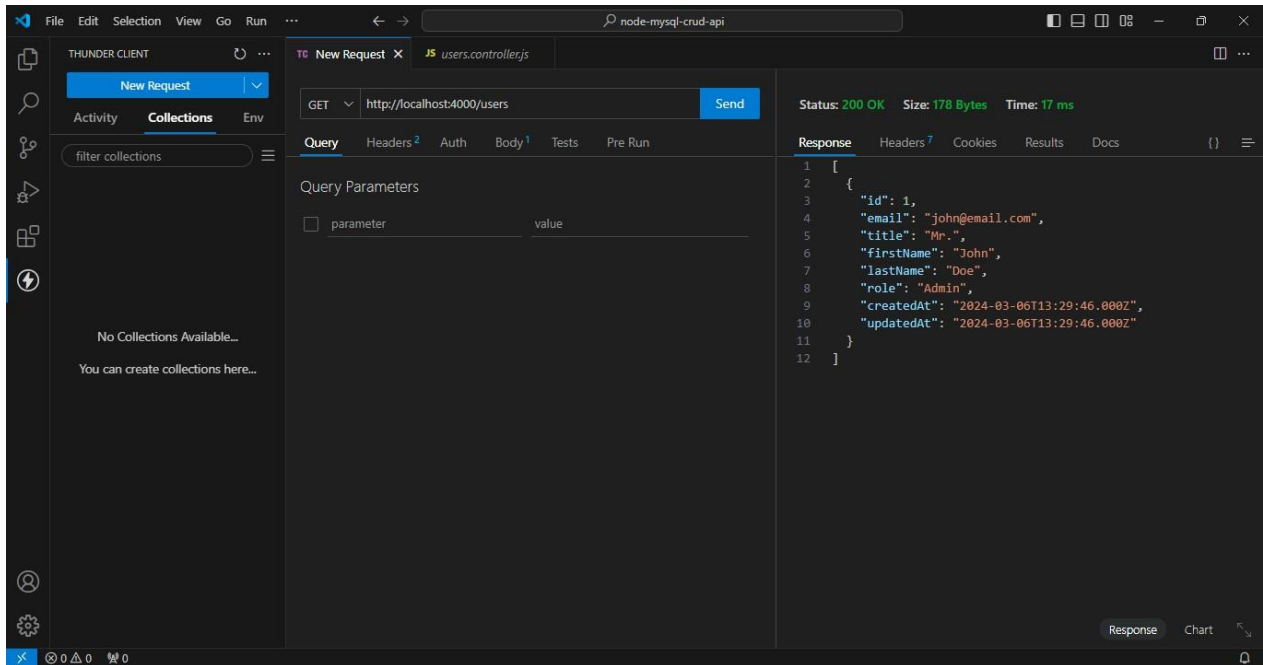
Changes reflected on the database after creating a user.

```
MySQL 8.3 Command Line Cli X
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| node-mysql-crud-api |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

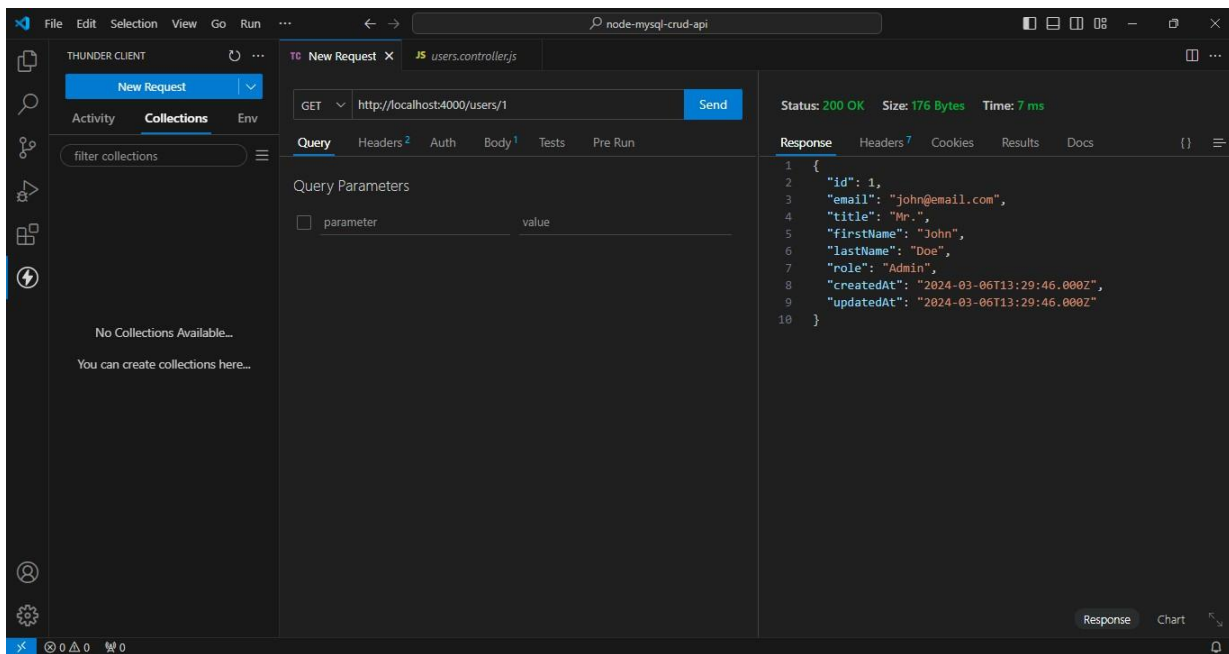
mysql> use node-mysql-crud-api;
Database changed
mysql> show tables;
+-----+
| Tables_in_node-mysql-crud-api |
+-----+
| users |
+-----+
1 row in set (0.00 sec)

mysql> select * from users;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | email | passwordHash | title | firstName | lastName | role | createdAt |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | john@email.com | $2a$10$WbxOVoeUm4xAPA04i.SYu4EGH0kNEBmP/NC422gie2epWNgC7jZO | Mr. | John | Doe | Admin | 2024-03-06 13:29:46 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

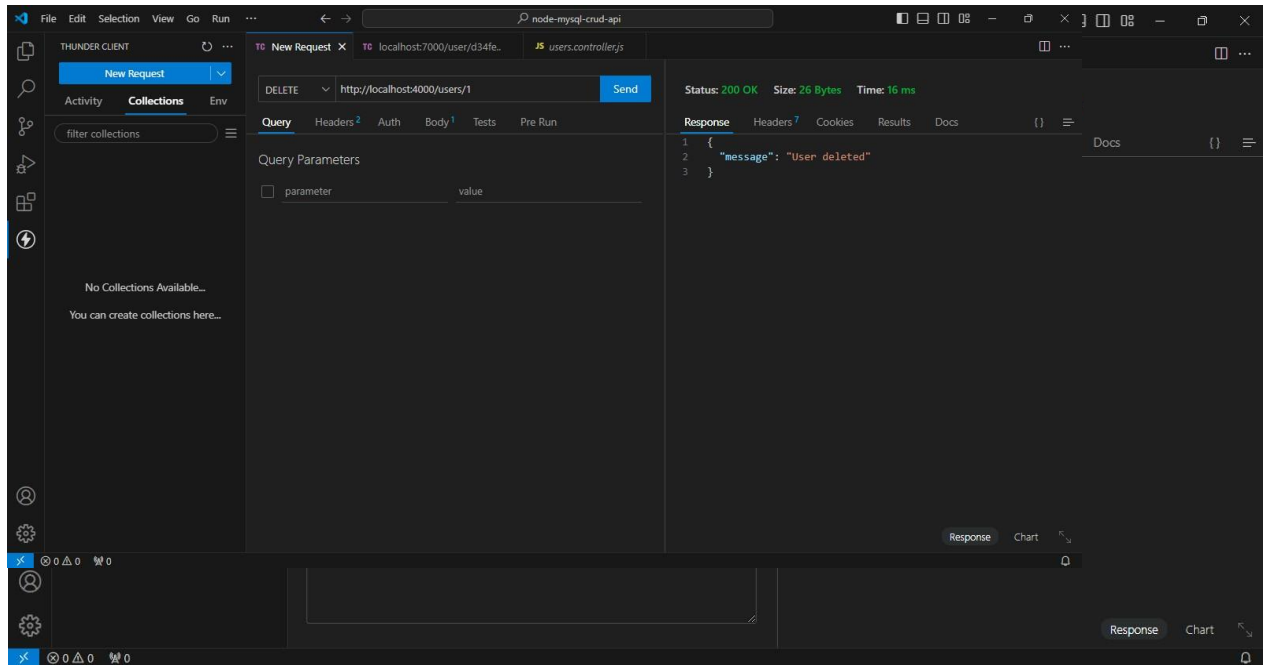
Get all users



Get a user by ID



Update a user by ID



Changes reflected on the database after updating a user.

```
mysql> select * from users;
+-----+-----+-----+-----+-----+-----+-----+
| id | email | passwordHash | title | firstName | lastName | role | createdAt |
+-----+-----+-----+-----+-----+-----+-----+
| 3 | lemonj@email.com | $2a$10$nYvIpPwPTUerUhZ0D7PTW.FQHza.U18k0PdZ6Ti3vZ9cDIGF.M9WM | Mr. | John | Lemon | Admin | 2024-03-06 14:17:28 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> |
```

Delete a user by ID

Changes reflected on the database after deleting a user.

```
mysql> select * from users;
```

id	email	passwordHash	title	firstName	lastName	role	createdAt	updatedAt
1	john@email.com	\$2a\$10\$Wbx0VvoeUm4xAPA04i.SYu4EGH0kNEBmP/NC422gie2epWNgC7jZ0	Mr.	John	Doe	Admin	2024-03-06 13:29:46	

```
1 row in set (0.00 sec)

mysql> select * from users;
Empty set (0.00 sec)

mysql>
```

Terminal Logs after performing CRUD operations

```
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User' WHERE 'User'.e
mail = 'john@email.com' LIMIT 1;
Executing (default): INSERT INTO 'Users' ('id','email','passwordHash','title','firstName','lastName','role','createdAt','updatedAt') VALUES (DEFAULT
,?,?,?,?,?,?,?);
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User';
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User';
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User';
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User' WHERE 'User'.i
d = '3';
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User' WHERE 'User'.i
d = '3';
Executing (default): UPDATE 'Users' SET 'lastName'=?, 'email'=?, 'passwordHash'=?, 'updatedAt'=? WHERE 'id' = ?
Executing (default): SELECT 'id', 'email', 'title', 'firstName', 'lastName', 'role', 'createdAt', 'updatedAt' FROM 'Users' AS 'User' WHERE 'User'.i
d = '3';
Executing (default): DELETE FROM 'Users' WHERE 'id' = 3
```

