LU11c - CRUD Server and Postman

Learning Objectives

1. How to connect the server to the db
2. How to fetch data from the db and display it in POSTMAN
3. How to perform the CRUD operations on the server
4. How to perform the CRUD operations by using PoSTMAN

Sources

- SampleServer that can perform CRUD

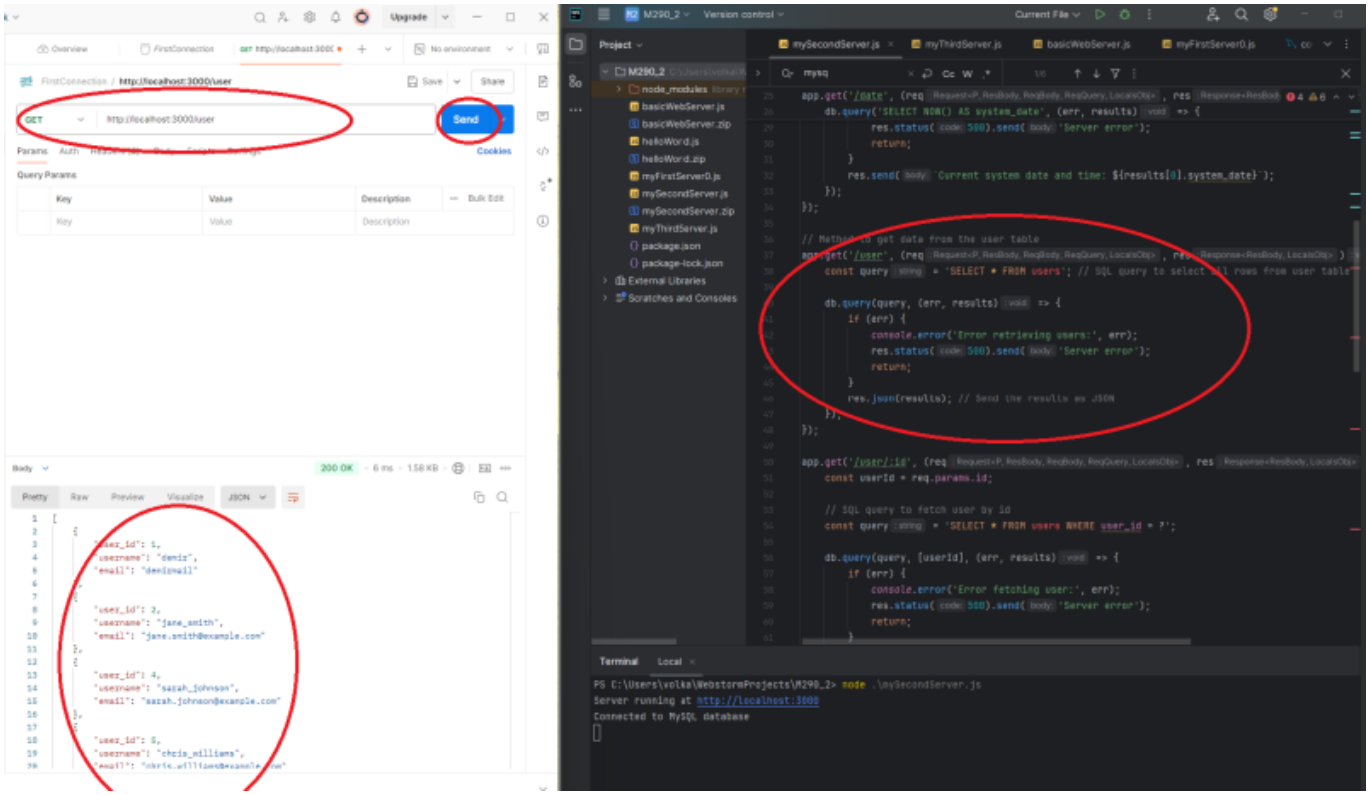
Introduction

At last chapter of our database journey we want to perform all CRUD operations within the server AND the client (POSTMAN). For reasons of efficiency we are going to start with the R = READ of CRUD.

R - READ: Fetching list of data

First of all, we want to display the list of users, which are store in the table users. The following code show how to get that list from the database.

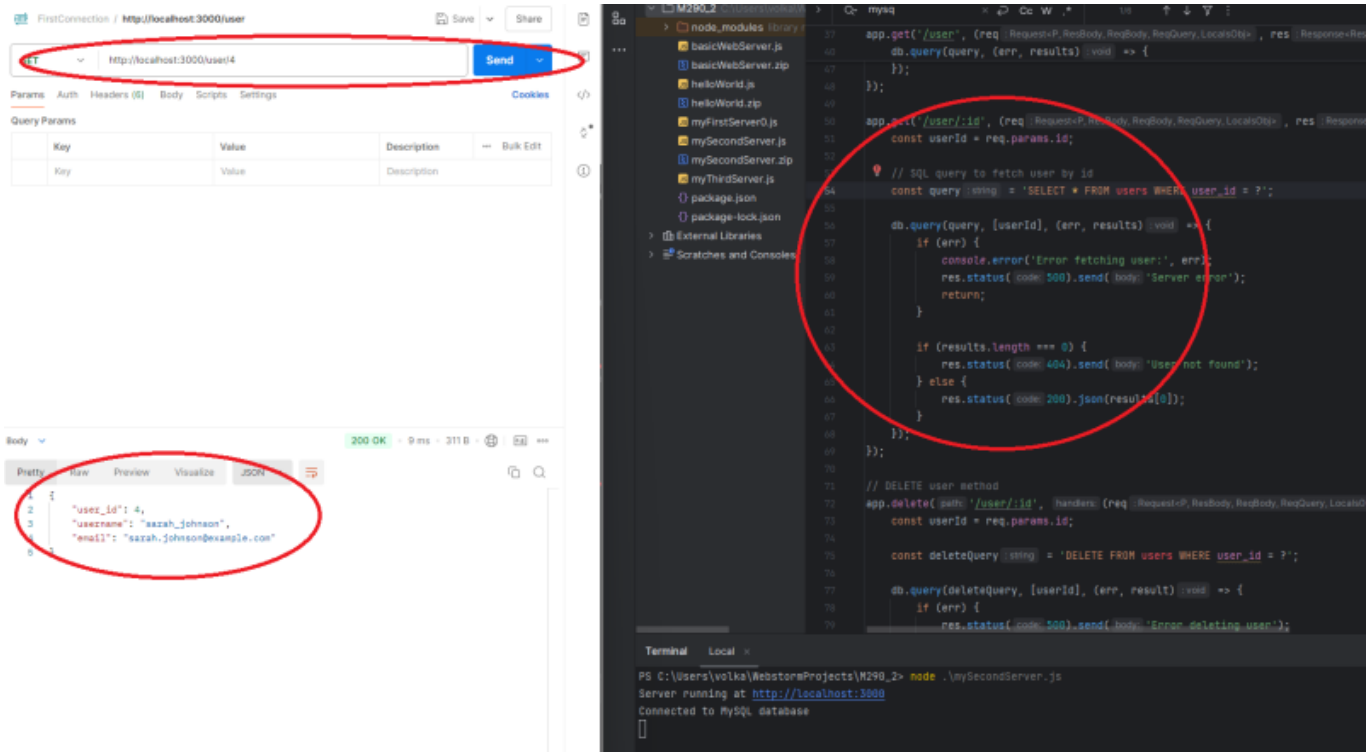
```
// Method to get data from the user table
app.get('/user', (req, res) => {
    const query = 'SELECT * FROM users'; // SQL query to select all rows
from user table
    db.query(query, (err, results) => {
        if (err) {
            console.error('Error retrieving users:', err);
            res.status(500).send('Server error');
            return;
        }
        res.json(results); // Send the results as JSON
    });
});
```



R - READ: Getting one speffic row of data

If we want one speficific individual to be displayed, we need to make some changes, such as filtering a user_id, as shown in the following code below.

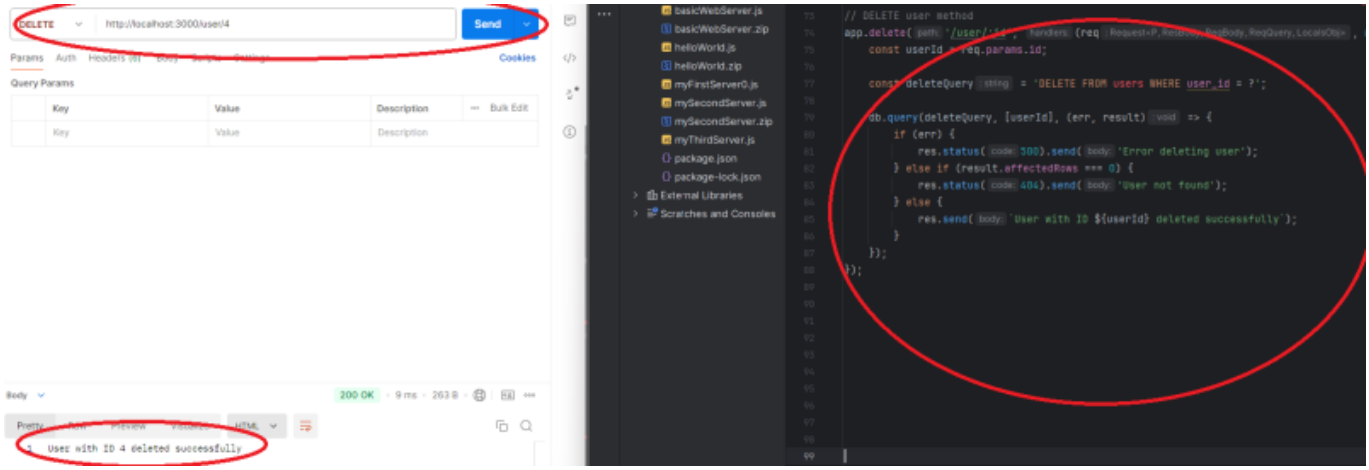
```
app.get('/user/:id', (req, res) => {
    const userId = req.params.id;
    // SQL query to fetch user by id
    const query = 'SELECT * FROM users WHERE user_id = ?';
    db.query(query, [userId], (err, results) => {
        if (err) {
            console.error('Error fetching user:', err);
            res.status(500).send('Server error');
            return;
        }
        if (results.length === 0) {
            res.status(404).send('User not found');
        } else {
            res.status(200).json(results[0]);
        }
    });
});
```



D - Deletion of one specific record

If we want to delete one specific row of data, we need another method to perform that.

```
app.delete('/user/:id', (req, res) => {
    const userId = req.params.id;
    const deleteQuery = 'DELETE FROM users WHERE user_id = ?';
    db.query(deleteQuery, [userId], (err, result) => {
        if (err) {
            res.status(500).send('Error deleting user');
        } else if (result.affectedRows === 0) {
            res.status(404).send('User not found');
        } else {
            res.send(`User with ID ${userId} deleted successfully`);
        }
    });
});
```



U - Update of one specfic row

Well, if we want to update one row of data, we need the http method PUT.

```
app.put('/user/:id', (req, res) => {
    const userId = req.params.id;
    const { username, email } = req.query; // Get username and email from
query parameters
    // Ensure that at least one field is provided
    if (!username && !email) {
        return res.status(400).send('At least one field (username or email)
must be provided for update.');
    }
    // Construct the update query
    const updateQuery = 'UPDATE users SET username = ?, email = ? WHERE
user_id = ?';
    const values = [username || null, email || null, userId]; // Use null
for any field not provided
    db.query(updateQuery, values, (err, result) => {
        if (err) {
            res.status(500).send('Error updating user');
        } else if (result.affectedRows === 0) {
            res.status(404).send('User not found');
        } else {
            res.send(`User with ID ${userId} updated successfully`);
        }
    });
});

{{:modul:m290:learningunits:lu09:theorie:crud_u.png?800| Update of one row
of data}}
```

C - CREATE: Insert a new row into the db

Finally, with the INSERT operation, we complete our CRUD requirements.

```
// insert a row
app.post('/user', (req, res) => {
    const { username, email } = req.query; // Get username and email from
query parameters
    // Ensure that both fields are provided
    if (!username || !email) {
        return res.status(400).send('Username and email are required for
insertion.');
    }
    // Construct the insert query
    const insertQuery = 'INSERT INTO users (username, email) VALUES (?, ?)';
    const values = [username, email]; // Use provided fields for insertion
```

```
    db.query(insertQuery, values, (err, result) => {
        if (err) {
            console.error('Error inserting user:', err);
            return res.status(500).send('Error inserting user');
        }
       // Respond with the ID of the newly created user
        res.status(201).send(`User added with ID: ${result.insertId}`);
    });
});
```