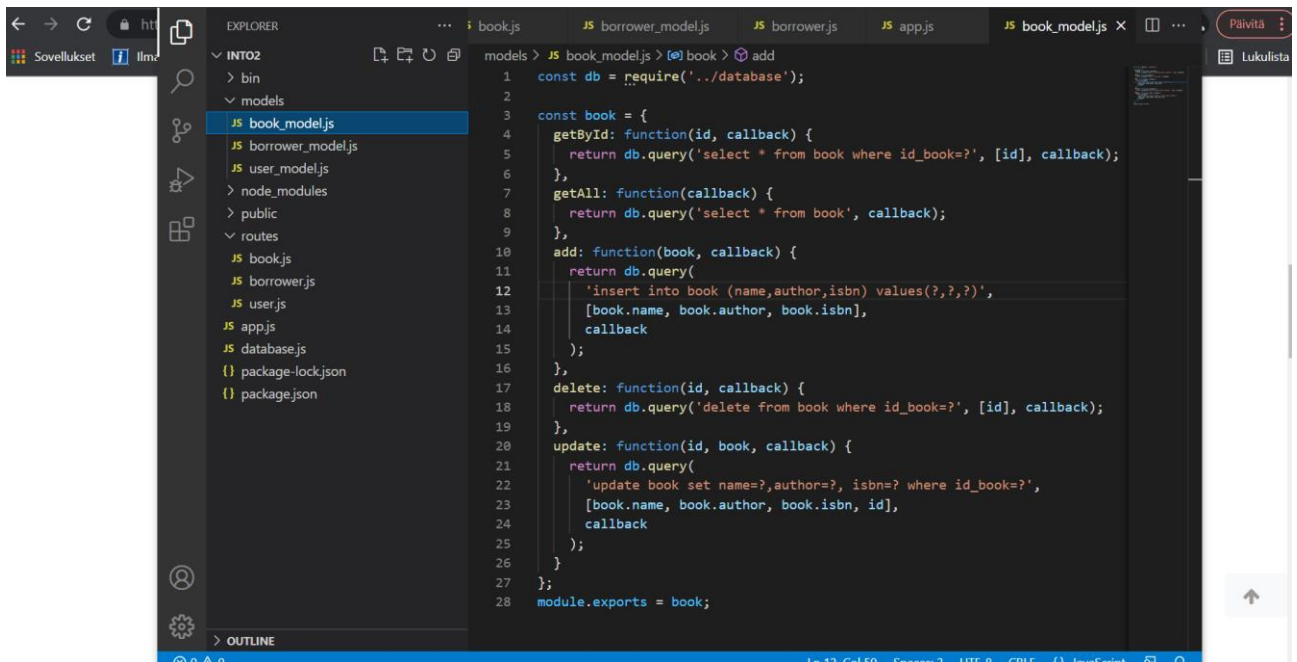


Tehtävä 17

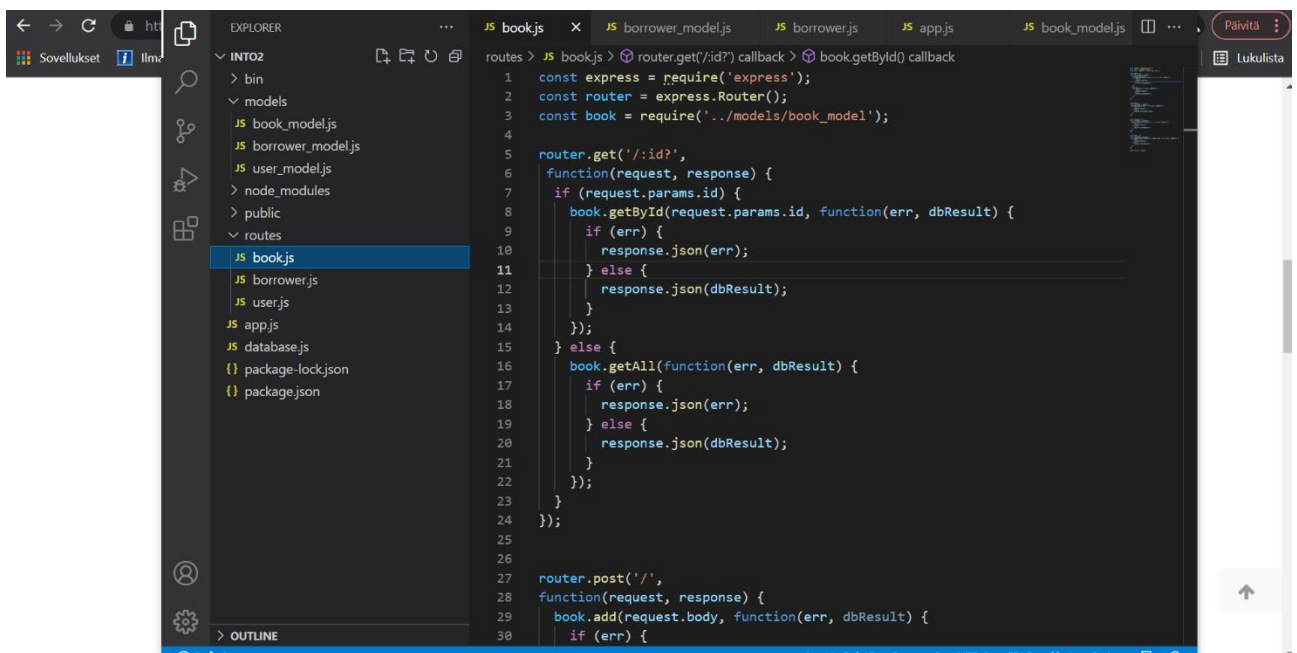
Anni Eno

Lähdekoodit



This screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The file `book_model.js` is selected under the `models` directory. The main editor window displays the code for `book_model.js`, which defines a database model for books with methods for getting, adding, deleting, and updating records.

```
1 const db = require('../database');
2
3 const book = {
4   getById: function(id, callback) {
5     return db.query('select * from book where id_book=?', [id], callback);
6   },
7   getAll: function(callback) {
8     return db.query('select * from book', callback);
9   },
10  add: function(book, callback) {
11    return db.query(
12      'insert into book (name,author,isbn) values(?,?,?)',
13      [book.name, book.author, book.isbn],
14      callback
15    );
16  },
17  delete: function(id, callback) {
18    return db.query('delete from book where id_book=?', [id], callback);
19  },
20  update: function(id, book, callback) {
21    return db.query(
22      'update book set name=?,author=?, isbn=? where id_book=?',
23      [book.name, book.author, book.isbn, id],
24      callback
25    );
26  }
27 };
28 module.exports = book;
```



This screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The file `book.js` is selected under the `routes` directory. The main editor window displays the code for `book.js`, which defines the Express.js routes for the book model, including endpoints for getting a book by ID, getting all books, and adding a new book.

```
1 const express = require('express');
2 const router = express.Router();
3 const book = require('../models/book_model');
4
5 router.get('/:id?',
6   function(request, response) {
7     if (request.params.id) {
8       book.getById(request.params.id, function(err, dbResult) {
9         if (err) {
10           response.json(err);
11         } else {
12           response.json(dbResult);
13         }
14       });
15     } else {
16       book.getAll(function(err, dbResult) {
17         if (err) {
18           response.json(err);
19         } else {
20           response.json(dbResult);
21         }
22       });
23     }
24   });
25
26 router.post('/',
27   function(request, response) {
28     book.add(request.body, function(err, dbResult) {
29       if (err) {
30         response.json(err);
31       } else {
32         response.json(dbResult);
33       }
34     });
35   });
```

GET

The screenshot shows the Postman application with a workspace named 'local_user'. A collection named 'Local_http' is expanded, showing a request named 'POST_Borrower'. The request is a GET method to the endpoint '({base_url})/borrower/'. The request body is set to 'x-www-form-urlencoded' and contains the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id_user	NULL	
<input checked="" type="checkbox"/> username	Root	
<input checked="" type="checkbox"/> password	admin	

The response is a 200 OK status with a 20 ms response time and 1.13 KB of data. The response body is shown in JSON format:

```
1 {
2   "id_borrower": 1,
3   "fname": "Mikki",
4   "lname": "Hiiri",
5   "streetAddress": "Ankkalinn 2"
6 }
```

POST

The screenshot shows the Postman application with the same workspace and collection. The request is now a POST method to the endpoint '({base_url})/borrower/'. The request body is set to 'x-www-form-urlencoded' and contains the following data:

Key	Value	Description
<input checked="" type="checkbox"/> fname	lines	
<input checked="" type="checkbox"/> lname	Ankka	
<input checked="" type="checkbox"/> streetAddress	Ankkalinn 111	

The response is a 200 OK status with a 59 ms response time and 1.06 KB of data. The response body is shown in JSON format:

```
1 {
2   "id_borrower": "NULL",
3   "fname": "Iines",
4   "lname": "Ankka",
5   "streetAddress": "Ankkalinn 111"
6 }
```

PUT (streetAddress päivitetiin)

The screenshot shows the Postman application with a workspace named 'local_user'. The selected collection is 'Local_http' and the endpoint is 'POST_Borrower'. The request method is 'PUT' and the URL is '({base_url})/borrower/3'. The request body is in 'x-www-form-urlencoded' format with the following data:

Key	Value	Description
fname	lines	
lname	Ankka	
streetAddress	Ankkalinn 12	

The response is a JSON object:

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "serverStatus": 2,
6   "warningCount": 0,
```

The status is 200 OK, 63 ms, 1.14 KB.

DELETE

The screenshot shows the Postman application with the same workspace and collection. The request method is 'DELETE' and the URL is '({base_url})/borrower/3'. The request body is in 'x-www-form-urlencoded' format with the following data:

Key	Value	Description
fname	lines	
lname	Ankka	
streetAddress	Ankkalinn 12	

The response is a JSON object:

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "serverStatus": 2,
6   "warningCount": 0,
```

The status is 200 OK, 28 ms, 1.1 KB.