

17: Tietokantayhteys ja CRUD Nodella ja Expressillä

Taulujen rakenne:

```
mysql> desc book;
```

Field	Type	Null	Key	Default	Extra
id_book	int	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
author	varchar(255)	YES		NULL	
isbn	varchar(20)	YES		NULL	

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

```
mysql> desc borrower;
```

Field	Type	Null	Key	Default	Extra
id_borrower	int	NO	PRI	NULL	auto_increment
fname	varchar(255)	YES		NULL	
lname	varchar(255)	YES		NULL	
streetaddress	varchar(255)	YES		NULL	

```
4 rows in set (0.00 sec)
```

Book GET toiminto:

Katsotaan mitä tietoa tauluun on jo laitettu.

The screenshot shows a web browser with a REST client interface. The URL bar shows `localhost:3000/book`. The request method is `GET`. The response status is `200 OK` with a time of `221 ms` and a size of `602 B`. The response body is a JSON array of book objects, displayed in a pretty-printed format.

```
{
  "id_book": 1,
  "name": "PHP Basic",
  "author": "Bob Jones",
  "isbn": "123-456-789-111-x"
},
{
  "id_book": 2,
  "name": "Statistics",
  "author": "Lisa Smith",
  "isbn": "222-333-444-555-y"
},
{
  "id_book": 3,
  "name": "Scouting for Boys",
  "author": "Robert Baden-Powell",
  "isbn": "221-336-125-587"
},
{
  "id_book": 4,
  "name": "Komppanian taisteluohje",
  "author": "Maavoimien Esikunta",
  "isbn": "111-222-333-444"
}
```

Book POST toiminto:

Syötetään tauluun uusi tietue.

Koska taulussa on id ja se on auto_increment, jätämme id syötön pois.

The screenshot shows the VS Code interface with a REST client extension. A POST request is configured to `localhost:3000/book` with the following form data:

Key	Value
name	Tien päällä
author	Matti Esko
isbn	987-456-321-258

The response is a 200 OK status with the following JSON body:

```
{  "fieldCount": 0,  "affectedRows": 1,  "insertId": 5,  "info": "",  "serverStatus": 2,  "warningStatus": 0,  "changedRows": 0}
```

A tooltip for the 200 OK status explains: "Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action."

Book PUT toiminto:

Haluan muuttaa tietuetta id:ssä 5, laitan kenttään localhost:3000/book/5. (/5 meinaa tietueen ID:tä)

Tässä halusin muuttaa ainoastaan "author" kohtaa, joten kaikki muut tiedot pitää olla oikein, jos jätän muut tiedot NULL:iksi niin tiedot name, isbn myös muuttuvat NULL:iksi.

The screenshot shows a REST client interface with a PUT request to `localhost:3000/book/5`. The request body is configured as `x-www-form-urlencoded` with the following data:

Key	Value
name	Tien päällä
author	Janne Tulkki
isbn	987-456-321-258

The response is a JSON object with the following fields:

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "info": "Rows matched: 1  Changed: 1  Warnings: 0",
6   "serverStatus": 2,
7   "warningStatus": 0,
8   "changedRows": 1
9 }
```

The status bar indicates a successful response: `Status: 200 OK Time: 127 ms Size: 383 B`.

Book DELETE toiminto:

Tässä haluan poistaa koko tietueen ID:stä 5.

The screenshot shows a REST client interface with the following details:

- URL:** `localhost:3000/book/5`
- Method:** `DELETE`
- Body:** Empty (selected as `x-www-form-urlencoded`)
- Response:** `200 OK`, Time: `129 ms`, Size: `343 B`
- Response Body (JSON):**

```
{  "fieldCount": 0,  "affectedRows": 1,  "insertId": 0,  "info": "",  "serverStatus": 2,  "warningStatus": 0,  "changedRows": 0}
```