

17: Tietokantayhteys ja CRUD Nodella ja Expressillä

Database tiedosto:

```
JS database.js > ...
1  const mysql = require('mysql2');
2  const connection = mysql.createPool({
3    host: 'localhost',
4    user: 'testi',
5    password: 'xxxx',
6    database: 'netdb'
7  });
8  module.exports = connection;
```

Book.js tiedosto:

```
routes > JS book.js > router.get('/:id') callback > book.getById callback
1  const express = require('express');
2  const router = express.Router();
3  const book = require('../models/book_model');
4
5  router.get('/',
6    function (request, response) {
7      book.getAll(function (err, dbResult) {
8        if (err) {
9          response.json(err);
10         } else {
11           console.log(dbResult);
12           response.json(dbResult);
13         }
14       });
15     });
16
17  router.get('/:id',
18    function (request, response) {
19      book.getById(request.params.id, function (err, dbResult) {
20        if (err) {
21          response.json(err);
22        } else {
23          response.json(dbResult);
24        }
25      });
26    });
27
28
29  router.post('/',
30    function(request, response) {
31      book.add(request.body, function(err, dbResult) {
32        if (err) {
33          response.json(err);
34        } else {
35          response.json(dbResult);
36        }
37      });
38    });
39
40
41  router.delete('/:id',
42    function(request, response) {
43      book.delete(request.params.id, function(err, dbResult) {
44        if (err) {
45          response.json(err);
46        } else {
47          response.json(dbResult);
48        }
49      });
50    });
51
52
53  router.put('/:id',
54    function(request, response) {
55      book.update(request.params.id, request.body, function(err, dbResult) {
56        if (err) {
57          response.json(err);
58        } else {
59          response.json(dbResult);
60        }
61      });
62    });
63
64  module.exports = router;
```

Book_model.js tiedosto:

```
models > JS book_model.js > ...
1  const db = require('../database');
2
3  const book = {
4    getAll: function(callback) {
5      return db.query('select * from book', callback);
6    },
7    getById: function(id, callback) {
8      return db.query('select * from book where id_book=?', [id], 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;
```

App.js tiedosto:

```
JS app.js > ...
1  var express = require('express');
2  var path = require('path');
3  var cookieParser = require('cookie-parser');
4  var logger = require('morgan');
5
6
7  var bookRouter = require('./routes/book');
8
9  var app = express();
10
11  app.use(logger('dev'));
12  app.use(express.json());
13  app.use(express.urlencoded({ extended: false }));
14  app.use(cookieParser());
15  app.use(express.static(path.join(__dirname, 'public')));
16
17
18  app.use('/book', bookRouter);
19
20
21  module.exports = app;
22
```

GET-metodi:

GET

localhost:3000/book/

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Query Params

	Key
	Key

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

JSON ▾

↻

```
1  [
2    {
3      "id_book": 1,
4      "name": "PHP Basic",
5      "author": "Bob Jones",
6      "isbn": "123-456-789-111-x"
7    },
8    {
9      "id_book": 2,
10     "name": "Statistics",
11     "author": "Lisa Smith",
12     "isbn": "222-333-444-555-y"
13   }
14 ]
```

POST-metodi:

POST localhost:3000/book/

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key	Value
<input checked="" type="checkbox"/>	name	Tuntematon Kimi Räikkönen
<input checked="" type="checkbox"/>	author	Kari Hotakainen
<input checked="" type="checkbox"/>	isbn	111-222-333-444
	Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON

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

```
14  {
15    "id_book": 3,
16    "name": "Tuntematon Kimi Räikkönen",
17    "author": "Kari Hotakainen",
18    "isbn": "111-222-333-444"
19  }
20 }
```

PUT-metodi:

The screenshot shows a REST client interface with a PUT request to `localhost:3000/book/3`. The request body is set to `x-www-form-urlencoded`. The form data includes:

Key	Value
name	Tuntematon Kimi Räikkönen
author	Kari Hotakainen
isbn	123-456-789-012

The response body is shown in JSON format:

```
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 }
```

Isbn muutettu eri arvoon:

The screenshot shows the response body in JSON format, indicating the ISBN has been updated:

```
1 [
2   {
3     "id_book": 3,
4     "name": "Tuntematon Kimi Räikkönen",
5     "author": "Kari Hotakainen",
6     "isbn": "123-456-789-012"
7   }
8 ]
```

DELETE-metodi: (huom! kirja oli aluksi book_id3, kämmin vuoksi tein uuden ja se meni nyt id4)

DELETE localhost:3000/book/4

Params Authorization Headers (9) Body ● Pre-request Script Tests

Query Params

Key
Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON ▾

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

Book_id4 poistettu:

GET localhost:3000/book/4

Params Authorization Headers (9) Body ● Pre-re

Query Params

Key
Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON ▾

```
1 []
```

Borrower taulun (CRUD)

Borrower_model.js tiedosto:

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

Borrower.js tiedosto:

```
routes > JS borrower.js > router.post('/') callback > borrower.add() callback
1  const express = require('express');
2  const router = express.Router();
3  const borrower = require('../models/borrower_model');
4
5  router.get('/',
6    function (request, response) {
7      borrower.getAll(function (err, dbResult) {
8        if (err) {
9          response.json(err);
10         } else {
11           console.log(dbResult);
12           response.json(dbResult);
13         }
14       });
15     });
16
17  router.get('/:id',
18    function (request, response) {
19      borrower.getById(request.params.id, function (err, dbResult) {
20        if (err) {
21          response.json(err);
22        } else {
23          response.json(dbResult);
24        }
25      });
26    });
27
28
29  router.post('/',
30    function(request, response) {
31      borrower.add(request.body, function(err, dbResult) {
32        if (err) {
33          response.json(err);
34        } else {
35          response.json(dbResult);
36        }
37      });
38    });
39
40
41  router.delete('/:id',
42    function(request, response) {
43      borrower.delete(request.params.id, function(err, dbResult) {
44        if (err) {
45          response.json(err);
46        } else {
47          response.json(dbResult);
48        }
49      });
50    });
51
52
53  router.put('/:id',
54    function(request, response) {
55      borrower.update(request.params.id, request.body, function(err, dbResult) {
56        if (err) {
57          response.json(err);
58        } else {
59          response.json(dbResult);
60        }
61      });
62    });
63
64  module.exports = router;
65
```


App.js tiedosto:

```
JS app.js > ...
1  var express = require('express');
2  var path = require('path');
3  var cookieParser = require('cookie-parser');
4  var logger = require('morgan');
5
6
7  var bookRouter = require('./routes/book');
8  var borrowerRouter = require('./routes/borrower');
9
10
11  var app = express();
12
13  app.use(logger('dev'));
14  app.use(express.json());
15  app.use(express.urlencoded({ extended: false }));
16  app.use(cookieParser());
17  app.use(express.static(path.join(__dirname, 'public')));
18
19
20  app.use('/book', bookRouter);
21  app.use('/borrower', borrowerRouter);
22  |
23
24  module.exports = app;
25
```

GET-metodi:

The screenshot shows a web browser's developer tools interface. At the top, the method is 'GET' and the URL is 'localhost:3000/borrower/'. Below this, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', and 'Tests'. The 'Body' tab is selected, showing a JSON response. The response is a JSON array containing two objects, each representing a borrower with fields for 'id_borrower', 'fname', 'lname', and 'streetAddress'.

Key
Key

Query Params

Body

Pretty Raw Preview JSON

```
[
  {
    "id_borrower": 1,
    "fname": "Matti",
    "lname": "Meikäläinen",
    "streetAddress": "Kivitie 1"
  },
  {
    "id_borrower": 2,
    "fname": "Jorma",
    "lname": "Uotinen",
    "streetAddress": "Aapistie 12"
  }
]
```

POST-metodi:

POST localhost:3000/borrower/

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key	Value
<input checked="" type="checkbox"/>	fname	Aku
<input checked="" type="checkbox"/>	lname	Ankka
<input checked="" type="checkbox"/>	streetAddress	Ankkalinnantie 1
	Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON

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

```
14 {
15   "id_borrower": 4,
16   "fname": "Aku",
17   "lname": "Ankka",
18   "streetAddress": "Ankkalinnantie 1"
19 }
20 ]
```

PUT-metodi:

PUT localhost:3000/borrower/4

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key	Value
<input checked="" type="checkbox"/>	fname	Roope
<input checked="" type="checkbox"/>	lname	Ankka
<input checked="" type="checkbox"/>	streetAddress	Ankkalinnantie 1

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON

```
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 }
```

Muutetut kentät:

```
1 [
2   {
3     "id_borrower": 4,
4     "fname": "Roope",
5     "lname": "Ankka",
6     "streetAddress": "Ankkalinnantie 1"
7   }
8 ]
```

Delete-metodi:

DELETE localhost:3000/borrower/4

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key	Value
<input checked="" type="checkbox"/>	fname	Roope
<input checked="" type="checkbox"/>	lname	Ankka
<input checked="" type="checkbox"/>	streetAddress	Ankkalinnantie 1

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON

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

borrower_id4 poistettu:

GET localhost:3000/borrower/4

Params Authorization Headers (9) **Body** Pre-request S

Query Params

	Key
	Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON

```
1 []
```