17: Tietokantayhteys ja CRUD Nodella ja Expressillä

Database tiedosto:

database.js > ...

```
const mysql = require('mysql2');
const connection = mysql.createPool({
  host: 'localhost',
  user: 'testi',
  password: 'xxxx',
  database: 'netdb'
});
module.exports = connection;
```

Book.js tiedosto:

routes > book.js > router.get('/:id') callback > book.getById() callback

```
const express = require('express');
const router = express.Router();
const book = require('../models/book_model');

router.get('/',
    function (request, response) {
        book.getAll(function (err, dbResult) {
            if (err) {
                response.json(err);
            } else {
                console.log(dbResult);
                response.json(dbResult);
            }
        })
    });

router.get('/:id',
    function (request, response) {
        book.getById(request.params.id, function (err, dbResult) {
            if (err) {
                response.json(err);
            } else {
                response.json(dbResult);
            }
        })
    });


router.post('/',
function(request, response) {
  book.add(request.body, function(err, dbResult) {
    if (err) {
      response.json(err);
    } else {
      response.json(dbResult);
    }
  });
});


router.delete('/:id',
function(request, response) {
  book.delete(request.params.id, function(err, dbResult) {
    if (err) {
      response.json(err);
    } else {
      response.json(dbResult);
    }
  });
});


router.put('/:id',
function(request, response) {
  book.update(request.params.id, request.body, function(err, dbResult) {
    if (err) {
      response.json(err);
    } else {
      response.json(dbResult);
    }
  });
});

module.exports = router;
```

Book_model.js tiedosto:

models > JS book_model.js > ...

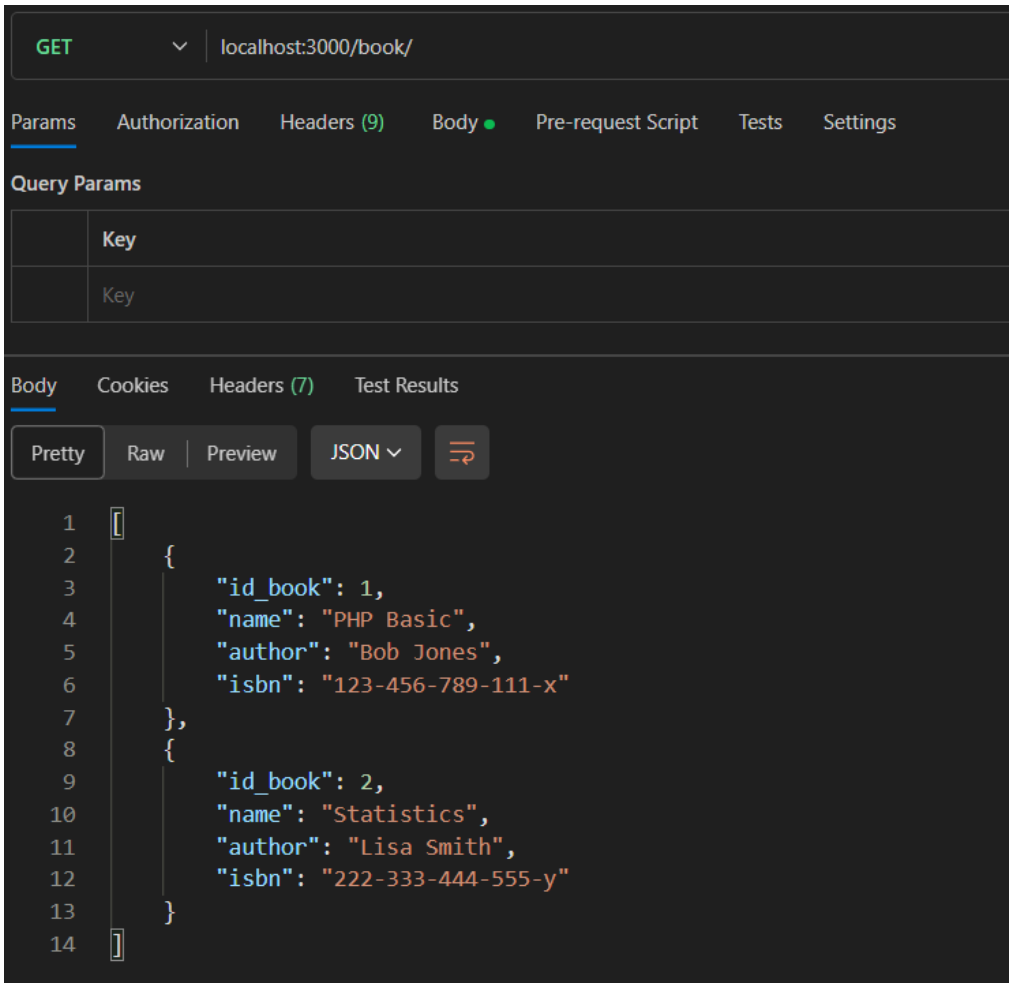
```
const db = require('../database');

const book = {
  getAll: function(callback) {
    return db.query('select * from book', callback);
  },
  getById: function(id, callback) {
    return db.query('select * from book where id_book=?', [id], callback);
  },
  add: function(book, callback) {
    return db.query(
      'insert into book (name,author,isbn) values(?,?,?)',
      [book.name, book.author, book.isbn],
      callback
    );
  },
  delete: function(id, callback) {
    return db.query('delete from book where id_book=?', [id], callback);
  },
  update: function(id, book, callback) {
    return db.query(
      'update book set name=?,author=?, isbn=? where id_book=?',
      [book.name, book.author, book.isbn, id],
      callback
    );
  }
};
module.exports = book;
```

App.js tiedosto:

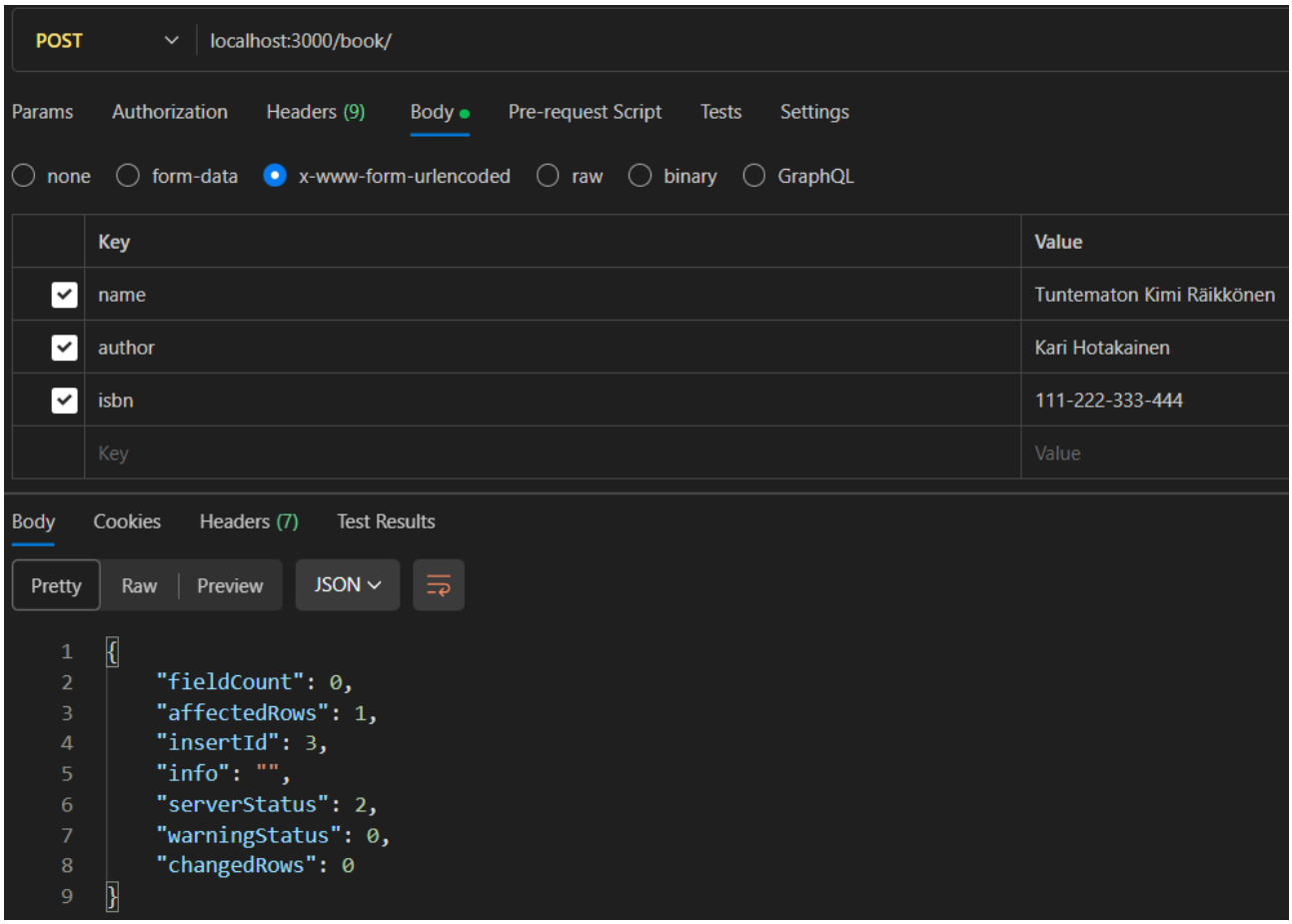
JS app.js > ...

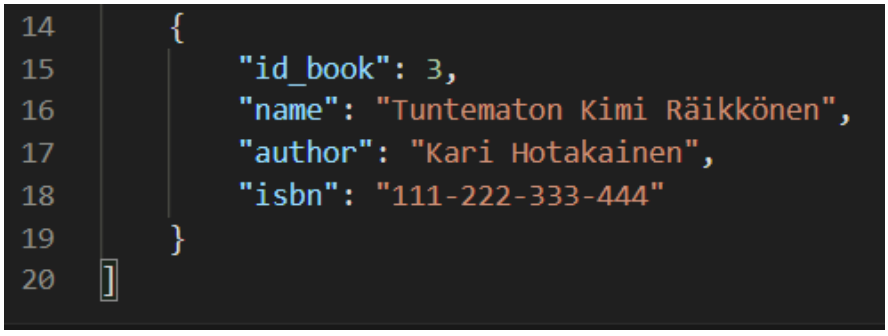
```
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');


var bookRouter = require('./routes/book');

var app = express();

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));


app.use('/book', bookRouter);


module.exports = app;

```

GET-metodi:

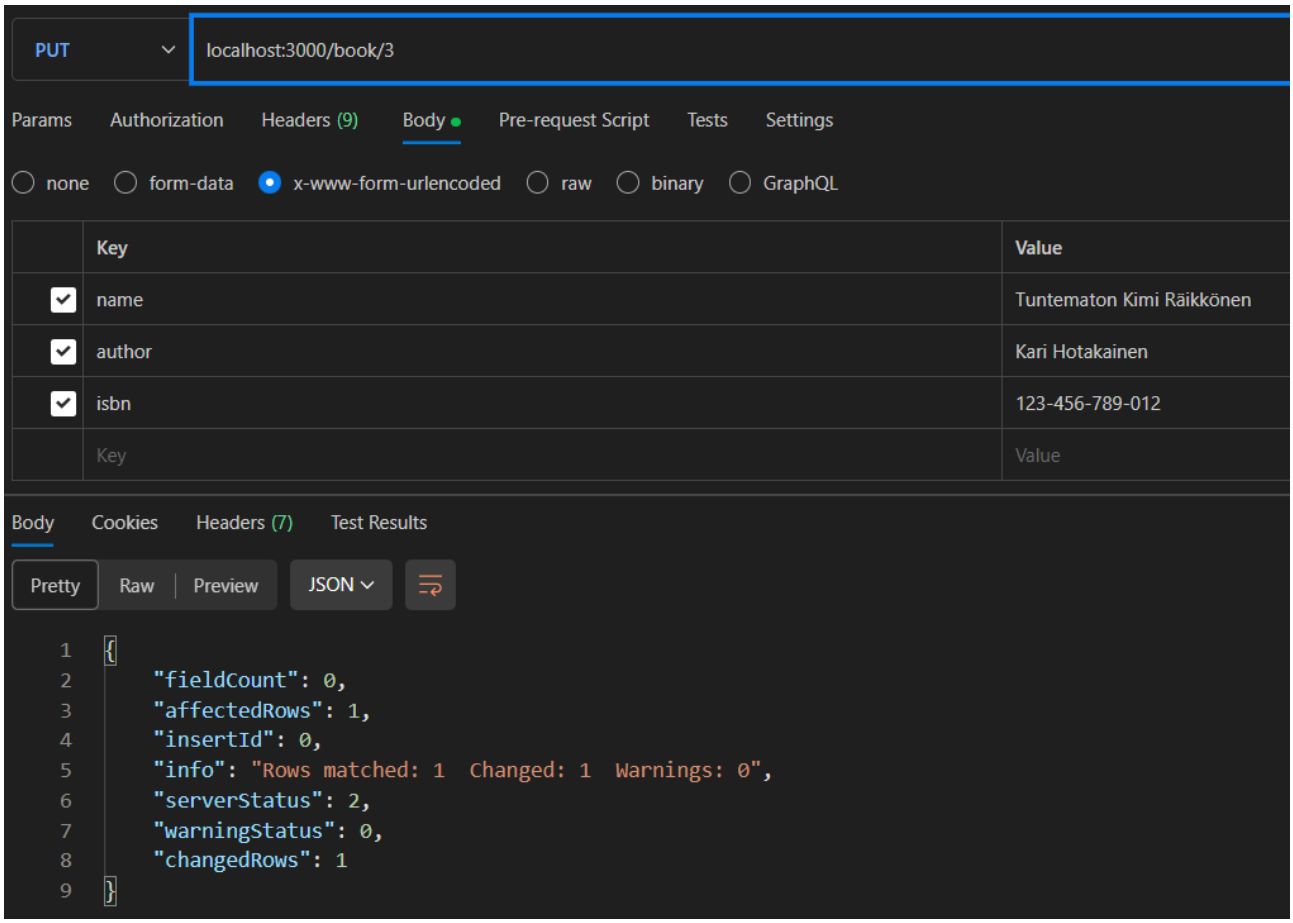

GET
localhost:3000/book/
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Query Params
Key
Key
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
JSON
[
{
"id_book": 1,
"name": "PHP Basic",
"author": "Bob Jones",
"isbn": "123-456-789-111-x"
},
{
"id_book": 2,
"name": "Statistics",
"author": "Lisa Smith",
"isbn": "222-333-444-555-y"
}
]

POST-metodi:

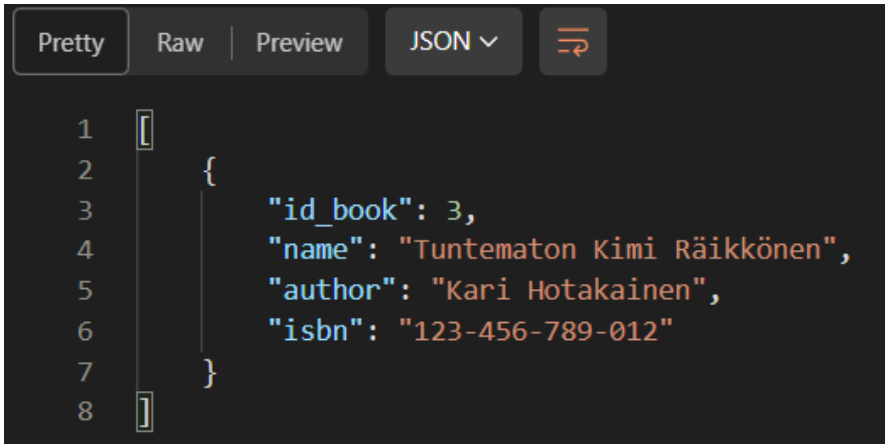

POST
localhost:3000/book/
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Key
Value
name
Tuntematon Kimi Räikkönen
author
Kari Hotakainen
isbn
111-222-333-444
Key
Value
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
JSON
{
"fieldCount": 0,
"affectedRows": 1,
"insertId": 3,
"info": "",
"serverStatus": 2,
"warningStatus": 0,
"changedRows": 0
}


{
"id_book": 3,
"name": "Tuntematon Kimi Räikkönen",
"author": "Kari Hotakainen",
"isbn": "111-222-333-444"
}
]

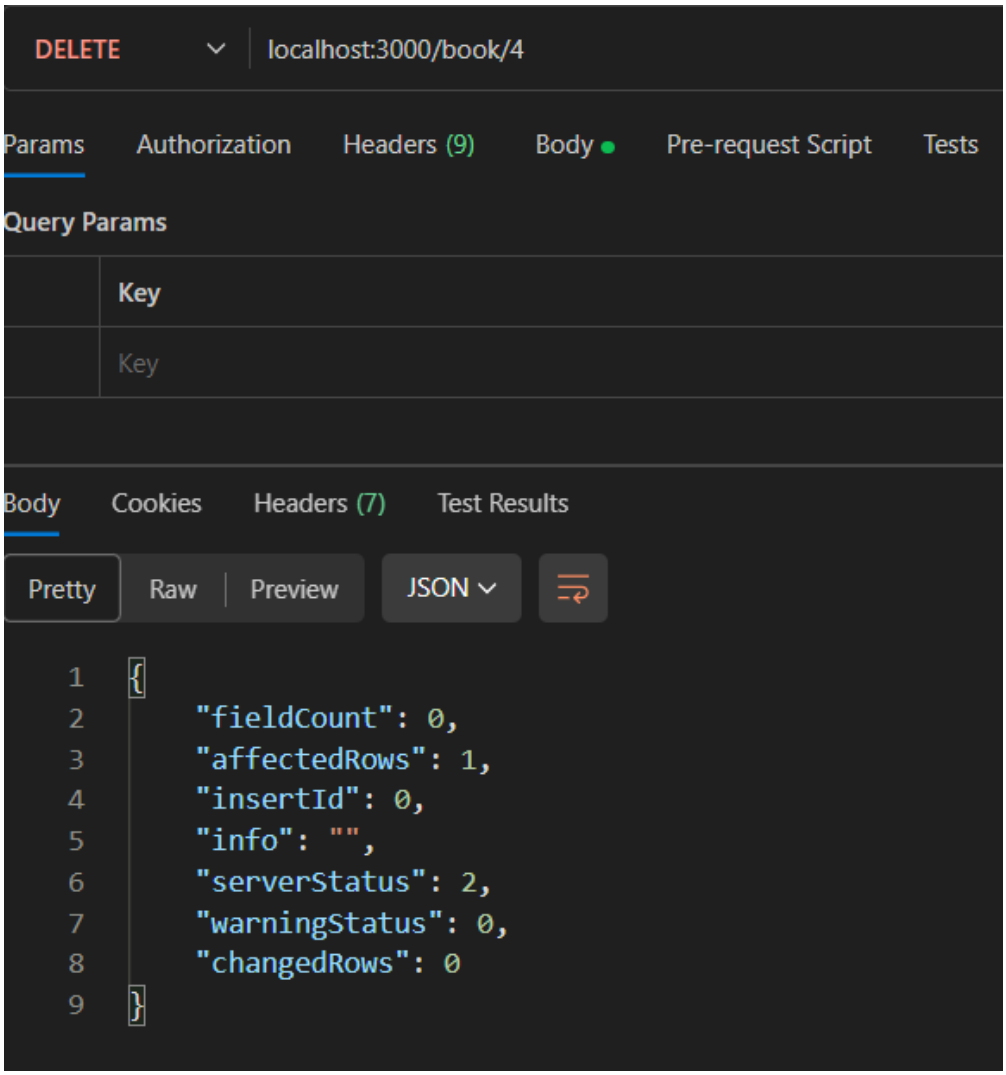
PUT-metodi:



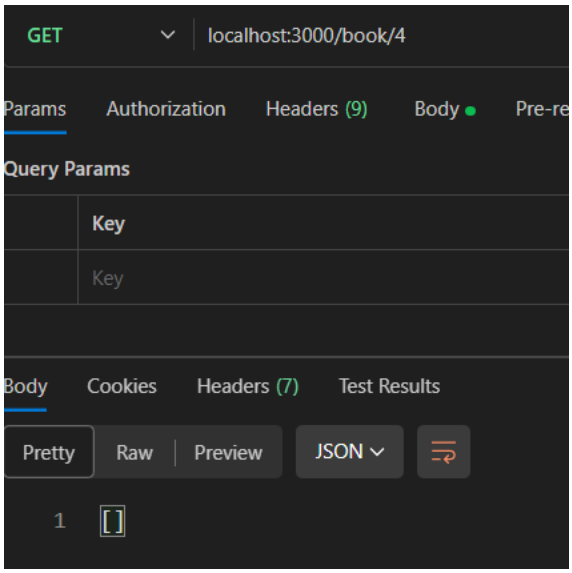
Isbn muutettu eri arvoon:



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



Book_id4 poistettu:



Borrower taulun (CRUD)

Borrower_model.js tiedosto:

models > JS borrower_model.js > borrower > update

```
const db = require('../database');

const borrower = {
  getAll: function(callback) {
    return db.query('select * from borrower', callback);
  },
  getById: function(id, callback) {
    return db.query('select * from borrower where id_borrower=?', [id], callback);
  },
  add: function(borrower, callback) {
    return db.query(
      'insert into borrower (fname,lname,streetAddress) values(?,?,?)',
      [borrower.fname, borrower.lname, borrower.streetAddress],
      callback
    );
  },
  delete: function(id, callback) {
    return db.query('delete from borrower where id_borrower=?', [id], callback);
  },
  update: function(id, borrower, callback) {
    return db.query(
      'update borrower set fname=?,lname=?, streetAddress=? where id_borrower=?',
      [borrower.fname, borrower.lname, borrower.streetAddress, id],
      callback
    );
  }
};
module.exports = borrower;
```

Borrower.js tiedosto:

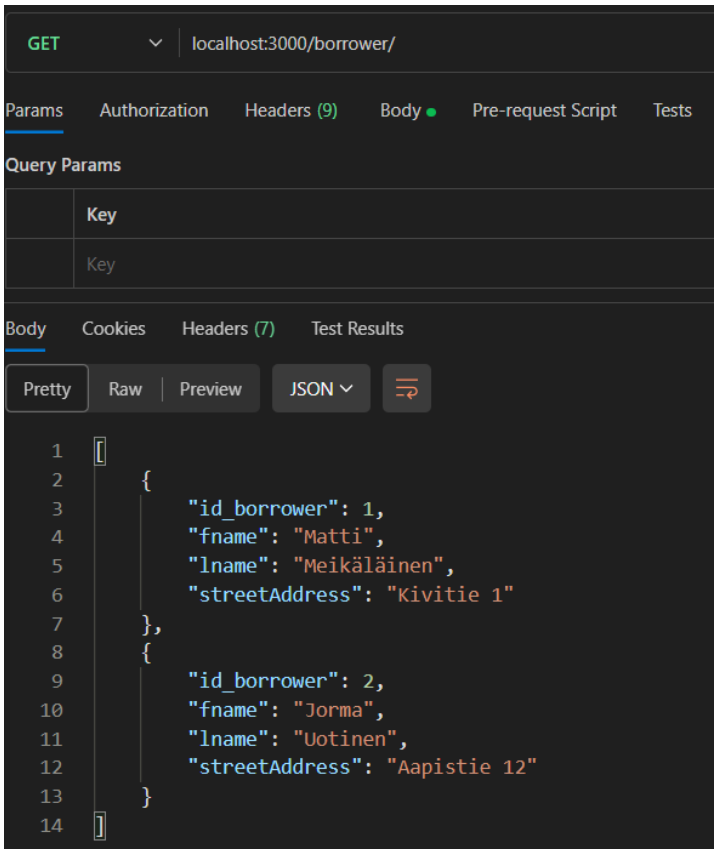
routes > JS borrower.js > router.post('/') callback > borrower.add() callback

```
const express = require('express');
const router = express.Router();
const borrower = require('../models/borrower_model');

router.get('/',
    function (request, response) {
        borrower.getAll(function (err, dbResult) {
            if (err) {
                response.json(err);
            } else {
                console.log(dbResult);
                response.json(dbResult);
            }
        })
    });

router.get('/:id',
    function (request, response) {
        borrower.getById(request.params.id, function (err, dbResult) {
            if (err) {
                response.json(err);
            } else {
                response.json(dbResult);
            }
        })
    });


router.post('/',
function(request, response) {
  borrower.add(request.body, function(err, dbResult) {
    if (err) {
      response.json(err);
    } else {
      response.json(dbResult);
    }
  });
});


router.delete('/:id',
function(request, response) {
  borrower.delete(request.params.id, function(err, dbResult) {
    if (err) {
      response.json(err);
    } else {
      response.json(dbResult);
    }
  });
});


router.put('/:id',
function(request, response) {
  borrower.update(request.params.id, request.body, function(err, dbResult) {
    if (err) {
      response.json(err);
    } else {
      response.json(dbResult);
    }
  });
});

module.exports = router;

```

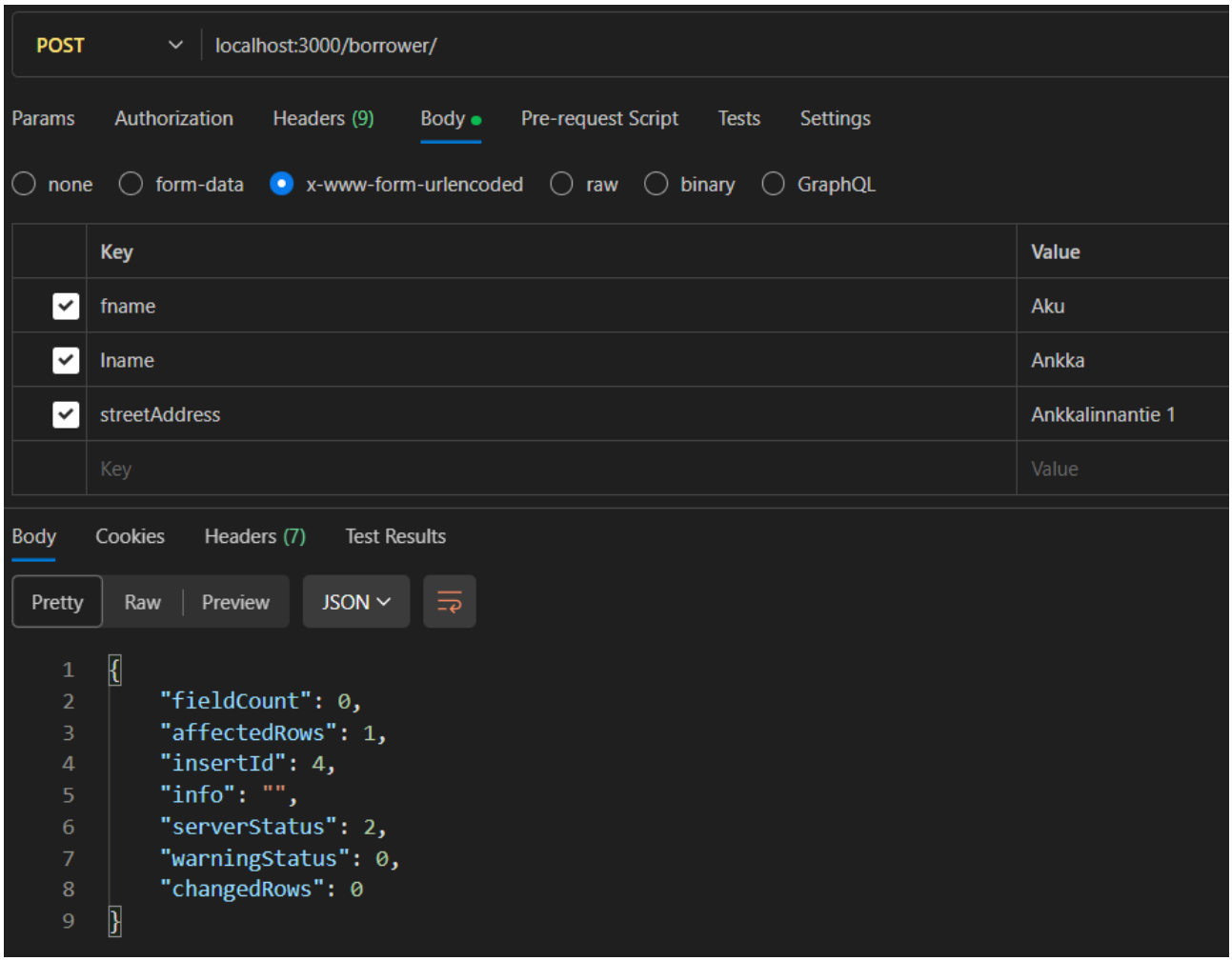
App.js tiedosto:

```
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:

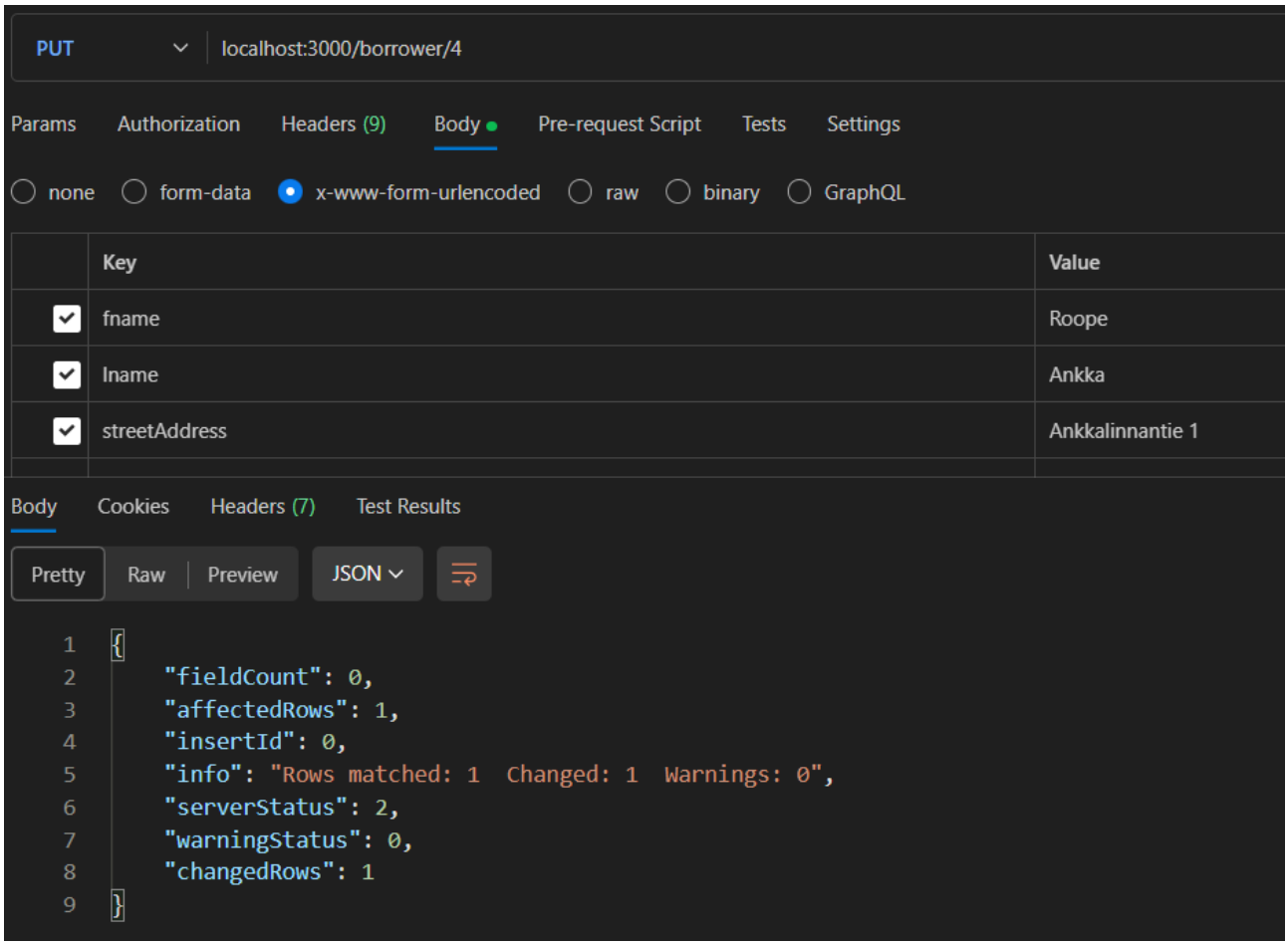


POST-metodi:


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
fname
Aku
lname
Ankka
streetAddress
Ankkalinnantie 1
Key
Value
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
JSON
{
"fieldCount": 0,
"affectedRows": 1,
"insertId": 4,
"info": "",
"serverStatus": 2,
"warningStatus": 0,
"changedRows": 0
}

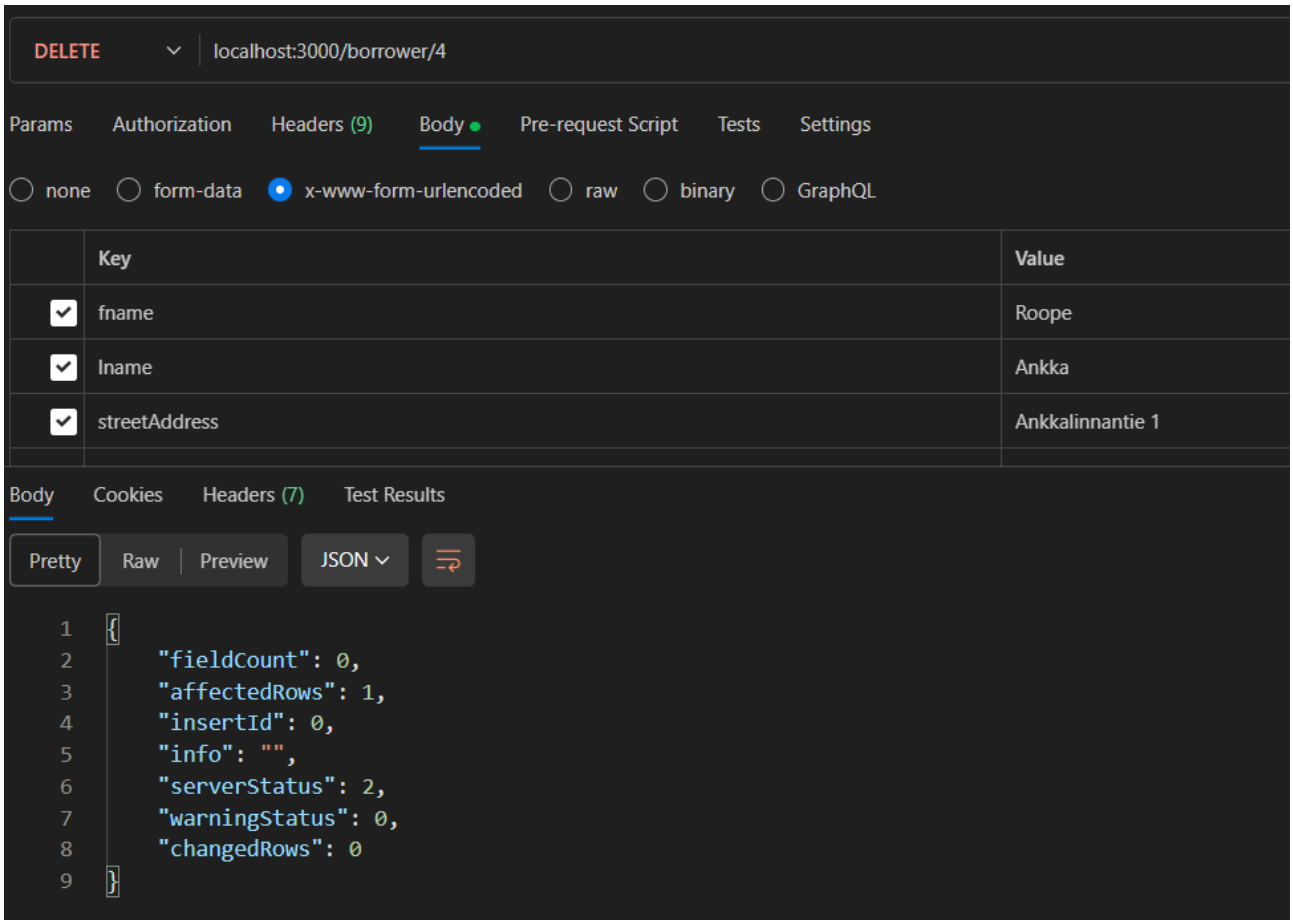

{
"id_borrower": 4,
"fname": "Aku",
"lname": "Ankka",
"streetAddress": "Ankkalinnantie 1"
}
]

PUT-metodi:



Muutetetut kentät:

Delete-metodi:



borrower_id4 poistettu: