**Internet-ohjelmointi ja tietokannat Osa 2: Tietokannan tekeminen**

Tehdään uusi branch nimeltä book

Tehdään tietokanta esimerkin mukaan, tämän sinä osaat itsekin :D

Connection string esimerkin mukaan

const mysql = require('mysql');

const myConnectionString = "mysql://netuser:netpass@localhost:3306/netdb";

const connection = mysql.createPool(myConnectionString);

module.exports = connection;

Seuraavaksi kontrolleri Routes-kansioon: Routes/book.js

const express = require('express');

const router = express.Router();

const book = require('../models/book\_model');

router.get('/:id?',

function(request, response) {

if (request.params.id) {

book.getById(request.params.id, function(err, dbResult) {

if (err) {

response.json(err);

} else {

response.json(dbResult);

}

});

} else {

book.getAll(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(request.body);

}

});

});

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;

Edellä on viittaus models-kansioon. Express generator ei ole luonut sitä, pitää itse tehdä.

../models/book\_model

book\_model on siis javascript-tiedosto. Tarkenninta ei tuohon kuitenkaan tavallisesti merkitä.

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

const book = {

getById: function(id, callback) {

return db.query('select \* from book where id\_book=?', [id], callback);

},

getAll: function(callback) {

return db.query('select \* from book', 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;

index.js ja users.js joutaa tässä vaiheessa pois, niitä ei tarvita.

**Muokataan app.js**

indexRouter ja usersRouter pois. Samoin

app.use('/', indexRouter);

app.use('/users', usersRouter);

Niiden tilalle

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

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

Javascriptin muuttuja var on globaali ja let on paikallinen, const on vakio. Se on sama kuin var. Varissa on se vika, että sitä voi vahingossa muuttaa mistä vaan.

Lisätään

const helmet = require('helmet');

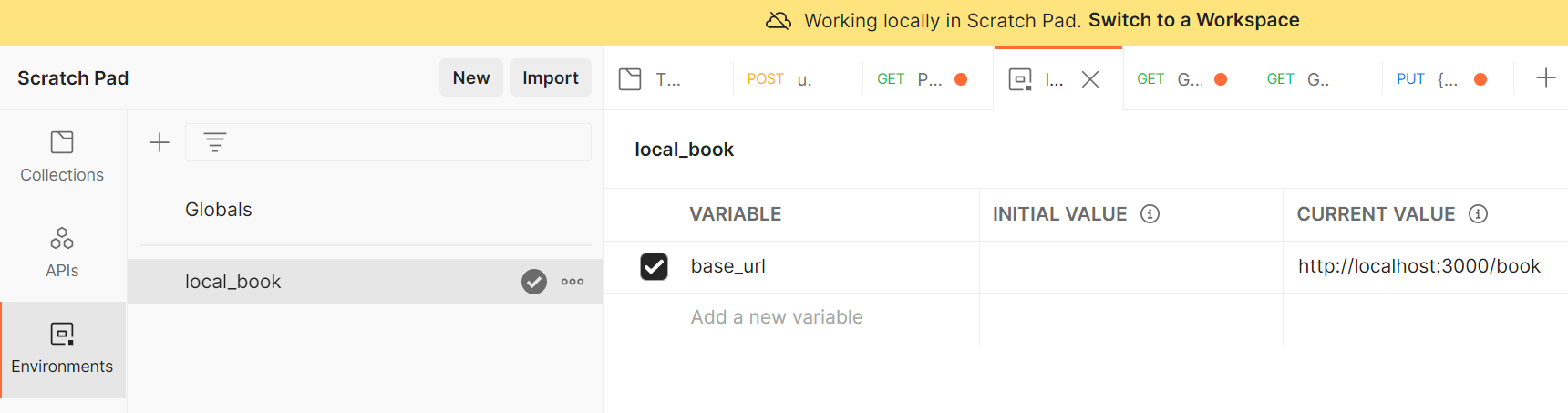
const cors = require('cors');

app.use(helmet());

app.use(cors());

Postmanilla pitää testata kaikki neljä end pointtia.

Tehdään ensin ympäristö Environment, vaikka nimellä **local\_book**



Nyt jos vaihtaa tietokannan muulle palvelimelle (esim Heroku), tarttee jatkossa tehdä vaan eri niminen Environment. Ei tartte kaikkia urleja kirjoittaa uudestaan !

A screenshot of a computer

Description automatically generatedPOST ja PUT. Muista Body ja urlencoded!

**Muista: GET/POST/PUT/DELETE {{base\_url}} jne**