Apiprojekti

Emmi Luokkakallio

Yleiskatsaus

- Tämä projekti tarjoaa REST API:n, joka hallinnoi siivousfirman tietoja SQLite-tietokannassa. API mahdollistaa siivoojien tietojen hakemisen ja lisäämisen sekä poistamisen ja päivittämisen.

Aikataulu

- Projekti on toteutettu lyhyessä ajassa, intensiivisellä työskentelyllä
  - 24.3.2025 4h intensiivijakso
  - 25.3.2025 6h intensiivijakso
  - 26.3.2025 6h intensiivijakso
  - 27.3.2025 2h viimeistely ja Projektin esittely

Teknologiat

- Node.js
- Express.js
- SQLite3
- Dotenv
- Body-parser

Haasteet

- Alkuhaasteena oli ymmärtää mitä tehtävässä halutaan, mitkä on vaatimukset ja kuinka lähteä sitä toteuttamaan.
  - → Keskustelu, tekoälyn hyödyntäminen
- JavaScriptilla koodaaminen ei ole vahvuus, joten se työllisti ja haastoi
  - → Dokumentaatioiden ja osaavamman tuki
- Curl komennot Windowsilla
  - → siirryttiin käyttämään Postman

CRUD: READ

- GET /cleaners palauttaa siivoojien tiedot.
- Pyynnön voi rajata hakuehtojen avulla.
- Jos hakuehtoja ei ole, palautetaan kaikkien siivoojien tiedot.
- Jos hakuehtoja on, palautetaan vain niiden siivoojien tiedot, jotka täyttävät hakuehdot.
- Jos hakuehtoja ei ole, palautetaan statuskoodi 200.
- Jos hakuehtoja on, palautetaan statuskoodi 200.
- Jos hakuehtoja ei ole, mutta tietokantaan ei saada yhteyttä, palautetaan statuskoodi 500.
- Jos hakuehtoja on, mutta tietokantaan ei saada yhteyttä, palautetaan statuskoodi 500.
- Jos hakuehtoja ei ole, mutta siivoojia ei löydy, palautetaan statuskoodi 404.
- Jos hakuehtoja on, mutta siivoojia ei löydy, palautetaan statuskoodi 404.

```
app.get('/cleaners', (req, res) => {
    const { fname, lname, salgrade, hire_date } = req.query;
    const allowedParams = ['fname', 'lname', 'salgrade', 'hire_date'];

    // Tarkista, ettei mukana ole tuntemattomia parametreja
    const unknownParams = Object.keys(req.query).filter(param => !allowedParams.includes(param));
    if (unknownParams.length > 0) {
        return res.status(400).json({ error: `Unknown parameter(s): ${unknownParams.join(', ')}` });
    }

    let sql = `SELECT * FROM cleaner WHERE 1=1`;
    let params = [];

    if (fname) {
        sql += ` AND LOWER(fname) = LOWER(?)`;
        params.push(fname);
    }
    if (lname) {
        sql += ` AND LOWER(lname) = LOWER(?)`;
        params.push(lname);
    }
    if (salgrade) {
        sql += ` AND salgrade = ?`;
        params.push(salgrade);
    }
    if (hire_date) {
        sql += ` AND hire_date = ?`;
        params.push(hire_date);
    }

    db.all(sql, params, (err, rows) => {
        if (err) {
            res.status(500).json({ error: err.message });
            return;
        }

        if (rows.length === 0) {
            res.status(404).json({ message: "No cleaners found with given search criteria" });
            return;
        }

        res.json(rows);
    });
```

CRUD: CREATE

- POST /cleaners lisää uuden siivoojan tietokantaan.
- Pyynnön rungossa tulee olla seuraavat kentät: fname, lname, salgrade ja hire_date.
- Jos jokin kenttä puuttuu, palautetaan statuskoodi 400.
- Jos kaikki onnistuu, palautetaan lisätyn siivoojan tiedot JSON-muodossa ja statuskoodi 201.
- Jos tietokantaan ei saada yhteyttä, palautetaan statuskoodi 500.
- Jos vääriä parametreja, palautetaan statuskoodi 400

```
app.post('/cleaners', (req, res) => {
    const { fname, lname, salgrade, hire_date } = req.body;
    const allowedParams = ['fname', 'lname', 'salgrade', 'hire_date'];

    // Tarkista tuntemattomat parametrit
    const unknownParams = Object.keys(req.body).filter(param => !allowedParams.includes(param));
    if (unknownParams.length > 0) {
        return res.status(400).json({ error: `Unknown parameter(s): ${unknownParams.join(', ')}` });
    }

    // Tarkista, ettei mitään pakollista kenttää puutu
    if (!fname || !lname || !salgrade || !hire_date) {
        return res.status(400).json({ message: 'Missing required fields' });
    }

    const sql = `INSERT INTO cleaner (fname, lname, salgrade, hire_date) VALUES (?, ?, ?, ?)`;
    const params = [fname, lname, salgrade, hire_date];

    db.run(sql, params, function (err) {
        if (err) {
            return res.status(500).json({ error: err.message });
        }

        res.status(201).json({
            id: this.lastID,
            fname,
            lname,
            salgrade,
            hire_date
        });
    });
});
```

CRUD: UPDATE

- PUT /cleaners/:id päivittää siivoojan palkkaluokan.
- Pyynnön rungossa tulee olla kenttä salgrade.
- Jos salgrade puuttuu, palautetaan statuskoodi 400.
- Jos siivoojaa ei löydy, palautetaan statuskoodi 404.
- Jos kaikki onnistuu, palautetaan päivitetyn siivoojan tiedot JSON-muodossa ja statuskoodi 200.
- Jos tietokantaan ei saada yhteyttä, palautetaan statuskoodi 500.

```
app.put('/cleaners/:id', (req, res) => {
    const id = req.params.id;
    const {salgrade} = req.body;

    if (!salgrade) {
        return res.status(STATUS_CODES.BAD_REQUEST).json({ message: 'Missing required fields' });
    }

    const sql = `UPDATE cleaner
                 SET salgrade = ?
                 WHERE id = ?`;
    const params = [salgrade, id];

    db.run(sql, params, function (err) {
        if (err) {
            return res.status(STATUS_CODES.INTERNAL_SERVER_ERROR).json({ error: err.message });
        }
        if (this.changes === 0) {
            return res.status(STATUS_CODES.NOT_FOUND).json({ message: 'Cleaner not found' });
        }

        res.status(STATUS_CODES.OK).json({
            'update': 'yes',
        'message': 'new salarydrade updated'
        });
    });
});
```

CRUD: DELETE

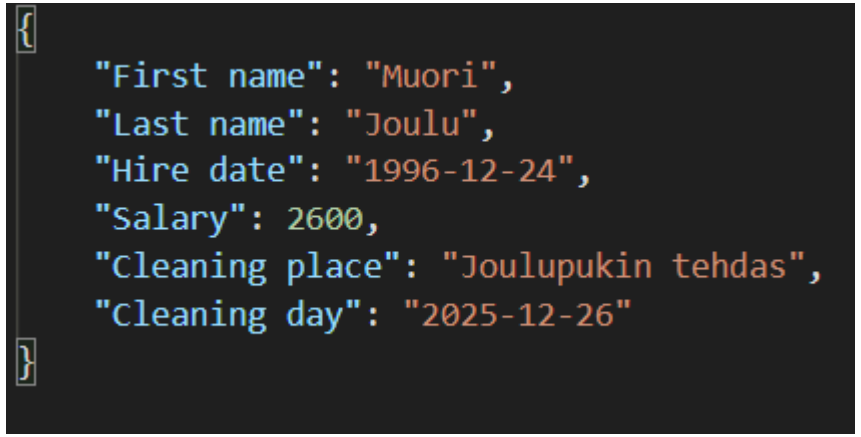
- delete/location/:name poistaa tietokannasta paikan nimen perusteella.
- Jos nimeä ei ole annettu, palautetaan statuskoodi 400
- Jos kaikki onnistuu, palautetaan statuskoodi 200.
- Muussa tapauksessa palautetaan statuskoodi 500.
- Jos paikkaa ei löydy, palautetaan statuskoodi 404.

```
app.delete('/locations/:name', (req, res) => {
    const name = req.params.name;
    if (!name) {
        return res.status(STATUS_CODES.BAD_REQUEST).json({ message: 'Missing required fields' })
    }
    const sql = `DELETE FROM location
                 WHERE LOWER(name) = LOWER(?)`;
    const params = [name];

    db.run(sql, params, function (err) {
        if (err) {
            return res.status(STATUS_CODES.INTERNAL_SERVER_ERROR).json({ error: err.message });
        }

        if (this.changes === 0) {
            return res.status(STATUS_CODES.NOT_FOUND).json({ message: 'Location not found' });
        }

        res.status(STATUS_CODES.OK).json({
            'delete': 'yes',
            'message': 'location deleted'
        });
    }
    );
```

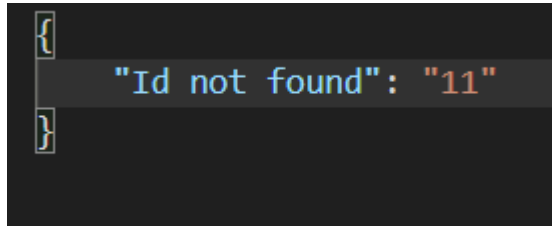
Testaus: ENDPOINT1 read by id

- Haetaan siivojan tiedot id mukaan
- Saadaan id mukaan tiedot siivoajan nimestä, palkkauspäivästä, palkka (salgrade-taulusta), siivouspäivästä (schedule-taulusta) ja siivouspaikasta (location-taulu) eli sql käsky sisältää relaatioita 3 muuhun tauluun (salgrade, location, schedule)

- curl --location 'localhost:8080/cleaners/2’

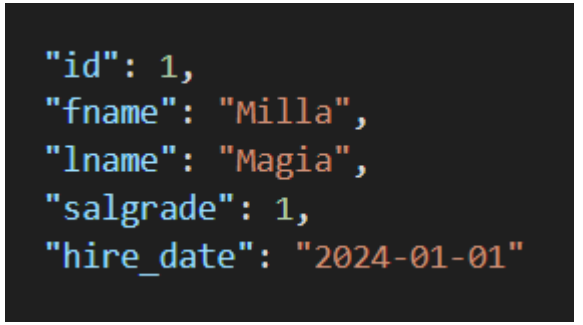


- Kun yritetään lukea sellaisen id tietoja, joita ei ole:
- curl --location 'localhost:8080/cleaners/11’



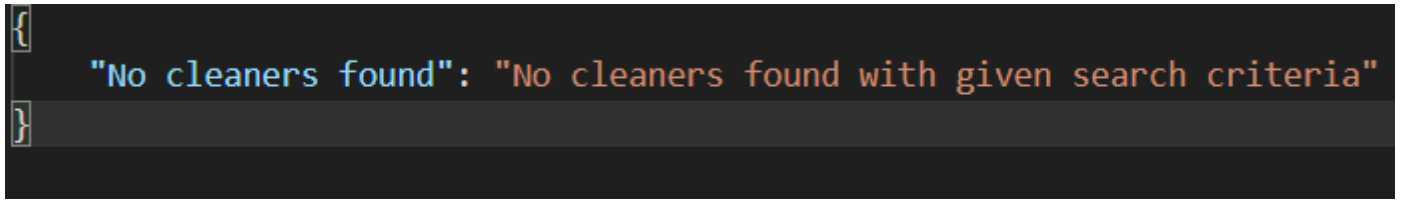
TESTAUS endpoint2: get ja useampi parametri

- Haetaan siivoajan tiedot eri parametrien mukaan joko yksin tai useammalla parametrilla kerrallaan
- ILMAN parametria → tulostaa kaikki siivoajat
  - curl --location 'localhost:8080/cleaners'
- YKSI parametri fname:
  - curl --location 'localhost:8080/cleaners?fname=Milla’
- 2 parametria fname, salgrade:
  - curl --location 'localhost:8080/cleaners?fname=Milla&salgrade=1’
- 4 parametria fname, lname, salgrademhiredate:
  - curl --location 'localhost:8080/cleaners?fname=Milla&salgrade=1&hire_date=2024-01-01&lname=magia’
- Huom. Ei case-sensitive, ja etsiminen onnistuu millä tahansa cleaner-taulun arvolla

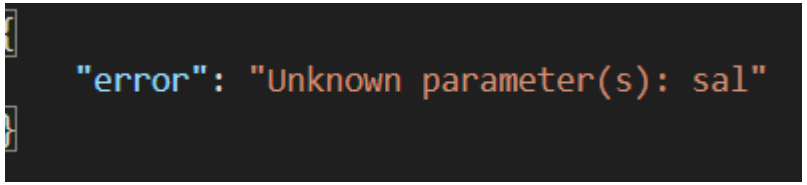


Endpoint2 virhesyötteet

- Virhesyöte sellaisella parametrin sisällöllä, joka ei löydä vastaavuutta taulusta
  - curl --location 'localhost:8080/cleaners?fname=Milla&salgrade=2&hire_date=2024-01-01&lname=magia’

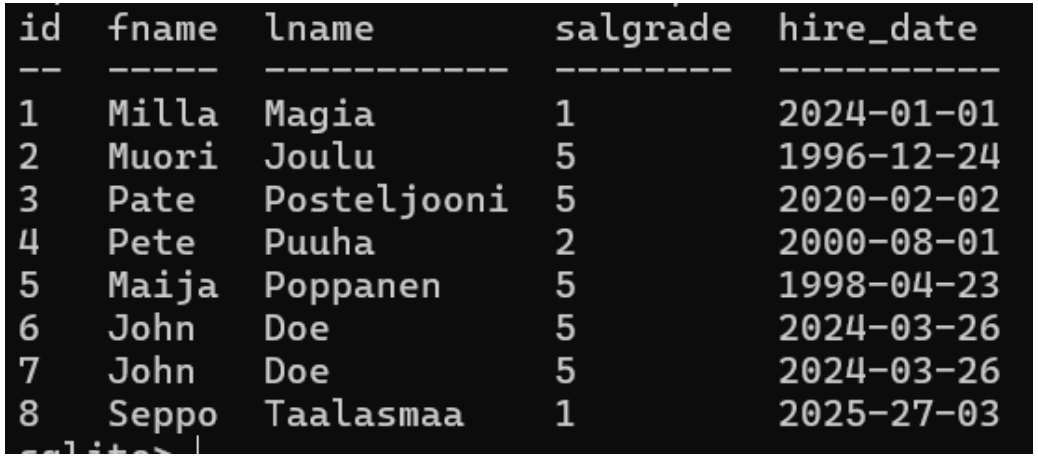


- Virhesyöte väärä parametri:
  - curl --location 'localhost:8080/cleaners?fname=Milla&sal=1000’



Testaus Endpoint3: create a cleaner

- Luodaan uusi siivoaja. Vaatii kaikki cleaner taulun parametrit
  - curl --location 'localhost:8080/cleaners?fname=seppo&lname=Taalasmaa&salgrade=1&hire_date=2025-03-27' \ --header 'Content-Type: application/json' \ --data '{"fname": "Seppo", "lname": "Taalasmaa", "salgrade": 1, "hire_date": "2025-27-03"}'



id	fname	lname	salgrade	hire_date
1	Milla	Magia	1	2024-01-01
2	Muori	Joulu	5	1996-12-24
3	Pate	Posteljooni	5	2020-02-02
4	Pete	Puuha	2	2000-08-01
5	Maija	Poppanen	5	1998-04-23
6	John	Doe	5	2024-03-26
7	John	Doe	5	2024-03-26
8	Seppo	Taalasmaa	1	2025-27-03

Endpoint3 virhesyötteet

- VAJAVAINEN CURL: curl --location --request POST 'localhost:8080/cleaners?fname=seppo&lname=Taalasmaa’

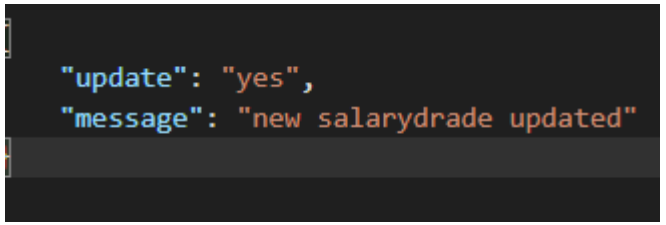
```
"message": "Missing required fields"
```

- VÄÄRIÄ PARAMETREJA
  - curl --location 'localhost:8080/cleaners?fname=seppo&lname=Taalasmaa&salgrade=1&hire_date=2025-03-27' \--header 'Content-Type: application/json' \--data '{"fname": "Maija", "lname": "Taalasmaa", "salgrade": 1, "hire_date": "2025-27-03", "sal": 1000}’

```
{
    "error": "Unknown parameter(s): sal"
}
```

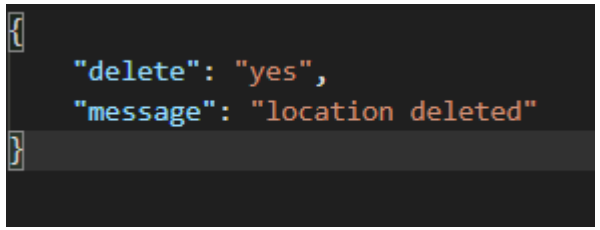
Testaus endpoint4 update

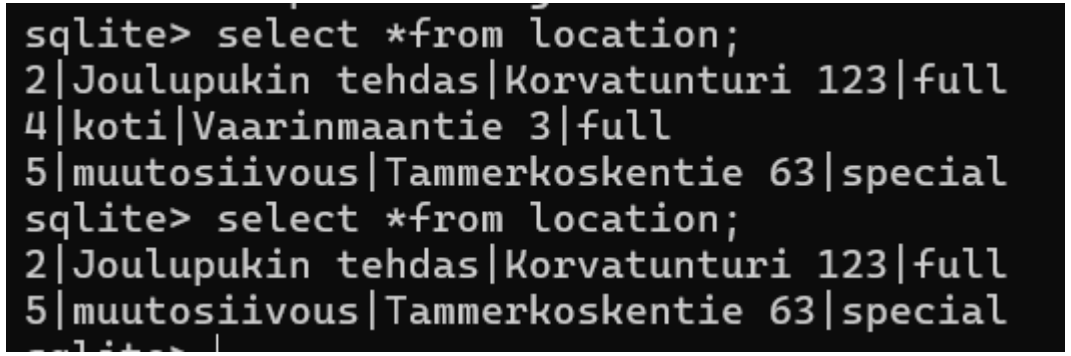
- Päivittää siivoojan palkan id mukaan, vaaditut/sallitut parametrit id ja salgrade
  - curl --location --request PUT 'localhost:8080/cleaners/2?salgrade=2' \--header 'Content-Type: application/json' \--data '{"salgrade": 2}’



Testaus: delete

- Poistaa lokaation nimen mukaan
  - curl --location --request DELETE 'localhost:8080/locations/koti’





Error handling

- Jokaisen endpointin aikana tehdään virheiden käsittelyä ja niihin vastataan sopivalla statuskoodilla, jotka on määritelty nimellä koodin alussa
- Tämän lisäksi app.use käsittelee tuntemattomat endpointit

```
app.use((req, res) => {
    res.type('json')
    let json = {
        'error': 'yes',
        'message': 'unknown endpoint'
    }
    res.status(STATUS_CODES.NOT_FOUND).json(json)
})
```

summary

- Projekti oli onnistunut
- Haasteista selvittiin ja koodi kehittyi tehtävän mukana esim. testausvaiheessa huomattiin vajavaisuutta koodissa