1 Alapok elsajátítása: Node.js és JavaScript

1.1 JavaScript alapok.

1.2 Node.js alapok

2 Express telepítése és első alkalmazás

2.1 Express telepítése

Kezdd egy új Node.js projekt létrehozásával, majd telepítsd az Express-t
npm install express
paranccsal.

2.2 Első szerver létrehozása:

Készíts egy alapvető Express szervert, ami egy egyszerű "Hello World" üzenetet ad vissza.

3 URL paraméterek és lekérdezési paraméterek kezelése

Cél: Megérteni, hogyan lehet URL paraméterekkel és lekérdezési paraméterekkel dolgozni.

Az Express-ben az adatok fogadásának főbb módjai a következők:

1. **Lekérdezési paraméterek** (req.query)

```
// GET /search?name=John&age=30

app.get('/search', (req, res) => {
    const name = req.query.name; // "John"
    const age = req.query.age;   // "30"
    res.send(`Name: ${name}, Age: ${age}`);
});
```

2. **Útvonal paraméterek** (req.params)

```
// GET /users/123

app.get('/users/:id', (req, res) => {
    const userId = req.params.id; // "123"
    res.send(`User ID: ${userId}`);
});
```

3. **Törzs (body) paraméterek** (req.body) – JSON és URL-kódolt adatok.

HTML

```
<form method="POST" action="/users">
  <input type="text" name="name">
  <input type="number" name="age">
  <button type="submit">Submit</button>
</form>
```

JS:

```
app.use(express.urlencoded({ extended: true })); // Middleware az URL-kódolt adatok kezeléséhez

app.post('/users', (req, res) => {
    const { name, age } = req.body;
    res.send(`Received user: ${name}, Age: ${age}`);
});
```

4. **Fájlok feltöltése** (multer middleware-rel)

HTML:

```
<form action="/upload" method="POST" enctype="multipart/form-data">
  <input type="file" name="myfile">
  <button type="submit">Upload</button>
</form>
```

JS:

```
const multer = require('multer');
const upload = multer({ dest: 'uploads/' }); // Mappába menti a fájlokat

app.post('/upload', upload.single('myfile'), (req, res) => {
    res.send(`File uploaded: ${req.file.originalname}`);
});
```

5. **Fejléc adatok** (req.headers)

```
app.get('/headers', (req, res) => {
    const userAgent = req.headers['user-agent'];
    res.send(`User Agent: ${userAgent}`);
});
```

6. **Cookie-k fogadása** (cookie-parser middleware-rel)

```
const cookieParser = require('cookie-parser');
app.use(cookieParser());

app.get('/cookies', (req, res) => {
   const myCookie = req.cookies.myCookieName;
   res.send(`Cookie value: ${myCookie}`);
});
```

Gyakorlati feladat:

- Hozz létre egy útvonalat, amely egy URL paraméter alapján fogad be egy adatot, például: /users/:id, és visszaadja az adott felhasználót.
- Készíts egy olyan útvonalat, amely lekérdezési paramétereket (*query parameters*) kezel, például: */search?name=John*.

4 Statikus fájlok kiszolgálása

Hogyan szolgálhatók ki statikus fájlok (HTML, CSS, képek stb.) Express alkalmazásból.

Gyakorlat:

- Készíts egy alap weboldalt, amelynek a HTML fájljait, CSS stílusait és képeit az Express a /public mappából szolgáltatja ki.
- Használd az express.static() middleware-t a statikus fájlok kiszolgálására.
-

5 Routing (útvonalkezelés)

Routing: Tanuld meg, hogyan kezelheted a különböző útvonalakat (app.get(), app.post(), stb.) az Express-ben.

- **Dinamikus útvonalak**: Ismerd meg a dinamikus útvonalak használatát, például hogyan lehet paramétereket átadni az URL-ben (/users/:id).

Gyakorlati feladat:

- Készíts egy szervert, amely különböző útvonalakon különböző adatokat szolgáltat.
- Készíts egy egyszerű "felhasználó" (user) API-t, amelyben a felhasználók adatait egy tömbben tárolod (kezdetben az adatok lehetnek statikusak).
    - GET /users – adjon vissza egy listát az összes felhasználóról.
    - POST /users – fogadjon el egy új felhasználót JSON formátumban és adja hozzá a tömbhöz.

    - GET /users/:id – adja vissza az adott felhasználót a megadott id alapján.
    - PUT /users/:id – frissítse az adott felhasználót az id alapján.
    - DELETE /users/:id – törölje a felhasználót az id alapján.
- Implementálj egyszerű validációkat, például azt, hogy
    - a felhasználó neve kötelező legyen POST esetén.
    - születési dátuma két dátum közötti legyen (18 és 140 év közötti)
    - fizetése a havi minimálbérnél nagyobb legyen (teljes munkaidőben foglalkoztatott munkavállalónak 266 800 Ft, szakképzettséget igénylő munkakörben 326 000 Ft.)
    -

6 REST API készítése

Express remekül alkalmas RESTful API-k készítésére. Tanuld meg, hogyan kezelheted a különböző HTTP metódusokat (GET, POST, PUT, DELETE), és hogyan strukturálhatod API-jaidat.

Gyakorlat:

- **Adatkezelés és JSON válaszok**: Készíts egy REST API-t, amely JSON adatokat szolgáltat.
- Hozz létre egy egyszerű CRUD (*Create, Read, Update, Delete*) alkalmazást, amely pl. felhasználók adatait kezeli.

7 Adatbázis integráció

Adatbázis csatlakoztatása: Tanuld meg, hogyan integrálhatsz adatbázist (pl. MongoDB, MySQL) az Express alkalmazásodba.

ORM használata: Opcionálisan tanulj meg egy ORM-et (pl. Mongoose vagy Sequelize), hogy könnyebben dolgozhass az adatbázissal.

Gyakorlat:

- Hozz létre egy alkalmazást, amely adatokat kér le és tárol egy adatbázisban.
- Integrálj egy MongoDB vagy MySQL adatbázist az alkalmazásba.
- Töltsd le az adatokat az adatbázisból (pl. felhasználói adatokat) és jelenítsd meg őket a REST API-n keresztül.
- Adatok beillesztése és frissítése az adatbázisban a POST és PUT műveletek segítségével.

8 Middleware használata

Middleware koncepciója: Az Express egyik kulcsa a middleware, amely segít a bejövő kérések feldolgozásában (pl. hitelesítés, logolás, hibakezelés).

- **Beépített és saját middleware**: Használd az Express beépített middleware-jeit, mint a express.json() vagy a express.static(), és tanulj meg saját middleware-t készíteni.

Gyakorlat:

- Implementálj egy middleware-t, ami minden kérés előtt naplózza a kérés időpontját.
- Hozz létre egy egyszerű logger middleware-t, amely minden kérésnél kiírja a konzolra az időbélyeget, az útvonalat és a HTTP metódust.
- Készíts egy hibakezelő middleware-t, amely kezeli a nem létező útvonalakat (404-es hibák).
- Alkalmazz egy body-parser middleware-t a JSON adat kezelésére a POST és PUT metódusok esetén.

9 Sablonmotorok használata

Sablonmotorok bevezetése: Express támogat több sablonmotort, mint a **Pug**, **EJS**, **Handlebars**.

Gyakorlat:

- **Sablonmotor beállítása és használata**: Készíts egy egyszerű sablont egy adott motorral, és adj vissza dinamikus tartalmat.
- Integrálj egy templating motort, például **Pug**-ot vagy **EJS**-t az alkalmazásba.
- Készíts egy dinamikus HTML oldalt, amely a szerverről kapott adatokat jeleníti meg *(pl. felhasználók listája*).

10 Hibakezelés és biztonság

- **Hibakezelés**: Tanuld meg, hogyan kezeld a hibákat az Express-ben, mind szinkron, mind aszinkron módon (Promise-ok).
- **Biztonsági intézkedések**: Ismerkedj meg a biztonsági gyakorlattal, pl. CORS kezelése, input validáció, titkosítás.

Gyakorlat:

- Implementálj alapvető hibakezelést és biztonsági middleware-t (pl. Helmet).

11 Autentikáció és jogosultságkezelés

Hitelesítés: Tanuld meg, hogyan implementálhatsz hitelesítést (pl. JSON Web Token vagy session alapú hitelesítés).

- **Jogosultságok kezelése**: Kezeld a felhasználói jogosultságokat a különböző API végpontokon.

Gyakorlat:

- Hozz létre egy bejelentkezési rendszert, ahol felhasználók tokenek segítségével tudnak autentikálni.
- Készíts egy egyszerű bejelentkezési rendszert, ahol a felhasználók regisztrálhatnak és bejelentkezhetnek.
- Használj JWT-t (JSON Web Token) a hitelesítéshez és a védett útvonalakhoz.
- Védj le egyes útvonalakat, hogy csak bejelentkezett felhasználók férhessenek hozzájuk.
- Kezeld a session-öket az **express-session** csomag segítségével.

12 File feltöltés kezelése

Cél: Megismerni, hogyan kezelhetünk fájlokat Express alkalmazásban.

- Implementálj egy fájlfeltöltést, amely lehetővé teszi képek vagy dokumentumok feltöltését a szerverre a **multer middleware** segítségével.
- A feltöltött fájlokat tárold egy dedikált mappában, és jelenítsd meg őket egy oldalon.

13 Paginating és Sorting egy API-ban

Cél: Adatok lapozásának és rendezésének kezelése.

- Készíts egy útvonalat, ahol nagy mennyiségű adatot kell lapozni (pl. felhasználók listája), és adj hozzá lapozást (pagination).
- Add hozzá a rendezési (sorting) lehetőséget egy lekérdezési paraméter alapján (pl. ?sort=name).

14 Tesztelés és telepítés

- **Tesztelés**: Tanuld meg, hogyan tesztelheted az Express alkalmazásod automatikusan (pl. Mocha, Chai vagy Jest segítségével).
- **Telepítés**: Ismerkedj meg a telepítési folyamatokkal, pl. hogyan telepíthetsz alkalmazást Heroku-ra vagy más cloud platformra.

Gyakorlat:

- Írj teszteket az API végpontjaidhoz, majd telepítsd az alkalmazásodat egy felhőszolgáltatóra.

15 További források: