G1F

Ich kann eine Anbindung an eine NoSQL Datenbank implementieren. (z. B. API)

Fragenstellung und Lernziele

In dieser Uebung geht es darum, eine minimalistische API in TypeScript und Express zu erstellen, die eine Verbindung zu einer MongoDB herstellt. Dabei sollen folgende Lernziele erreicht werden:

- Einrichten eines Docker-Containers mit aktivierter Authentifizierung
- Importieren von CSV-Daten in eine NoSQL-Datenbank
- Konfigurieren einer Datenbank (testdb) und einer Collection (people) mit den Feldern name, age, city
- Implementieren und Testen von REST-Endpunkten (GET und POST)
- Verifizieren von Lese- und Schreibrechten anhand eines Read-Only-Benutzers

Umsetzung

Container Setup

Docker-compose Datei erstellen:

```
services:
  mongodb:
    image: mongo:8.0
    container_name: products
    command: ["mongod", "--auth"]
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: secret123
    ports:
      - "27018:27017"
    volumes:
      - ./mongo-data:/data/db
      - ./products.csv:/tmp/products.csv:ro
    restart: unless-stopped
```

Container starten:

```
docker-compose up -d
```

CSV-Daten:

```
productId,name,category,price
P001,Smartphone,Electronics,699.99
P002,Laptop,Electronics,1299.00
P003,Desk Chair,Furniture,149.50
P004,Coffee Mug,Kitchen,12.99
```

Import CSV-Daten in MongoDB:

```
docker exec products mongoimport `
  --username admin `
  --password secret123 `
  --authenticationDatabase admin `
  --db productsdb `
  --collection products `
  --type csv `
  --headerline `
  --file /tmp/products.csv
```

Typescript API

Wir wollen folgende Endpunkte implementieren:

```
GET /people - liefert alle Dokumente

GET /people/:id – liefert ein Dokument per ObjectId

POST /people – legt ein neues Dokument an
```

Projektsetup

```
pnpm init -y
pnpm add express mongodb dotenv
pnpm add -D typescript @types/express @types/node
pnpm tsc --init
```

`tsconfig.json` Datei erstellen:

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "CommonJS",
    "rootDir": "src",
    "outDir": "dist",
```

```
    "strict": true,
    "esModuleInterop": true
  }
}
```

`.env` Datei erstellen:

```
MONGO_URI=mongodb://admin:secret123@localhost:27018
DB_NAME=productsdb
PORT=3001
```

API-Code:

```
import express, { Request, Response, NextFunction } from 'express';
import { MongoClient, ObjectId } from 'mongodb';
import dotenv from 'dotenv';

dotenv.config();

const uri = process.env.MONGO_URI;
const dbName = process.env.DB_NAME || 'testdb';
const port = Number(process.env.PORT || 3000);

if (!uri) {
  console.error('Fehler: MONGO_URI nicht gesetzt');
  process.exit(1);
}

const client = new MongoClient(uri, {});

async function main() {
  try {
    await client.connect();
    console.log('✅ Verbunden mit MongoDB');

    const db = client.db(dbName);
    const people = db.collection('people');

    const app = express();
    app.use(express.json());

    // GET /people - alle Datensätze ausgeben
    app.get('/people', async (req: Request, res: Response, next: NextFunc
      try {
        const docs = await people.find().toArray();
        res.json(docs);
      } catch (err) {
        next(err);
      }
```

```
    });

    // GET /people/:id – einzelnes Dokument per ID
    app.get('/people/:id', async (req: Request, res: Response, next: Next
      try {
        const { id } = req.params;
        if (!ObjectId.isValid(id)) {
          return res.status(400).json({ error: 'Ungültige ID' });
        }
        const doc = await people.findOne({ _id: new ObjectId(id) });
        if (!doc) {
          return res.status(404).json({ error: 'Nicht gefunden' });
        }
        res.json(doc);
      } catch (err) {
        next(err);
      }
    });

    // POST /people – neuen Datensatz anlegen
    app.post('/people', async (req: Request, res: Response, next: NextFun
      try {
        const { name, age, city } = req.body;
        if (!name || typeof age !== 'number' || !city) {
          return res.status(400).json({ error: 'Fehlende Felder oder fals
        }
        const result = await people.insertOne({ name, age, city });
        res.status(201).json({ insertedId: result.insertedId });
      } catch (err) {
        next(err);
      }
    });

    // globale Error-Middleware
    app.use((err: Error, req: Request, res: Response, _next: NextFunction
      console.error(err);
      res.status(500).json({ error: 'Interner Serverfehler' });
    });

    app.listen(port, () => {
      console.log(`🚀 Server läuft auf http://localhost:${port}`);
    });
  } catch (err) {
    console.error('Verbindungsfehler:', err);
    process.exit(1);
  }
}

main();
```

`package.json` Skripts ergänzen:

```
// innerhalb von "scripts":
"build": "tsc",
"start": "node dist/index.js"
```

```
pnpm build
pnpm start
```

Nachweis

Praktische Übung:

```
Ⓐ Oliver ■ ~/../school/m165/165 ○ ᚠG1F
# curl http://localhost:3001/products | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   443  100   443    0     0  32713      0 --:--:-- --:--:-- --:--:-- 34076
[
  {
    "_id": "681c9da20ab742495142d33c",
    "productId": "P002",
    "name": "Laptop",
    "category": "Electronics",
    "price": 1299
  },
  {
    "_id": "681c9da20ab742495142d33d",
    "productId": "P004",
    "name": "Coffee Mug",
    "category": "Kitchen",
    "price": 12.99
  },
  {
    "_id": "681c9da20ab742495142d33e",
    "productId": "P003",
    "name": "Desk Chair",
    "category": "Furniture",
    "price": 149.5
  },
  {
    "_id": "681c9da20ab742495142d33f",
    "productId": "P001",
    "name": "Smartphone",
    "category": "Electronics",
    "price": 699.99
  }
]
```