5. NodeJS házi feladat

Dr. Simon Balázs (sbalazs@iit.bme.hu), BME IIT, 2023.

A továbbiakban a **neptun** szó helyére a saját Neptun-kódot kell behelyettesíteni, csupa **kisbetűkkel** (a NodeJS projektnévben nem szereti a nagybetűket).

1 A feladat leírása

A feladat egy filmadatbázis rendszer elkészítése, és ennek RESTful szolgáltatásként való publikálása.

A szolgáltatás által kapott és visszaadott adatstruktúrákat az alábbi TypeScript-kód írja le:

```
export interface IMovie
{
    title: string;
    year: number;
    director: string;
    actor: string[];
}

export interface IMovieList
{
    movie: IMovie[];
}

export interface IMovieId
{
    id: number;
}

export interface IMovieIdList
{
    id: number[];
}
```

2 A szolgáltatás

A szolgáltatást egy NodeJS alkalmazásban kell megvalósítani. Az alkalmazást pontosan ugyanazokkal a lépésekkel kell létrehozni, mint amelyek a tutorial 2. és 3. fejezetében szerepelnek, azzal a különbséggel, hogy az alkalmazás neve: **node_neptun** (a saját Neptun-kód csupa **kisbetűkkel** szerepeljen). A tutorial-tól eltérő modulok használata tilos!

Csak az **src** és az **app** könyvtárban szabad módosításokat végezni. Az implementációs nyelv szabadon választható: lehet TypeScript nyelven dolgozni az **src** könyvtáron belül, vagy natív JavaScript nyelven az **app** könyvtáron belül.

A szolgáltatásnak adatokat kell tárolnia az egyes hívások között egy MongoDB adatbázisban:

- a MongoDB adatbázis neve: **neptun**
- az adatbázisban a kollekció neve: **Movies**

A szolgáltatás báziscíme a következő URL:

`http://localhost:3000`

A szolgáltatás funkcionalitása ugyanaz, mint a 2. REST házi feladatban szereplő szolgáltatásé. A szolgáltatás megvalósításához szükséges üzenetek formátuma is ugyanaz, mint a 2. REST háziban szerepelt, de most csak a JSON formátumot kell támogatni. A fenti TypeScript üzenetek JSON-sorosított változata éppen ezeket adja.

A szolgáltatás tehát a következő hívásokat támogatja a fenti báziscímtől számítva:

- **`GET /movies`**
  - bemenet HTTP body: nincs
  - kimenet HTTP body: **`IMovieList`**
- **`GET /movies/{id}`**
  - bemenet HTTP body: nincs
  - kimenet HTTP body: **`IMovie`**
- **`POST /movies`**
  - bemenet HTTP body: **`IMovie`**
  - kimenet HTTP body: **`IMovieId`**
- **`PUT /movies/{id}`**
  - bemenet HTTP body: **`IMovie`**
  - kimenet HTTP body: nincs
- **`DELETE /movies/{id}`**
  - bemenet HTTP body: nincs
  - kimenet HTTP body: nincs
- **`GET /movies/find?year={year}&orderby={field}`**
  - bemenet HTTP body: nincs
  - kimenet HTTP body: **`IMovieIdList`**

3 A kliens

Beadandó klienst nem kell készíteni. Saját célra készíthető kliens, de azt nem kell beadni.

4 Beadandók

Beadandó egyetlen ZIP fájl. Más tömörítési eljárás használata tilos!

A ZIP fájl gyökerében egyetlen könyvtárnak kell lennie, a szolgáltatás alkalmazása:

- **node_neptun:** a szolgáltatás NodeJS projektje teljes egészében

Mielőtt a projektet becsomagolnánk a ZIP-be, a projekten belül töröljük a **node_modules** könyvtárat! Ez a könyvtár elég nagy és egyébként sem lesz figyelembe véve a kiértékelés során.

Az alkalmazásnak parancssorból futnia kell a következő parancsok kiadása után:

```
...\node_neptun>npm install
```

```
...\node_neptun>npm start
```

A TypeScript kódokat még a feltöltés előtt le kell fordítani JavaScript kódra, amennyiben valaki a TypeScript nyelvet választja!

A szolgáltatásnak a 2. REST háziban szereplő JSON formátumokat kell támogatnia. Más JSON formátum/elnevezés használata tilos!

Fontos: a dokumentumban szereplő elnevezéseket és kikötéseket pontosan be kell tartani!

Még egyszer kiemelve: a **neptun** szó helyére mindig a saját Neptun-kódot kell behelyettesíteni, csupa **kisbetűkkel**!